C, C++, Java, Python, PHP, JavaScript and Linux For Beginners

"The only true wisdom is in knowing you know nothing."

— Socrates

Manjunath.R

#16/1, 8th Main Road, Shivanagar, Rajajinagar, Bangalore560010, Karnataka, India

*Corresponding Author Email: manjunath5496@gmail.com

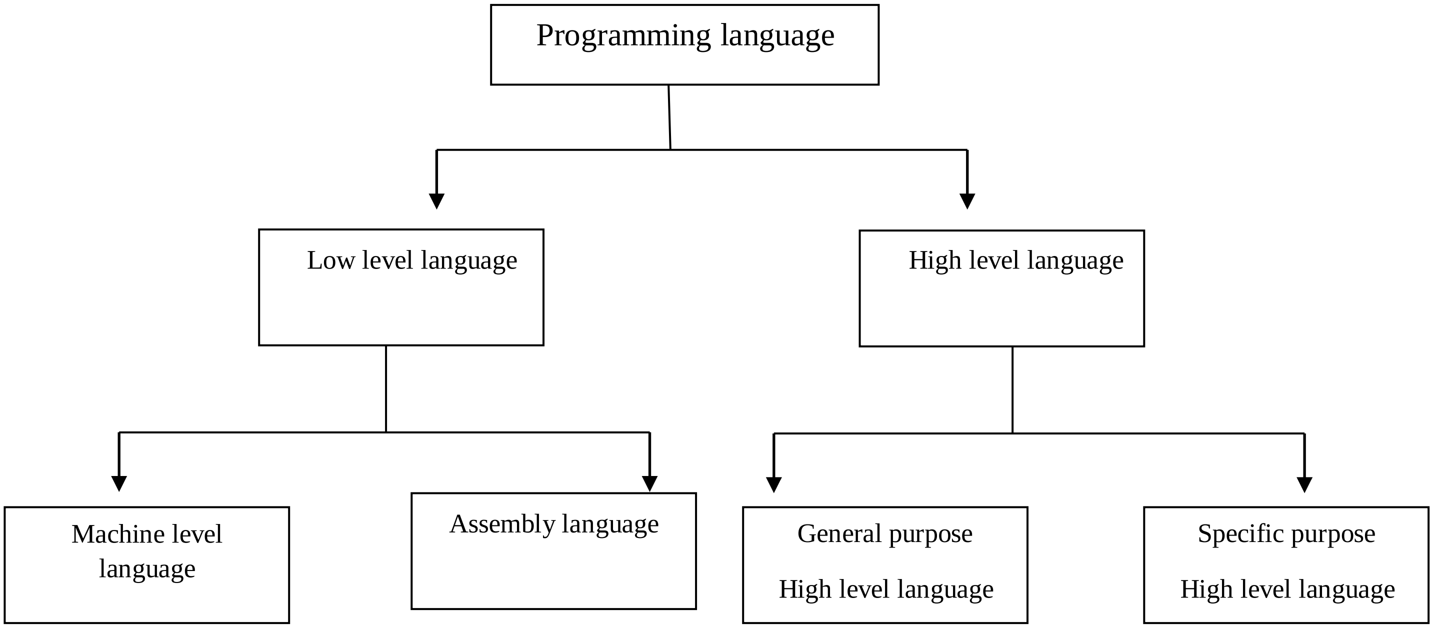
*Website: http://www.myw3schools.com/

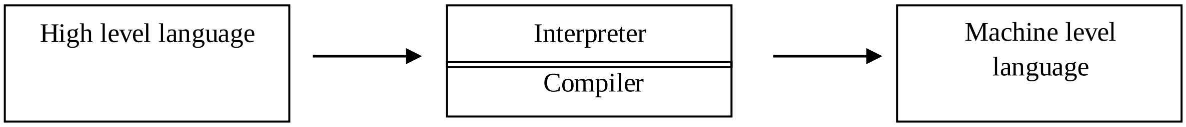
- Educational institutions are teaching it
- Corporate societies are employing it
- Pupils need it
- (Pedagogues desire it... ;)
- (Coders perceive it... :)

This BOOK highlights some basic knowledge of artificial intelligence formal constructed programming languages (like C, C++, Java, PHP, Python, JavaScript and XML) designed to interact with the hardware in a more powerful way and to communicate instructions to a machine, particularly an electronic device which is capable of receiving information (data) in a particular form and of performing a sequence of operations in accordance with a predetermined but variable set of procedural instructions (program) to produce a result in the form of information or signals. If you read this book in a public place (on a commuter train, at the beach, or on the dance floor at the Restaurants, for example), you can read proudly, with a chip on your shoulder and with your head held high. C, C++, Java, PHP, Python and JavaScript are hot stuff, and you're cool because you're reading about it. If you are just learning what kind of animals C, C++, Java, PHP, Python and JavaScript are, this BOOK will make an excellent companion to any tutorial and serve as a source of knowledge to your specific questions. And, by reading this BOOK, you'll have a broad, basic knowledge of C, C++, Java, Python, JavaScript [scripting language used in millions of Web pages such as Internet Explorer, Firefox, Chrome, Opera, and Safari] and PHP. This book is for all programmers, whether you are a novice or an experienced pro. The beginner will find its carefully paced discussions and many examples especially helpful. Of course those who have already familiar with programming are likely to derive more benefits from this book. After completing this book you will find yourself at a moderate level of expertise in C, C++, Java, PHP, Python and JavaScript programming from where you can take yourself to next levels.

"What Will Understanding C, C++, Java, PHP, Python and JavaScript (the world's most popular case-sensitive object oriented scripting language introduced to create dynamic web pages) Do for Me?"

C, C++, Java, PHP, Python, JavaScript and Java programs are commonly referred to as software and this software is essential to a fast and obedient, smart processing unit called computer because it controls everything the computer does (i.e., performance of a suite of computer operations and storage of the results in its memory can be manipulated by using notable programming languages in existence such as C, C++ and Java etc.). Contemplating **Android application development** is a great choice as per current market scenario and the importance of Android application development for businesses of today cannot be emphasized on enough. Android is everywhere. In a study that spans the Americas, Europe, Asia, and the Middle East, GlobalWebIndex reports that Android tablets outnumber iPads by more than 34 million. More than a million apps are available for download at the digital distribution platform operated by Google (double the number of apps that were available in the last few years). And more than 9 million developers write code using Java, XML− the languages that empowers an array of software intended for mobile devices that features an operating system, core applications and middleware. Nowadays, website is considered as the window to the world of internet and no company can think about making its business and marketing requirements big without having its very own website to get in touch with millions of web surfers all over the world including their potential customer and the global client base and moreover website serves as the primary step towards having better business sensibility and drives the company to market the services and products they offer. Javascript is a full-fledged dynamic scripting programming language, integrated with HTML document, to provide dynamic interactivity on websites (without which there is no fastest and most efficient way to deliver the content of the website) and it presents the Front End (The representation part, what user sees) and Back End (Controlling of all the requests, made by users) is fulfilled by scripting languages (like PHP, Python etc.).





C Language Basic Syntax Rules

```
#include<stdio.h>  →  Including header files
int main()  →  main()function must be there
{
// Hello World Program  →  Single line comment
printf("Hello,world!");  →  Semicolon after each statement
return 0;
}  →  Program enclosed by curly braces
```

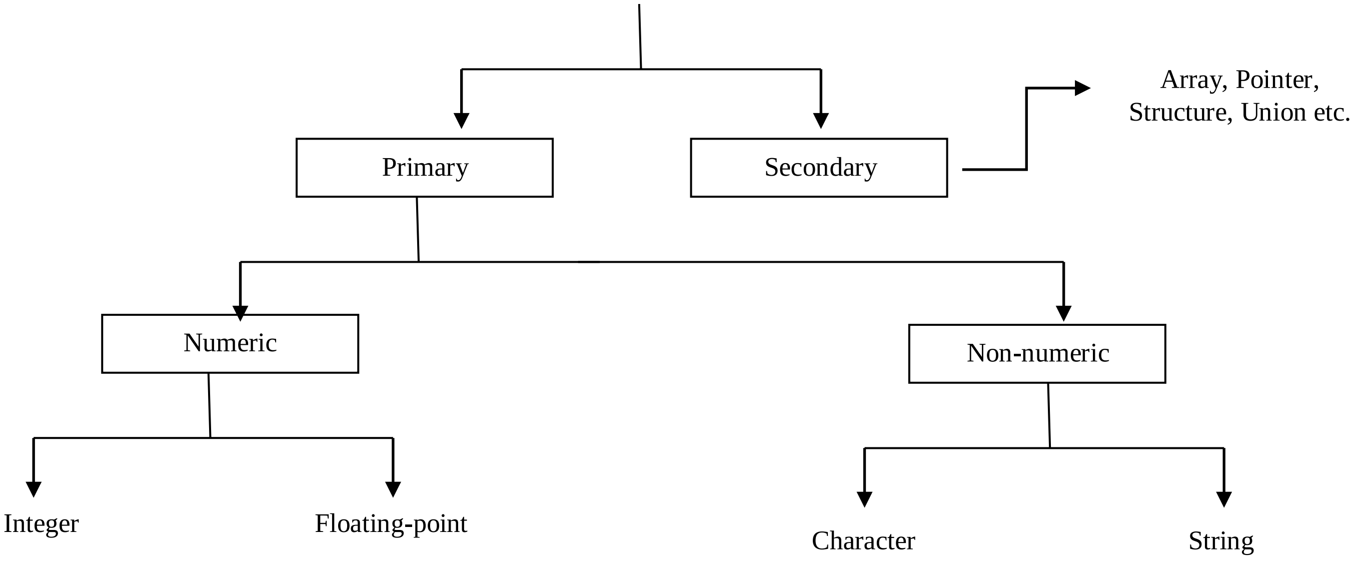
C Keywords

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	unsigned

Special Characters in C Programming

,	<	>	.	_
(	)	;	$	:
%	[	]	#	?
'	&	{	}	"
^	!	*	/	|
-	\	~	+	

C Constants



- **Integer** → 246, 0, -3679, +35, 9777, -36026 etc.
- **Floating-point** → 246.23, 0000.23, -36.79, +35.56, 9.777, -360.216 etc.
- **Character → 'b', '?', '@', '#' etc.**
- **String** → "C", "Java", "Python", "JavaScript" etc.

Data types:

Types	Size in bytes	Keyword
Integer	2	int
Floating-point	4	float
Double	8	double
Character	1	char

Escape Sequences:

Sequence	Meaning
\a	System alarm (bell)
\b	Backspace
\f	Form feed
\n	Newline
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab
\\	Backslash
\'	Single quote
\"	Double quote
\?	Question mark
\0	End of string

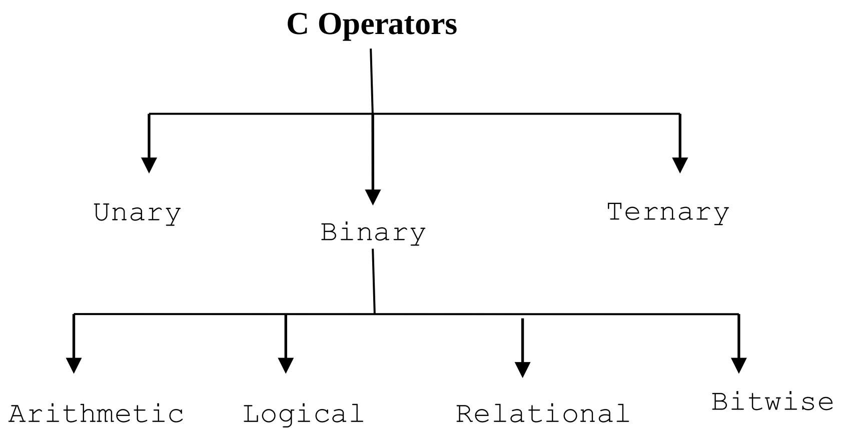
Input output functions:

	Input	Output
Formatted	scanf()	printf()

Unformatted	getchar()
	gets()
		putchar()
		puts()

Format specifiers in C

Format Specifier	Meaning
%c	Read a Single Character
%d	Read a Decimal integer
%e	Read a Floating-point number
%f	Read a Floating-point number
%g	Read a Floating-point number
%h	Read a short integer
%i	Read a Decimal or hexadecimal or octal number
%o	Read an octal number
%p	Read a pointer
%s	Read a string
%u	Read an Unsigned integer
%x	Read a hexadecimal number
%n	Prints nothing
%%	Prints % character



- **Arithmetic operators**

Operations	Operator
Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus	%

- **Relational operators**

Operator	Meaning
<	Lesser than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to

- **Logical operators**

Operator	Meaning
&&	Logical AND
||	Logical OR
!	Logical NOT

- **Bitwise operators**

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Exclusive –OR (XOR)
~	1's complement
<<	Left shifting of bits
>>	Right shifting of bits

Shorthand operators

Operator	Example	Equivalent construct
+	a=a+b	a+=b
-	a=a-b	a-=b
*	a=a*b	a*=b
/	a=a/b	a/=b
%	a=a%b	a%=b
&	a=a&b	a&=b
|	a=a|b	a|=b
^	a=a^b	a^=b
<<	a=a<<b	a<<=b
>>	a=a>>b	a>>=b

Mathematical Functions

Function	**Description**
sqrt(x)	Return the square root of x
exp(x)	Return exponential (e^x)
log(x)	Return natural logarithm of x (base e)
log10(x)	Return logarithm of x (base 10)

fabs(x)
abs(x)
	Return absolute value of x

floor(x)	Return a value rounded to the next lower integer
pow(x,y)	Return x raised to power y (x^y)
fmod(x,y)	Return floating-point remainder of x/y (with same sign of x)
sin(x)	Return the sine of x
cos(x)	Return the cosine of x
tan(x)	Return the tangent of x
acos(x)	Return arc cosine of x [$\cos^{-1}(x)$]
asin(x)	Return arc sine of x [$\sin^{-1}(x)$]
atan(x)	Return arc tangent of x [$\tan^{-1}(x)$]

C Header files

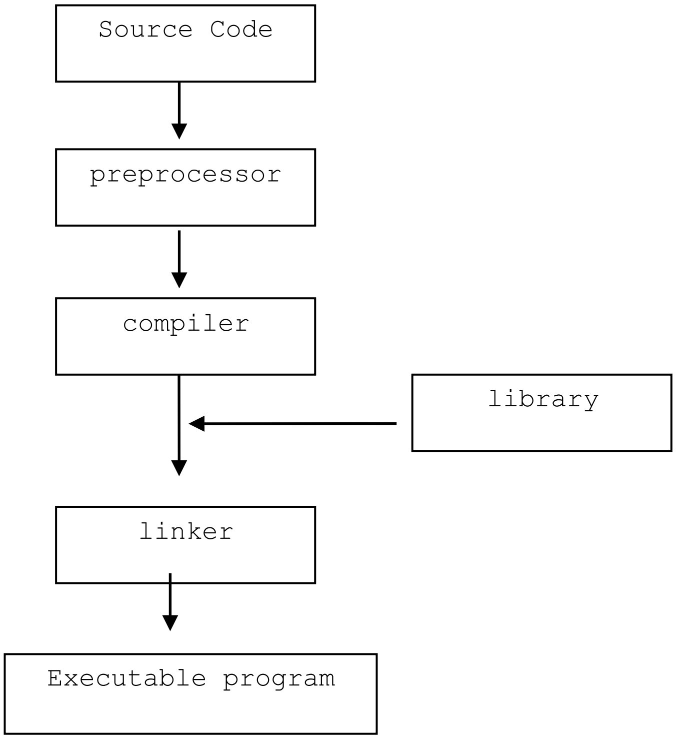
Header file	Description
stdio.h	Input/Output functions
conio.h	Console Input/Output functions
stdlib.h	Some standard library functions
math.h	Mathematical functions
string.h	String manipulation functions
ctype.h	Character handling functions
time.h	Time computing functions
malloc.h	Memory allocation / deallocation functions
graphics.h	Graphical functions
dos.h	Function linking DOS routines
wctype.h	Functions to classify and transform individual wide characters
limits.h	Functions define various symbolic names
float.h	Functions define set of various platform-dependent constants related to floating point values.

Preprocessor Directives

Preprocessor directive	Use
#define	To define a macro
#include	Inserts a particular header from another file.
#if	To test whether a compile –time condition is true
#undef	To undefine a macro
#else	To specify the alternative action if a test fails

#endif	To end the preprocessor condition

Development of C Program:



Basic structure of C Program

```
preprocessor statements
global declaration;
main()
{
declarations;
statements;
}
user defined function
```

ASCII Table

ASCII	Char	ASCII	Char	ASCII	Char	ASCII	Char
0	NUL (null)	32	SPACE	64	@	96	`
1	SOH (start of heading)	33	!	65	A	97	a
2	STX (start of text)	34	"	66	B	98	b
3	ETX (end of text)	35	#	67	C	99	c
4	EOT (end of transmission)	36	$	68	D	100	d
5	ENQ (enquiry)	37	%	69	E	101	e
6	ACK (acknowledge)	38	&	70	F	102	f
7	BEL (bell)	39	'	71	G	103	g
8	BS (backspace)	40	(	72	H	104	h
9	TAB (horizontal tab)	41	)	73	I	105	i
10	LF (NL line feed, new line)	42	*	74	J	106	j
11	VT (vertical tab)	43	+	75	K	107	k
12	FF (NP form feed, new page)	44	,	76	L	108	l
13	CR (carriage return)	45	-	77	M	109	m
14	SO (shift out)	46	.	78	N	110	n
15	SI (shift in)	47	/	79	O	111	o
16	DLE (data link escape)	48	0	80	P	112	p
17	DC1 (device control 1)	49	1	81	Q	113	q
18	DC2 (device control 2)	50	2	82	R	114	r
19	DC3 (device control 3)	51	3	83	S	115	s
20	DC4 (device control 4)	52	4	84	T	116	t
21	NAK (negative acknowledge)	53	5	85	U	117	u
22	SYN (synchronous idle)	54	6	86	V	118	v
23	ETB (end of trans. block)	55	7	87	W	119	w
24	CAN (cancel)	56	8	88	X	120	x
25	EM (end of medium)	57	9	89	Y	121	y
26	SUB (substitute)	58	:	90	Z	122	z
27	ESC (escape)	59	;	91	[	123	{
28	FS (file separator)	60	<	92	\	124	|
29	GS (group separator)	61	=	93	]	125	}
30	RS (record separator)	62	>	94	^	126	~
31	US (unit separator)	63	?	95	_	127	DEL

Data Conversion Functions

Function	Use
atof()	Converts string to float
atoi()	Converts string to int
atol()	Converts string to long
ecvt()	Converts double to string
fcvt()	Converts double to string
gcvt()	Converts double to string
itoa()	Converts int to string
ltoa()	Converts long to string
strtod()	Converts string to double
strtol()	Converts string to long integer
strtoul()	Converts string to an unsigned long integer
ultoa()	Converts unsigned long to string

Character Classification Functions

Function	Use
isalnum()	Tests for alphanumeric character
isalpha()	Tests for alphabetic character
isdigit()	Tests for decimal digit
islower()	Tests for lowercase character
isspace()	Tests for white space character
isupper()	Tests for uppercase character
isxdigit()	Tests for hexadecimal digit
tolower()	Tests character and converts to lowercase if uppercase
toupper()	Tests character and converts to uppercase if lowercase

String Manipulation Functions

Function	Use
strchr()	Appends one string to another
strchr()	Finds first occurrence of a given character in a string
strcmp()	Compares two strings
strcmpi()	Compares two strings without regard to case
strcpy()	Copies one string to another
strdup()	Duplicates a string
stricmp()	Compares two strings without regard to case (identical to strcmpi)
strlen()	Finds length of a string
strlwr()	Converts a string to lowercase
strncat()	Appends a portion of one string to another
strncmp()	Compares a portion of one string with portion of another string
strncpy()	Copies a given number of characters of one string to another
strnicmp()	Compares a portion of one string with a portion of another without regard to case
strrchr()	Finds last occurrence of a given character in a string
strrev()	Reverses a string
strset()	Sets all characters in a string to a given character
strstr()	Finds first occurrence of a given string in another string
strupr()	Converts a string to uppercase

Searching and Sorting Functions

Function	Use
bsearch()	Performs binary search
lfind()	Performs linear search for a given value
qsort()	Performs quick sort

File Handling Functions

Function	Use
fopen()	opens new or existing file
fprintf()	write data into the file
fscanf()	reads data from the file
fputc()	writes a character into the file
fgetc()	reads a character from file
fclose()	closes the file
fseek()	sets the file pointer to a given position
fputw()	writes an integer to file
fgetw()	reads an integer from file
ftell()	returns current position
rewind()	sets the file pointer to the beginning of the file

Memory allocation Functions

Function	Use
malloc()	allocates the specified number of bytes

realloc()	increases or decreases the size of the specified block of memory, moving it if necessary
calloc()	allocates the specified number of bytes and initializes them to zero
free()	releases the specified block of memory back to the system

Directory Control Functions

- chdir() = Changes current working directory
- getcwd() = Gets current working directory
- fnsplit() = Splits a full path name into its components
- findfirst() = Searches a disk directory
- findnext() = Continues findfirst search
- mkdir() = Makes a new directory
- rmdir()= Removes a directory

Algorithm → a step by step procedure to solve a particular problem

Example:

Algorithm to compute the area of a sphere

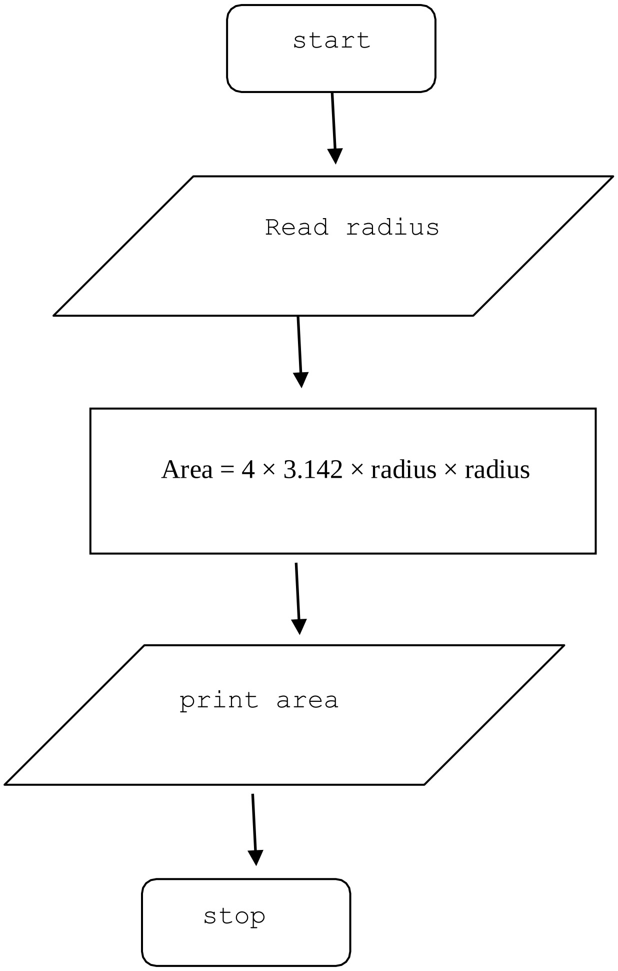
Step 1: Read radius

Step2: Compute Area [Area = 4 × 3.142 × radius × radius]

Step3: Print the Area

Step4: Stop [End of algorithm]

Flowchart → a diagrammatic representation of an algorithm



C Program

```
#include<stdio.h>

int main()

{

int radius, Area;

radius = 2;

area = 4 * 3.14 * r * r;

printf("Area of a sphere = %d", Area);

return 0;

}
```

Logical statements

```
>     Greater than              7 > 3 is TRUE
<     Less than                 3 < 7 is TRUE
>=    greater than or equal     9 >= 9 is TRUE
<=    less than or equal        3 <= 7 is TRUE
==    Equal to                  14 == 14 is TRUE
!=    not equal to              7 != 3 is TRUE
```

Basic If Syntax

```
if (condition) {
  Execute all statements inside the body if the condition is true
}
```

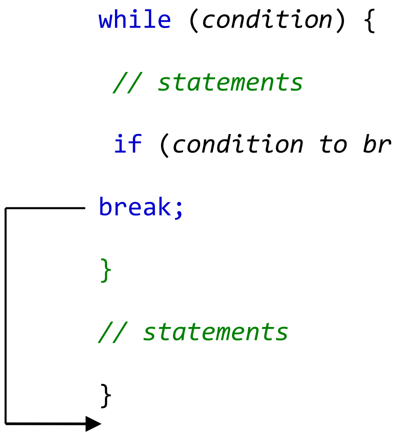
Else

```
if (condition) {
  Execute all statements inside the body if the condition is true
} else {
  Execute all statements inside the body if the condition is false
}
```

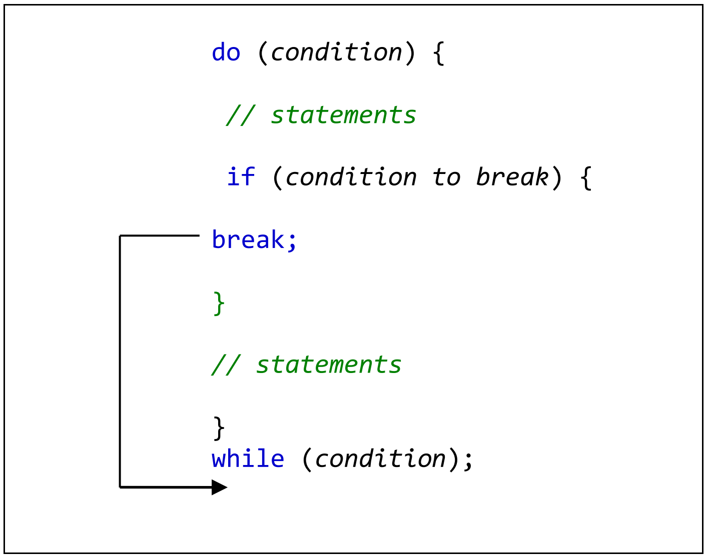
Else if

```
if (condition1) {
  Execute all statements inside the body if condition1 is true
} else if (condition2) {
  Execute all statements inside the body if the condition1 is false and condition2 is
true
} else {
  Execute all statements inside the body if the condition1 is false and condition2 is
false
}
```

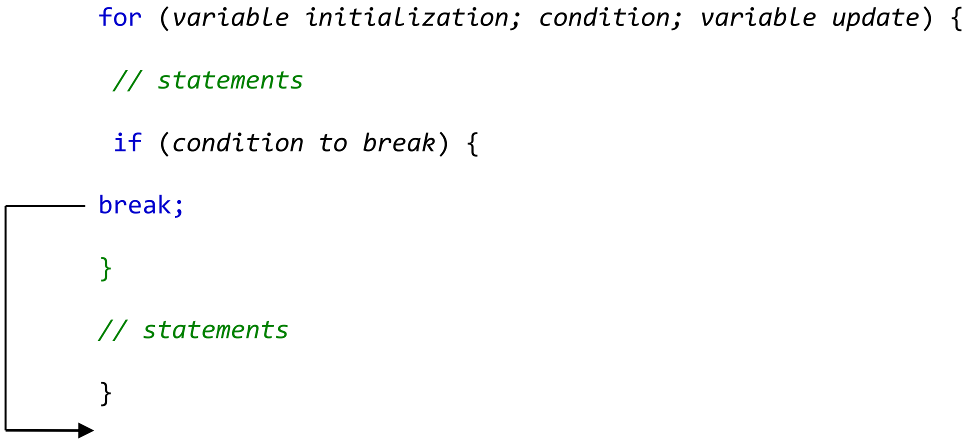
Break



```
while (condition) {
 // statements
 if (condition to break) {
break;
}
// statements
}
```

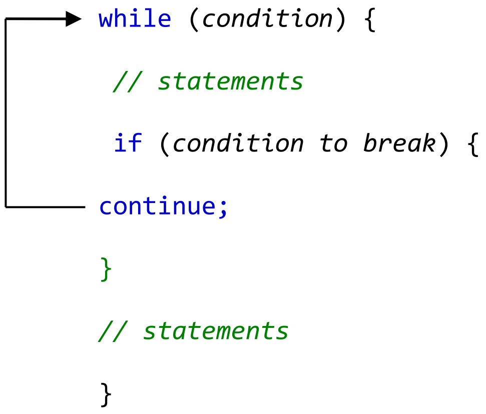


```
do (condition) {
 // statements
 if (condition to break) {
break;
}
// statements
}
while (condition);
```

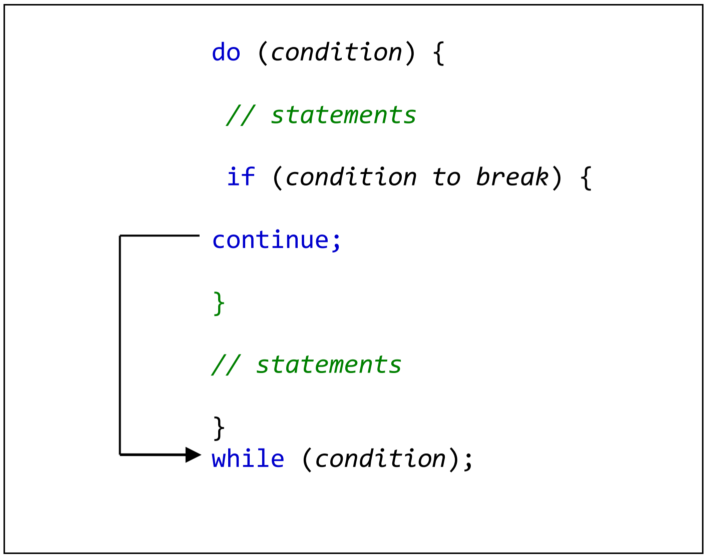


```
for (variable initialization; condition; variable update) {
 // statements
 if (condition to break) {
break;
}
// statements
}
```

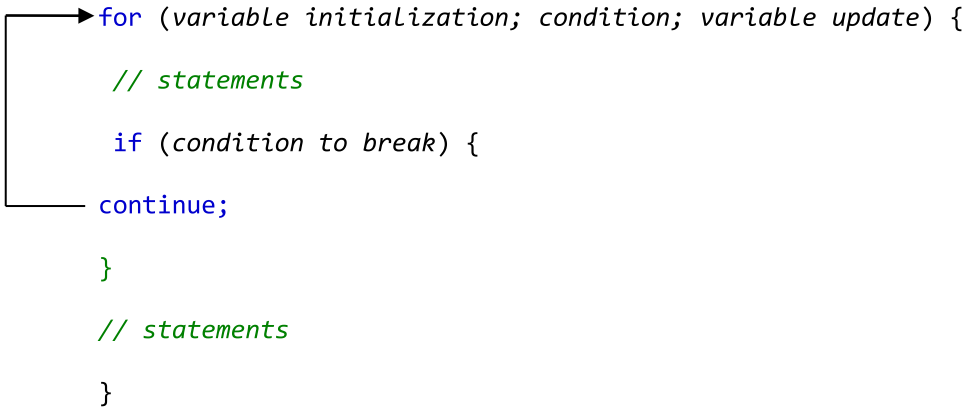
Continue



```
while (condition) {
 // statements
 if (condition to break) {
continue;
}
// statements
}
```



```
do (condition) {
 // statements
 if (condition to break) {
continue;
}
// statements
}
while (condition);
```



```
for (variable initialization; condition; variable update) {
 // statements
 if (condition to break) {
continue;
}
// statements
}
```

Switch case

```
switch (expression) {

  case 1:
    // statements

    break;

  case 2:
    // statements

    break;

  default:
    // statements

}
```

Loops

- **for Loop**

```
for (variable initialization; condition; variable update) {

 // statements

}
```

- **while Loop**

```
while (condition) {

 // statements

}
```

- **do while Loop**

```
do {
 // statements
}
while (condition);
```

> "The computer programmer is a creator of universes for which he alone is the lawgiver. No playwright, no stage director, no emperor, however powerful, has ever exercised such absolute authority to arrange a stage or field of battle and to command such unswervingly dutiful actors or troops."
>
> **— Joseph Weizenbaum**

Example of *goto* statement:

```
#include <stdio.h>
int main()
{
   int sum=0;
   for(int i = 1; i<=9; i++){
  sum = sum+i;
  if(i==5){
     goto addition;
  }
   }

   addition:
   printf("%d", sum);

   return 0;
}
```

Example of *nested-if* statement:

```
#include <stdio.h>

int main() {
    int a = 10;

    if (a == 10)
    {
        if (a < 15)
        printf("a is smaller than 15");

        if (a < 12)
            printf("a is smaller than 12");
        else
            printf("a is greater than 15");
    }

    return 0;
}
```

Example of *nested else if Statement*:

```
#include <stdio.h>

int main() {
    int a = 20;

    if (a == 10)
        printf("a is 10");
    else if (a == 15)
        printf("a is 15");
    else if (a == 20)
        printf("a is 20");
    else
        printf("a is not present");
}
```

C Programming

Paradigm	Imperative (procedural), structured
Designed by	Dennis Ritchie
Developer	Dennis Ritchie & Bell Labs (creators); ANSI X3J11 (ANSI C); ISO/IEC JTC1/SC22/WG14 (ISO C)
First appeared	1972; 48 years ago
Stable release	C18 / June 2018; 2 years ago
Typing discipline	Static, weak, manifest, nominal
OS	Cross-platform
Filename extensions	.c, .h
Major implementations	
K&R C, GCC, Clang, Intel C, C++Builder, Microsoft Visual C++, Watcom C	
Dialects	
Cyclone, Unified Parallel C, Split-C, Cilk, C*	
Influenced by	
B (BCPL, CPL), ALGOL 68, Assembly, PL/I, FORTRAN	
Influenced	
Numerous: AMPL, AWK, csh, C++, C--, C#, Objective-	

```
C, D, Go, Java, JavaScript, Julia, Limbo, LPC, Perl, PHP, Pike, Processing, Python, Ring, Rust
                         , Seed7, Vala, Verilog (HDL), Nim
```

A general-purpose, procedural, High Level Programming most widely used computer Language (which uses alphabets, digits, punctuations and some special symbols and cannot be executed directly without being converted into machine level language [the language which uses only 0 and 1]) developed by a man named **Dennis Ritchie** in 1970s at **Bell Telephone laboratories (now named AT & T Bell laboratories), Murray Hill, New Jersey** to develop system application that directly interacts with the hardware devices such as drivers, kernels, etc. **[UNIX operating system]** using the two early programming languages − **Basic Combined Programming Language (BCPL)** and **BASIC (Beginner's All-purpose Symbolic Instruction Code) language.**

Uses: used in the development of

- Operating systems like **LINUX**, UNIX.
- CAD/ CAM Applications and Word processors
- Embedded systems like **ATMs**, printers.
- RDBMS MySQL, Language Compilers and Interpreters, Print Spoolers, Loaders, Linkers, Assemblers, Text Editors, Automation tools, Network Drivers.

Most of the state-of-the-art software has been developed using C. C has also greatly influenced many other popular languages [considered as the base for other programming languages − mother language], especially C++, which was originally designed as an enhancement to C. Most of the programming languages follow C syntax [For example: C++, Java, C#, etc.]

Advantages:

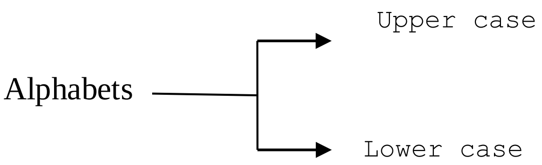
- Relatively simple language
- Reliable (able to be trusted)

- Easy to understand and supports a rich set of data types
- Easy to use, write, modify and debug and quick to learn
- can be compiled on a variety of computer platforms
- C supports a rich set of operators:
    - The unary minus,!, ++, -- and ~ → unary operators
    - The +, − *, ? are arithmetic operators. % operator is another arithmetic operator which results in the remainder after an integer division.
    - The <, <=, >=< = =, != are relational operators.
- C supports **bitwise operators** – Bitwise AND, OR, XOR, bitwise complement, left and right shift operators.
- C provides a variety of conditional control statements such as
- *if*-statement
- *if-else* statement
- *nested-if* statement
- *switch* statement

C [The ALGOL-based language formalized in 1988 by the American National Standard Institute (ANSI) and can handle low-level activities] is called a **structured programming language** because it divides the problem into smaller modules called functions or procedures each of which handles a particular responsibility. Hence it is simple and easy to understand and well suited for small size implementation. However this is not restricted. A large size implementation is possible but complex design and full **object oriented design** cannot be implemented (because complex design concepts like **Polymorphism** and **inheritance** are not available in C).

C is widely used in IOT applications.

Characters Set:



Alphabets

Upper case

Lower case

Digits [0 – 9]

Special characters

~	Tilde
!	Exclamation mark
#	Number sign
$	Dollar sign
%	Percent sign
^	Caret
&	Ampersand
*	Asterisk
(	Lest parenthesis
)	Right parenthesis
_	Underscore
+	Plus sign
|	Vertical bar
\	Backslash
`	Apostrophe
-	Minus sign
=	Equal to sign
{	Left brace

}	Right brace
[	Left bracket
]	Right bracket
:	Colon
"	Quotation mark
;	Semicolon
<	Opening angle bracket
>	Closing angle bracket
?	Question mark
,	Comma
.	Period
/	Slash

White spaces

- Blank space
- New line
- Carriage return
- Horizontal tab

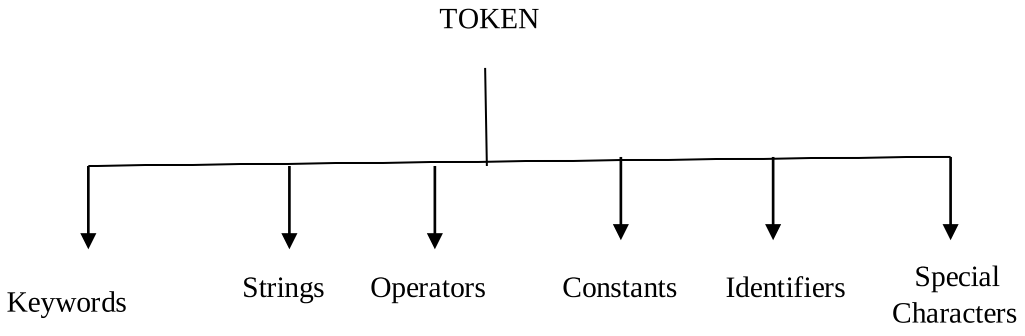
Limitation: C can't be used for internet programming like Java, .Net, PHP, etc.

Features of C:

- Simple
- Machine Independent or Portable
- Mid-level programming language
- structured programming language
- Rich Library

- Memory Management
- Fast Speed
- Pointers
- Recursion
- Extensible
- Robust
- Highly portable

TOKEN is the smallest unit in a 'C' program.

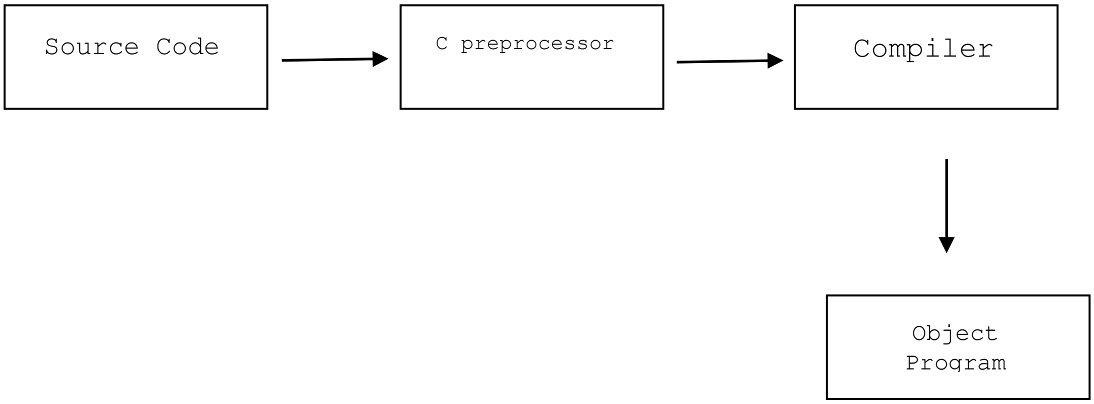


A Simple C program basically comprises of the following parts:

- Preprocessor Commands
- Functions
- Variables
- Statements & Expressions
- Comments

```
/* My First C Program */ ───▶ Comment

#include<stdio.h> ───▶ preprocessor command
```

```
int main() → main function. Execution of C program begins from main()

{ → Beginning of the main function

printf("Hello,world!"); → output statement

return 0; → terminates the execution of the main function

} → End of the main function
```



Stages in C Program compilation

Process of C Program Execution:

Every 'C' program follows a basic structure. A C program:

```
#include<stdio.h> //Pre-processor directive
int main() //main function declaration
{
printf("Hello,world!"); // output the string on a display screen
return 0; //terminating function

}
```

is written using **Text Editor** , such as [**Notepad** (in case of Windows Operating System), *vim* or vi (in case of Linux Operating System)] and saved with **[.c] Extension** – *for example, hello.c*. File Saved with [.c] extension is called **Source Program** or Source Code. C Source code with **[.c] Extension** is sent to preprocessor first.

The preprocessor generates an expanded source code:

```
The contents of <stdio.h> would be pasted at the location of #include<stdio.h>
int main()
{
printf("Hello,world!");
return 0;
}
```

Expanded source code is given as input to compiler where the expanded source program is compiled (i.e., the program is entirely read and translated to instructions the computer can understand i.e., machine understandable / readable language i.e., to **machine code sequence** of 0s and 1s). If the C compiler finds any error during compilation, it provides information about the error to the programmer.

The programmer has to review code and re-edit the program. After re-editing program, **Compiler** again check for any error. If program is error-free then it is sent to assembler [where the code is assembled and converted into object code. Now a simple**.obj file** is generated].

```
#include<stdio.h>
int main()
{
printf("Hello,world!");
return 0;
}
```

```
0100000000000000000
0111111111111111111
0101010101010101010
0000000111111111111
0000000000000001010
1111000000000000000
```

C compilation process converts the source code taken as input into the object code or machine code.

The object code is sent to linker (where the object code is linked with appropriate libraries). Then it is converted into a single executable code. A simple**.exe file** is generated.

The executable code is sent to loader (where the executable code is loaded into memory and then it is executed). After execution, output:

```
Hello,world!
```

is displayed on the console screen.

`#include<stdio.h>` → preprocessor statement

This statement begins with # symbol and is also called preprocessor directive. This statement directs the C preprocessor to include header file [`stdio.h`] and also symbolic constants [`standard input output functions`] into a C program.

- **stdio** means standard input output and **stdio.h** means standard input output header file (**printf()** and **scanf()** are not part of the **C**

language, because there is no input or output defined in C language itself - **stdio.h** comprises standard input output functions like scanf, printf etc.

- and allows standard input /output operations - note: scanf is an input function and printf is an output function (note: Letter f denote formatted) and it is included into the C program by writing the statement **#include <stdio.h>**).
- If a program is written without the statement:

  ```
  #include<stdio.h>
  ```

  then the C compiler can't compile and a compilation error is displayed on the screen (because C compiler fails to recognize the functions such as **printf()** and **scanf()**).

We can also write **#include "stdio.h"** [user define library header file] instead of **#include <stdio.h>** [predefine library header file] but **#include "stdio.h"** is added by programmer and **#include <stdio.h>** is already exist in compiler. So the statement **#include <stdio.h>** is generally preferred and the statement #include "stdio.h" is generally ignored.

main()→ main function

As the name itself indicates this is the main function of every C Program. Parentheses () indicate a function and the word **main** indicate the name of the function. main() **implies:** main function. Execution of C program begins from main(). No C program is executed without main(). It is case sensitive language: only lower case letters (or small letters) must be used and should not be enclosed by a semicolon. There must be one and only one main() function in every C program.

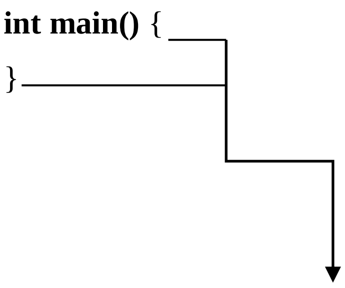
We can represent the main function in various forms, such as:

- main()
- int main()

- `void main()`
- `main(void)`
- `void main(void)`
- `int main(void)`

As we know **C is Platform dependent language**. So the Operating system needs to know when the program execution ends. So when there is value returns from the main function, the Operating System get to know that the program execution is over. **`int main()` implies:** `main()` should return integer value.

- If the main function **returns 0 to the operating system**, then the program has completed execution successfully.
- If the main function **returns 1 to the operating system**, then the program has not completed execution successfully.



int main() {

}

body of the main function within which the sequence of instructions to the computer to perform specific operations in the form of statements [input-output statements, arithmetic statements, control statements and other statements] are **written and executed**.

The left curly brace

```
{
```

implies: the beginning of the main function and the right curly brace

```
}
```

implies: the end of the main function

These braces can also be used to indicate the beginning and end of user-defined functions and compound statements.

`return 0;` → implies the exit status of execution of the program i.e., at this point, main function returns back the control of the computer to the operating system since the **execution is terminated** at this point and once a **return statement** i.e., **`return 0;`** is executed, no further instructions within the main function are executed.

For example:

```
#include<stdio.h>
int main()
{
printf("Hello,world!");
return 0;
printf("Hello,world!");
}
```

Output on the screen:

```
Hello,world!
```

`;` → implies **semicolon** or **statement terminator** [*a delimiter of the declaration*] → In C, each statement must end with a semicolon. A program is a well-defined set of instructions and each well-defined instruction (in the form of a statement) is ended by a semicolon (which is **C language punctuation** − like a period in English i.e., in an English paragraph each sentence is ended by a full stop which tells that one sentence ends and another begins, semicolon implies the end of one logical entity − that one instruction (or statement) ends and another begins).

`/* My First C Program */` → Comment

A good programmer who writes codes understood by a human is better than a programmer who generates codes understood only by the machine. So, it is highly recommended to insert comments. Comment is explanatory note on some instruction. The statement to be commented on must be enclosed within `/*` and `*/`. Comment statement is not compiled and executed.

`printf()`→ `the standard way of displaying output on the screen and` output function of the C language which makes provision to print the output:

```
Hello,world!
```

on the screen. **Parentheses ()** indicate a function and the word *printf* indicate the name of the function.

The text

`Hello,world!`

should be enclosed by the **double quotation marks** **`("     ")`** and should be written within the **printf** function and this printf function should be ended with the **semicolon** i.e.,

```
printf("Hello,world!");
```

Some of the standard input-output functions in C are given below:

scanf()	gets()
printf()	puts()
getchar()	getch()
putchar()	getche()

Unformatted Input statements

- **getchar() function**

```
main()
{
char letter;
letter =getchar();
:
}
```

- **gets() function**

```
main()
{
char name[26];
printf("Enter your name");
gets(name);
:
}
```

Unformatted Output statements

- **putchar() function**

```
main()
```

```
{
char ch;
putchar(ch);
:
}
```

```
#include<stdio.h>

int main()

{

char letter;

letter = getchar();

putchar(letter);

return 0;

}
```

- **puts() function**

```
main()
{
char name[20];
puts(name);
}
```

```
#include<stdio.h>

int main()

{

char name[20];

printf("Enter your name");

gets(name);

puts(name);
return 0;

}
```

Computer Language	Purpose
FORTRAN	Scientific and Engineering
ALGOL	Scientific and Engineering
COBAL	Business
BASIC	Business, Scientific and Engineering
PI/1, PASCAL	General purpose

	Example	Number of bytes
Single Character	A, a, X, 5	1
Single Digit	1,3,8,0	1
Single Symbol	?, *, #,$	1
Whole Number	22,-102,3610	2
Fractional number	0.1,1.25,0.56729	4
Instruction	Add, Multiply	(1 - 3)
Big Whole / Fractional number	922112580, 12550.126	(4 - 10)

Decimal Number	Binary Number	Hexadecimal Number
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B

12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14
21	10101	15
22	10110	16
23	10111	17
24	11000	18
25	11001	19
26	11010	1A
27	11011	1B
28	11100	1C
29	11101	1D
30	11110	1E
31	11111	1F
32	100000	20
64	1000000	40
128	10000000	80
256	100000000	100

- **Program 1.1**

  C program to print the word "**hello Bill Gates**" on screen

```
#include<stdio.h>
int main()
{
printf("hello Bill Gates");
return 0;
}
```

The output on the screen:

```
hello Bill Gates
```

The first line tells the compiler to include standard input output header file **`#include <stdio.h>`** to perform reading and printing of data. The second line is the main function of a C program. The body of C program contains only two statements i.e.

```
printf("hello Bill Gates");
```

When this statement is taken for execution, `main()` calls the `printf()` and `printf()is` included in **`<stdio.h>`** . Then `printf()` prints `hello Bill Gates` on the computer screen.

```
return 0;
```

In C program it is optional to include `"return 0;"` at the end of the main function and the compiler includes it automatically.

Macro substitution: Process of replacing an identifier of a C program by a symbolic constant. This can be achieved by the directive `#define`.

```
#include<stdio.h>
#define PI 3.14
int main()
{
int r, area;
r = 2;
area = PI * r * r;
printf("The area of the circle = %d", area);
return 0;
}
```

```
 #include<stdio.h>
#define CONDITION if(x>y)
```

```
#define PRINT printf("x is greater than y");

int main()
{

int x, y;
x=16;
y=2;

CONDITION

PRINT

return 0;
}
```

- Blank, tab, and newline are collectively called whitespaces.
- *C is case sensitive* that means a variable named 'age' and 'AGE' are different.

- **Program 1.2**

  C program to print

```
  *

*****

*****

*****

*****
```

on screen

```
#include<stdio.h>
int main()
{
```

```
printf("\n    *      ");
printf("\n  *****    ");
printf("\n  *****    ");
printf("\n  *****    ");
printf("\n  *****    ");
return 0;
}
```

The output on the screen:

```
  *
*****
*****
*****
*****
```

If new line sequence **`(\n)`** is not included in the above program then the output on the screen is:

```
*********************
```

Write a program to print the following outputs:

(a)

```
   *
  ****
 *******
  ****
   *
```

(b)

```
****************

* *

* Hello World! *

* *

****************
```

(c)

```
Braces come in pairs!

Comments come in pairs!

All statements end with a semicolon!

Spaces are optional!

Must have a main function!

C is done mostly in lowercase. It's a case-sensitive language
```

Answers:

(a)

```
#include<stdio.h>
int main()
{
printf("\n           *      ");
printf("\n         ****     ");
printf("\n        *******   ");
printf("\n         ****     ");
printf("\n          *      ");
return 0;
}
```

(b)

```
#include<stdio.h>
int main()
{
printf("\n           ****************         ");
printf("\n                * *                 ");
printf("\n           * Hello World! *         ");
printf("\n                * *                 ");
printf("\n           ****************         ");
return 0;
}
```

(c)

```
#include<stdio.h>
int main()
{
printf("\n Braces come in pairs!");
printf("\n Comments come in pairs!");
printf("\n  All statements end with a semicolon!");
printf("\n  Spaces are optional!");
printf("\n Must have a main function!");
printf("\n C is done mostly in lowercase. It's a case-sensitive language");
return 0;
}
```

Variables → identifiers

The variables represent a particular memory location where the data can be stored. They are used to denote arrays, functions, name of files, constants etc.

Example: sum, area, length, width, radius, age, country etc.

Rules for writing variable names

- Keywords should not be used.

- Special characters should not be used as variables
- Representing the variable names in lowercase is a **virtue programming practice**.
- There is no limit on the number of characters in a variable name.

Valid variables	**Invalid variables**
`ab`	`8ab`
`total_mark`	`total mark`
`gross_weight_2020`	`gross-weight-2020`
`area_of_sphere`	`area_ _of_ _sphere`

There are two methods of assigning values to variables

Method 1:

```
int x=1;
```

Method 2:

```
int x;

x=1;
```

- **Program 1.3**

  C program to find the area of a circle

```
#include<stdio.h>
int main()
{
int r, area;
r = 2;
area = 3.14 * r * r;
printf("The area of the circle = %d", area);
return 0;
}
```

The output on the screen:

```
The area of the circle = 12
```

int → Keyword used to indicate an integer number. Any integer is a sequence of digits without a decimal point.

Valid	Invalid
-248	3,266
14028	-34.0
+1988	+5,456.3
0	999999999999
56780	

int means the *data type* [an attribute that tells what kind of data that value can own] is *integer* and the storage size of int data type is usually **2 bytes** (*consumes a total of 16 bits in memory*). And it can take 232 distinct states from -2147483648 to 2147483647. If your computer is an 8-bit device, then the range of int is given by:

$$-2^{8-1} <= \text{integer number} <= +2^{8-1} - 1$$

$$-128 <= \text{integer number} <= +127$$

- **Note:**

  - A string, for example, is a data type that is used to categorize text and an integer is a data type used to categorize **whole numbers** or **non fractional** values [0,-5, 10].
  - We can't store **decimal values** using int data type.
  - If we use int data type to store decimal values, decimal values will be shortened and we will get only whole number. In this case, **float data type** can be used to store decimal values in a variable.

The statement

```
int r, area;
```

imply that we are declaring two ***integer variables*** r , area and these variables represent some memory location where the data is stored. Each variable is associated with a value. The process of giving values to variables is called assignment of values. The assignment operator or *storage operator* (" = ") is used to assign a value to a variable.

The statements

```
r = 2;
area = 3.14 * r * r;
```

imply that we are assigning the values to the *declared variables* where: r → represents the memory location where the ' value ' [2] should be stored and area → represents the memory location where the ' value ' [`3.14 * r * r = 3.14 * 2 * 2 = 12`] should be stored.

Comma in the statement

```
int r, area;
```

imply **variable separator** [used to separate the character groups. No space should be used after the last character group. For example, `int r, area   ;`]

The statement

```
printf("The area of the circle = %d", area);
```

make provision to print the output:

```
The area of the circle = 12
```

on the screen.

In the statement

```
printf("The area of the circle = %d", area);
```

format string or **format specifier** %d indicates that the integer value to be displayed after the statement

```
The area of the circle =
```

enclosed by *double quotes* needs to be taken from the ***variable*** `area`.

The area of the circle is 12. 56 (for r = 2) but **`The area of the circle = 12`** is displayed on the screen because **data type** int is used instead of **float** and subsequences beginning with % [format specifier] **%d** is used instead of **%f**.

If **`float r, area;`** is written instead of **`int r, area;`** and

If the statement

```
printf("The area of the circle = %f", area);
```

is written instead of

```
printf("The area of the circle = %d", area);
```

i.e.,

```
#include<stdio.h>
int main()
```

```
{
float r, area;
r = 2;
area = 3.14 * r * r;
printf("The area of the circle = %f", area);
return 0;
}
```

Then the output on the screen:

```
The area of the circle = 12.56
```

- **float** means the **data type is float**. This keyword is used to indicate a floating-point number. They are termed floating-point because of the shifting of the decimal point either to the left or the right of some digits during manipulation.

mantissa-exponent notation:

$$0.000000000013 \rightarrow 1.3 \times 10^{-11} \rightarrow 1.3\ \mathrm{E}{-11}$$

Valid	**Invalid**
-263.269	0,0
2.3698 E+03	-3.3.3
0.0268963 E+06	-9999999999.9999999 (out of range)
3689.6 E-04	-+44.44

The statement

```
float r, area;
```

imply that we are declaring the ***floating-point variables*** r, area.

- **Floating point variable** means fractional variable or decimal number (for example: 1.5, 2.5, 3.5, 4.7 … etc.) whereas integer means **non-fractional variable** or **whole number** (for example: 1, 2, 3, 4 … etc.)
- **Data type float** is used instead of int [and format string %f is used instead of %d] because if the **data type int** is used instead of float then the result will not be clearly outputted i.e., instead of 12.56 the computer displays only 12.

If the statement

```
printf("The area of the circle = %2f", area);
```

is written instead of the statement

```
printf("The area of the circle = %f", area);
```

Then the **output on the screen** is:

```
The area of the circle =             12.56
```

i.e., the statement

```
printf("The area of the circle = %f", area);
```

yields the output:

```
The area of the circle = 12.56
```

whereas the statement

```
printf("The area of the circle = %2f", area);
```

yields the output:

```
The area of the circle =             12.56
```

If you want to supply the *value for* r through the key board, then the statement

```
r =2;
```

should be replaced by the statements

```
printf("Enter any number:");
scanf("%d", &r);
```

i.e., the program should be rewritten as:

```
// a simple example of c language that gets input from the user and prints the area of
the circle.

#include<stdio.h>
int main()
{
float r, area;
printf("Enter any number:");
scanf("%d", &r);
area = 3.14 * r * r;
printf("The area of the circle = %f", area);
return 0;
}
```

The output on the screen:

```
Enter any number:

If you enter the number 2

The area of the circle = 12.56

will be outputted on the screen.
```

We use print() to put up an output message, and scanf() to read the user input into ***variable*** r.

The statement

```
printf("Enter any number:");
```

make provision to print the text

```
Enter any number:
```

on the screen.

scanf() → formatted input function.

To read the values for the variables in a C program from the keyboard, C provides a function called **scanf().** It is **equivalent to the** READ statement in FORTRAN or PASCAL.

scanf() → accepts numeric, character and string type of data. It is included in stdio.h.

& *symbol* imply the address and **&r** *imply the* address of memory location where the value of input variable [r] should be stored. **%d** specification is used to input integer data through the keyboard.

The statement

```
scanf("%d", &r);
```

make provision to **enter a** *number* for **r** through the keyboard and tells the input function **`scanf`** to read the number entered through the keyboard for r (*which is a integer*) and store it in the address of r in the computer memory (i.e., store the number in `&r`).

Note:

As told earlier: when you **enter an *integer for*** r through the keyboard, this integer will be stored in the computer memory. If you want to know the storage size of the integer in computer memory (i.e., space occupied by the entered integer in the computer memory), you need to appeal to the following program:

```
#include <stdio.h>
int main()
{
int r;
r=10;
printf("size of r = %d", sizeof(r));
return 0;
}
```

The output on the screen:

```
size of r = 4
```

i.e., integer entered for r i.e., 10 has occupied **a space of 4 bytes in the computer memory**.

Field width:

```
scanf("%3d", &a);
```

the digit 3 indicates field width of the input number.

If input number = 3485 [4 digits]

Only 3 digits [348] are stored in the memory location of an input number. If the input number has more digits than the specified field width then all extra digits are not stored.

```
scanf("%[^\n]", message);
```

go on typing any ASCII printable characters till the new line [\n] character is pressed.

Unary minus

```
a = 5, b = 6
c = a + (-b) = 5 + (-6) = -1
```

c= −1 because b was initially a positive integer variable when operated by a unary minus gets its value changed to negative.

Void data type

A void data type doesn't contain or return any value. It is mostly used for defining functions in 'C'.

Example:

```
void displayData()
```

- Decimal constant contains digits from 0-9.

```
Example: 111, 1234
```

- Octal constant contains digits from 0-7, and these types of constants are always preceded by 0.

```
Example: 012, 065
```

- Hexadecimal constant contains a digit from 0-9 as well as characters from A-F. Hexadecimal constants are always preceded by 0X.

```
Example: 0X2, 0Xbcd
```

- **Write a program to print the circumference of the circle (given r = 2.5)**

Answer:

```
#include<stdio.h>
int main()
{
float r, area;
r = 2.5;
circumference = 2* 3.14 * r;
printf("The circumference of the circle = %f", circumference);
return 0;
}
```

- **Write a program to print the area of the rectangle (given l = 2.5 and b = 3)**

Answer:

```
#include<stdio.h>
#include<stdio.h>
int main()
{
float l, b, area;
l = 2.5;
b = 3;
area = l*b;
printf("The area of the rectangle = %f", area);
return 0;
}
```

- **Format Specifiers in C**

Data Type	Specifier	Example
int	%d	14, -15, 780, +098
float	%f or %e	12.03, +5.65, 32.56 E-03
char	%c	'A', 'b', '\n', 'E'
double	%lf or %le	3.141569
long int	%ld	-218000

- **Program 1.3**

  C program to find the sum of two numbers

```
#include<stdio.h>
int main()
{
int a, b, sum;
a=1;
b=2;
sum = a + b;
printf("the sum of a and b = %d", sum);
return 0;
}
```

The output on the screen:

```
the sum of a and b = 3
```

If you want to assign the **floating -point numbers** i.e., fractional numbers for a and b (i.e., 1.5 for a and 2.6 for b) through the keyboard, then the statement

```
int a, b, sum;
```

should be replaced by the statement

```
float a, b, sum;
```

and the statement

```
printf("the sum of a and b = %d", sum);
```

should be replaced by the statement

```
printf("the sum of a and b = %f", sum);
```

i.e.,

```
#include<stdio.h>
int main()
{
float a, b, sum;
a=1.5;
b=2.6;
sum = a + b;
printf("the sum of a and b = %f", sum);
return 0;
}
```

The output on the screen:

```
the sum of a and b = 4.1
```

The statement

```
printf("the sum of a and b = %f", sum);
```

make provision to print the output:

```
the sum of a and b = 4.1
```

Even if the statement

```
printf("the sum of a and b = %f", sum);
```

is omitted from the C program, the program will be successfully executed but there will be **no display of the output** on the screen.

If you want to supply the values for a and b through the key board, then the statements

```
a=1.5;
b=2.6;
```

should be replaced by the statements

```
printf("Enter any two numbers:");
scanf("%f %f", &a, &b);
```

i.e., the **program** should be rewritten as:

```
#include<stdio.h>
int main()
{
float a, b, sum;
printf("Enter any two numbers:");
scanf("%f %f", &a, &b);
sum = a+ b;
printf("the sum of a and b = %f", sum);
return 0;
}
```

The output on the screen:

```
Enter any two numbers:

If you enter two numbers 2.9 and 3.6
```

```
the sum of a and b = 6.5

will be outputted on the screen with trailing zero's to fill up memory.
```

As Said Earlier:

- **ampersand ("&")** imply the *address* and **[&a** and **&b]** imply the addresses of the declared floating-point variables a and b stored in the **computer memory** i.e., when we enter a number for a and b through the keyboard, these numbers are read by scanf() function and they are stored in the computer memory (i.e., the number entered for a is stored in the address of a (i.e., stored in &a) and the number entered for b is stored in the address of b (i.e., stored in &b)).

There are **2 format strings** in the statement

```
scanf("%f %f", &a, &b);
```

- **one format string** %f corresponds to `&a` i.e., %f tells the **scanf() function** to read the number entered through the keyboard for a and store it in the **address of a** in the computer memory.
- and the **other format string** %f corresponds to `&b` i.e., %f tells the **scanf() function** to read the number entered through the keyboard for b and store it in the **address of b** in the computer memory.

If the two format strings are separated by a comma i.e.,

```
scanf("%f, %f", &a, &b);
```

Then the **compilation error** will be displayed on the screen.

- **Note:**

The statement

```
scanf("%f %f", &a, &b);
```

read the two numbers 2.9 and 3.6 entered through the **keyboard** and store them in the **computer memory** [i.e., the number 2.9 is stored in the address of a (i.e., stored in &a) and the number 3.6 is stored in the address of b (i.e., stored in &b)].

The statements

```
printf("Enter any two numbers:");
scanf("%f %f", &a, &b);
```

can also be replaced by the statements:

```
printf("Enter any number:");
scanf("%f", &a);
printf("Enter any number:");
scanf("%f", &b);
```

i.e.,

```
#include<stdio.h>
int main()
{
float a, b, sum;
printf("Enter any number:");
scanf("%f", &a);
printf("Enter any number:");
scanf("%f", &b);
sum = a+ b;
printf("the sum of a and b = %f", sum);
return 0;
}
```

Then the **output on the screen**:

```
Enter any number:

If you enter a number 2.9
Enter any number:
If you enter a number 3.6

the sum of a and b = 6.5 will be outputted on the screen.
```

If the statement

```
printf("the sum of a and b = %f", sum);
```

is replaced by the statement

```
printf("the sum of %f and %f = %f", a, b, sum);
```

Then the **output on the screen** is:

```
the sum of 2.9 and 3.6 = 6.5
```

In the statement

```
printf("the sum of %f and %f = %f", a, b, sum);
```

There are three format strings:

- The format string %f after the **statement** (the sum of) indicates that the value to be displayed needs to be taken from the **variable a**.
- The format string %f after the **statement** (the sum of %f and) indicates that the value to be displayed needs to be taken from the **variable b.**
- The format string %f after the **statement** (the sum of %f and %f =) indicates that the value to be displayed needs to be taken from the **variable sum**.

- **Program 1.4**

  C program to convert the temperature in Celsius to Fahrenheit

```
#include<stdio.h>
int main()
{
float C, F;
C=38.5;
F = 9*C/5 +32;
printf("temperature in Fahrenheit= %f", F);
// %f is used because the data type for F is float
return 0;
}
```

The output on the screen:

```
temperature in Fahrenheit= 101.3
```

- **Note:**

If × is used instead of * and F = $\frac{9C}{5}$ +32 is used of F = 9*C/5 +32, then the **compilation error** will be displayed on the screen.

If you want to supply the number **16 digits after decimal point** for C through the key board, then the above program should take the form:

```
#include<stdio.h>
int main()
{
double C, F;
printf("Enter any number:");
scanf("%lf", &C);
F = 9*C/5 +32;
printf("temperature in Fahrenheit= %lf", F);
// %lf is used because the data type for F is double
return 0;
}
```

- **Note:**

double → keyword used to denote a double precision floating-point number.

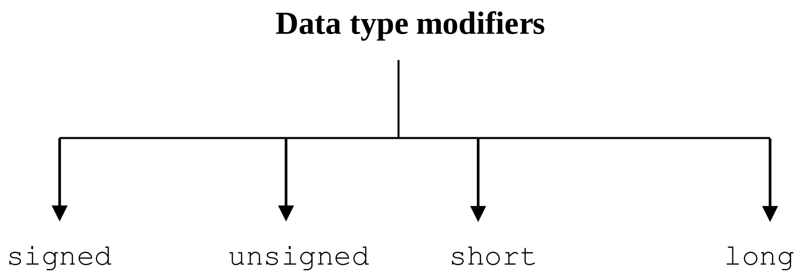
float usually stores a maximum of 6 digits after the decimal point. But double stores 16 significant digits after the decimal point.

For example:

$$234.0000000000000000$$

$$-\ 0.0000000099531510$$

The data type double and long double are used to store real numbers with precision up to 14 and 80 bits respectively.



Type modifier	Size in bytes
`int`	2
`signed int`	2
`unsigned int`	2
`short int`	2
`long int`	4
`float`	4
`double`	8
`char`	1
`signed char`	1
`unsigned char`	1
`unsigned short int`	2
`unsigned long int`	4
`long double`	10

Multiple assignment statement:

```
int x = y = 10;

float x = y = z = 6.76;
```

- **Write a program to print the sum of three numbers**

Answer:

```
#include<stdio.h>
int main()
{
int a, b, c, sum;
printf("Enter any three numbers:");
scanf("%d %d %d", &a, &b, &c);
sum = a + b + c;
printf("the sum of a, b and c = %d", sum);
return 0;
}
```

- **Write a program to print the Equivalent hexadecimal value of an integer**

Answer:

```
#include<stdio.h>
int main()
{
int a = 45;
printf("%x", a);
return 0;
}
```

Output on the screen:

```
2d
```

- **Write a program to print the area of a triangle, given**

area = $\sqrt{(s(s-a)(s-b)(s-c))}$ where s = $\frac{(a+b+c)}{2}$

Answer:

```
#include<stdio.h>
#include<math.h>
int main()
{
int a, b, c, s, area;
a = 3;
b= 4;
c=5;
s = (a + b + c) / 2;
area = sqrt ((s * (s-a) * (s-b) * (s-c));
printf("the area of the triangle = %d", area);
return 0;
}
```

- **Note:** since sqrt() is not part of C language or of standard input output file i.e., (**stdio.h file**), it is part of math file i.e., (**math.h file** which defines various mathematical functions) the statement

```
#include<math.h>
```

should be included in the C program otherwise the compilation error will be flagged on the screen stating that sqrt() is not declared or defined.

- **Note:** If the statement

```
area = (s (s-a) (s-b) (s-c))^1/2
```

is written instead of

```
area = sqrt ((s * (s-a) * (s-b) * (s-c));
```

Then the **compilation error** will be displayed on the screen because C does not support

```
area = (s (s-a) (s-b) (s-c))^1/2
```

- **Stuff you need to know about:**

1 kilobyte = 2^{10} = 1024 bytes

1 megabyte = $2^{10} \times 2^{10}$ = 1024 × 1024 bytes

1 gigabyte = $2^{10} \times 2^{10} \times 2^{10}$ = 1024 × 1024 × 1024 bytes

1 terabyte = $2^{10} \times 2^{10} \times 2^{10} \times 2^{10}$ = 1024 × 1024 × 1024 × 1024 bytes

- **Program 1.5**

C program to find the product of two numbers

```
#include<stdio.h>
int main()
{
int a, b, product;
a=1;
b=2;
product = a * b;
printf("the product of a and b = %d", product);
return 0;
}
```

The output on the screen:

```
the product of a and b = 2
```

If you want to insert a 10 digit number for a and b i.e.,

```
a=1000000000
b=3000000000
```

, then the statement:

```
int a, b, product;
```

should be replaced by the statement

```
long int a, b, product;
```

and %ld **should be used instead of** %d

i.e., the program should be rewritten as:

```
#include<stdio.h>
int main()
{
long int a, b, product;
a=1000000000;
b=2000000000;
product = a * b;
printf("the product of a and b = %ld", product);
return 0;
}
```

The output on the screen:

```
the product of a and b = 3000000000000000000
```

If you want to supply the values for a and b **through the key board**, then the statements

```
a=1;
b=2;
```

should be replaced by the statements

```
printf("Enter any two numbers:");
scanf("%d %d", &a, &b);
```

i.e.,

```
#include<stdio.h>
int main()
{
int a, b, product;
printf("Enter any two numbers:");
scanf("%d %d", &a, &b);
product = a* b;
printf("the product of a and b = %d", product);
return 0;
}
```

The output on the screen:

```
Enter any two numbers:
```

```
If you enter two numbers 1 and 3
the product of a and b = 3 will be outputted on the screen.
```

If you replace the statements

```
printf("Enter any two numbers:");
scanf("%d %d", &a, &b);
```

by the statements

```
printf("Enter any number:");
scanf("%d", &a);
printf("Enter any number:");
scanf("%d", &b);
```

Then the output on the screen will be:

```
Enter any number:
If you enter the number 3
Enter any number:
If you enter the number 3
the product of a and b = 9
will be outputted on the screen.
```

If the statement

```
printf("the product of a and b = %d"; product);
```

is written instead of the statement

```
printf("the product of a and b = %d", product);
```

i.e., instead of **variable separator** (i.e., comma) semicolon is used − Then the **compilation error** will be displayed on the screen.

- **Note:**

```
#include <stdio.h>
int main()
{
printf("Hello, World!");
printf("Hello, World!\b");
printf("Hello, World!\b");
printf("Hello, World!\b");
return 0;
}
```

i.e., if **back space** \b is used then

```
Hello, World!Hello, World!Hello, World!Hello, World!
```

will be outputted on the screen.

If **carriage return** \r is used instead of \b

i.e.,

```
#include <stdio.h>
int main()
{
printf("Hello, World!");
printf("Hello, World!\r");
printf("Hello, World!\r");
printf("Hello, World!\r");
return 0;
}
```

The output on the screen is:

```
Hello, World!Hello, World!
Hello, World!
Hello, World!
```

If **Horizontal tab** \t is used instead of \r

i.e.,

```
#include <stdio.h>
int main()
{
printf("Hello, World!\t");
printf("Hello, World!\t");
printf("Hello, World!\t");
printf("Hello, World!\t");
return 0;
}
```

The output on the screen is:

```
Hello, World!       Hello, World!     Hello, World!     Hello, World!
```

If **vertical tab** \v is used instead of \t

i.e.,

```
#include <stdio.h>
int main()
{
printf("Hello, World!\v");
printf("Hello, World!\v");
printf("Hello, World!\v");
printf("Hello, World!\v");
return 0;
}
```

The output on the screen is:

```
Hello, World!
Hello, World!
Hello, World!
Hello, World!
```

- **Program 1.5**

  C program to find the square of a number

```
#include<stdio.h>
int main()
{
int a, b;
a=2;
b = a * a;
printf("the square of a = %d", b);
}
```

The output on the screen:

```
the square of a = 4
```

If the statement

```
b = a * a;
```

is replaced by

```
b = pow((a), 2);
```

i.e., if the **above program** is rewritten as:

```
#include<stdio.h>
#include<math.h>
int main()
{
int a, b;
a=2;
b = pow((a), 2);
printf("the square of a = %d", b);
return 0;
}
```

Then there will be no display of **compilation error** on the screen or there will be no change in the output on the screen i.e.,

```
the square of a = 4
```

will be outputted on the screen.

which means:

```
b = pow((a), 2); is the same as b = a*a;
```

Since **`b = pow((a), 2);`** is used instead of **`b = a*a;`**

`#include<math.h>` should be included in the C program as ***b = pow((a), 2);*** is supported by `#include<math.h>`

Otherwise **compilation Error** will be displayed on the screen.

If you want to supply the integer value for a through the key board, then the statement

```
a=2;
```

is replaced by the statements

```
printf("Enter any number:");
scanf("%d", &a);
```

i.e.,

```
#include<stdio.h>
int main()
{
int a, b;
printf("Enter any number:");
scanf("%d", &a);
b = a * a;
printf("the square of a = %d", b);
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter a number 4
the square of a = 16 will be outputted on the screen.
```

- **Note:**
  - If `scanf(%d, &a);` is written instead of **`scanf("%d", &a);`**
  - If `printf(the square of a = %d, b);` is written instead of `printf("the square of a = %d", b);`
  - If the main function is followed by a semicolon i.e.,

    int main(); is written instead of int main()

  Then the compilation error will be displayed on the screen.

But if the **body of the main function** is followed by a semicolon i.e.,

```
int main()
{
};
```

is written instead of

```
int main()
{
}
```

There will be no display of the compilation error on the screen.

```
int main(); ⟶ ERROR


int main()
{
}; ⟶ NO ERROR
```

- **Write a program to print the cube of a number**

  Answer:

```
#include<stdio.h>
#include<math.h>
int main()
{
int a, b;
a=2;
b = pow((a), 3);
printf("the cube of a = %d", b);
return 0;
}
```

- **Write a program to print the energy of the substance using energy = mc^2**

  Answer:

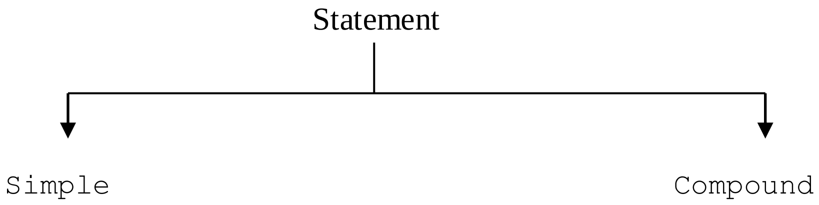
```
#include<stdio.h>
#include<math.h>
int main()
{
int m;
long int c, energy;
m=2;
c = 300000000;
energy = m * pow((c), 2);
printf("the energy of the substance  = %ld joules", energy);
return 0;
}
```

The syntax of *if* statement is:

```
if (this condition is true)
{
print this statement;
}
```

where condition → a logical expression that results in true or false.

```
#include<stdio.h>
int main()
{
int a;
printf("Enter any number \n");
scanf("%d", &a);
if((a%2)!=0){
printf("%d is an odd number\n", a);
}
return 0;
}
```



- **Simple statement** → a single statement
- **Compound statement** → a collection of two or more statements placed between the braces.

- **Program 1.6**

  C program to find the greatest of two numbers using if - else statement

- **The syntax of *if – else* statement** *(Conditional Statement)***:**

```
if (this condition is true)

{

print this statement;

}

else

{

print this statement;

}
```

If-else `statement` → termed as branching as a program decides which statement to execute based on the result of the evaluated condition.

```
#include<stdio.h>
int main()
{
int a, b;
a = 2;
b = 3;
if(a>b)
{
printf("a is greater than b");
}
else
{
printf("b is greater than a");
}
return 0;
}
```

The output on the screen:

```
b is greater than a
```

Since the **condition**

```
a>b
```

within the parentheses is not true, the statement **a** *is greater than* **b** is not executed; instead the **execution skips** and pass to print the statement b is greater than a.

In simpler words, (a>b) is the condition (i.e., **logical expression that results in true or false**) and if the condition (a>b) is true, then the statement:

```
{
printf("a is greater than b");
}
```

is executed to print the output:

a is greater than b

else the statement

```
{
printf("b is greater than a");
}
```

is executed to print the output:

```
b is greater than a
```

If you want to supply the **integer values for** a and b through the key board, then the statements

```
a=2;
b=3; should be replaced by the statements
printf("Enter any number:");
scanf("%d", &a);
printf("Enter any number:");
scanf("%d", &b);
```

i.e., the program should be rewritten as:

```
#include<stdio.h>
int main()
{
int a, b;
printf("Enter any number:");
scanf("%d", &a);
printf("Enter any number:");
scanf("%d", &b);
if(a>b)
{
printf("a is greater than b");
}
else
{
printf("b is greater than a");
}
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 6
Enter any number:
If you enter the number 3
a is greater than b
```

```
will be outputted on the screen.
```

Nested If-else Statement:

When a series of decision is required, nested if-else is used. Nesting means using one if-else construct within another one.

```
#include<stdio.h>
int main()
{
        int a =1;
        if(a<10)
        {
                if(num==1)
                {
                        printf("The value is:%d\n", a);
                }
                else
                {
                        printf("The value is greater than 1");
                }
        }
        else
        {
                printf("The value is greater than 10");
        }
        return 0;
}
```

The output on the screen:

```
The value is: 1
```

- **Program 1.7**

  C program to find the greatest of three numbers using if - else if - else statement

  - **The syntax of *if - else if - else* statement:**

```
if (this condition is true)

{

print this statement;

}

else if (this condition is true)

{

print this statement;

}

else

{

print this statement;

}
```

```c
#include<stdio.h>
int main()
{
int a, b, c;
printf("Enter any number:");
scanf("%d", &a);
printf("Enter any number:");
scanf("%d", &b);
printf("Enter any number:");
scanf("%d", &c);
if(a>b&&a>c)
{
printf("%d is greater than %d and %d", a, b, c);
}
else if (b>a&&b>c)
{
printf("%d is greater than %d and %d", b, a, c);
}
else
{
printf("%d is greater than %d and %d", c, b, a);
}
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 3
Enter any number:
If you enter the number 4
4 is greater than 3 and 2
will be outputted on the screen.
```

double ampersand "&&" imply and.

`(a>b&&a>c)`

`(b>a&&b>c)`

denote conditions.

i.e., the condition

`(a>b&&a>c)` imply:

a is greater than b and a is greater than c

and if this condition is true, then the statement

```
{
printf("a is greater than b and c");
}
```

is executed to print the output:

a is greater than b and c

and if the condition (a>b&&a>c) is not true

the statement

```
{
printf("a is greater than b and c");
}
```

is not executed; instead the execution skips and pass to the condition (b>a&&b>c)

and if this condition is true, then the statement

```
{
printf("b is greater than a and c");
}
```

is executed to print the **output**:

b is greater than a and c

and if the condition **(b>a&&b>c)** is not true, then the statement

```
{
printf("b is greater than a and c");
}
```

is not executed; instead the execution skips and the statement

```
{
```

```
printf("c is greater than b and a");
}
```

is executed to print the output:

c is greater than b and a

If the statements:

```
if(a>b&&a>c)
{
printf("%d is greater than %d and %d", a, b, c);
}
else if (b>a&&b>c)
{
printf("%d is greater than %d and %d", b, a, c);
}
else
{
printf("%d is greater than %d and %d", c, b, a);
}
```

are replaced by the **statements**:

```
if(a>b&&a>c)
printf("%d is greater than %d and %d", a, b, c);
else if (b>a&&b>c)
printf("%d is greater than %d and %d", b, a, c);
else
printf("%d is greater than %d and %d", c, b, a);
```

i.e., if the program is rewritten as:

```
#include<stdio.h>
```

```
int main()
{
int a, b, c;
printf("Enter any number:");
scanf("%d", &a);
printf("Enter any number:");
scanf("%d", &b);
printf("Enter any number:");
scanf("%d", &c);
if(a>b&&a>c)
printf("%d is greater than %d and %d", a, b, c);
else if (b>a&&b>c)
printf("%d is greater than %d and %d", b, a, c);
else
printf("%d is greater than %d and %d", c, b, a);
return 0;
}
```

There will be no display of error on the screen

c is greater than b and a

will be successfully outputted on the screen

- **What will be the output of the following program?**

```
#include <stdio.h>
int main()
{
int a, b;
a=2;
b=2;
if(a>b || a==b)
printf("a is greater than or equal to b");
else
printf("b is greater than a");
return 0;
}
```

Answer:

```
a is greater than or equal to b



Note:

symbol || denote OR i.e., a>b || a==b denote a is greater than or a is equal to b.
```

- **Program 1.8**

  C program to find the average of 10 numbers

```
#include<stdio.h>
int main()
{
int N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, X;
printf("Enter any 10 numbers:");
scanf("%d%d%d%d%d%d%d%d%d%d", &N1, &N2, &N3, &N4, &N5, &N6, &N7, &N8, &N9, &N10);
X = (N1 + N2 + N3 + N4 + N5 + N6 + N7 + N8 + N9 + N10) /10;
printf("the average of 10 numbers = %d", X);
return 0;
}
```

The output on the screen:

```
Enter any 10 numbers:
If you enter ten numbers 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10
the average of 10 numbers = 5
will be outputted on the screen.
```

- **Note:** The average of 10 numbers is 5.5, the output on the screen is 5 because the **data type int** is used instead of float.

- **Any mathematical expression should be written in C equivalent expression to prevent the display of compilation error on the screen because C language does not accept the general mathematical expressions.**

Mathematical expression	**C equivalent expression**
$x \times \frac{y}{z}$	x * y / z
$(ax + 1)\ (by + 2)$	(a * x + 1) * (b * y + 2)
$\frac{(a+b)^2}{(a-b)^2}$	(a+b) * (a+b) / (a-b) * (a-b) or pow((a+b), 2) / pow((a - b), 2)
$\log_{10} (\frac{x}{y} + c)$	log 10 (x/y + c)
ax^2+bx+c	a*x*x+b*x+c
$\ln x$	log(x)
$e^x + b$	exp (x) + b
$\sin\theta + \cos\theta$	sin (theta) + cos (theta)
$\alpha = \beta + \gamma$	alpha = beta + gamma
$\sqrt{x}$	sqrt(x)
$\sqrt[3]{x}$	cbrt(x)
$\sqrt{p^2 + q^2}$	sqrt (p*p + q*q)

$2a^2 + 3b$	2a *a + 3b + 2
a = e ***to the power of*** $\frac{x}{\sqrt{1+\sin\theta}}$	a = exp (x / sqrt (1 + sin (theta)))

- **What will be the output of the following programs:**

(a)

```
#include <stdio.h>
#include<math.h>
int main()
{
int a, b, x;
x=2;
b=2;
a = exp (x) + b;
printf("the value of a = %d", a);
return 0;
}
```

Answer:

```
the value of a = 9
```

(b)

```
#include <stdio.h>
#include<math.h>
int main()
{
int alpha, beta, gamma;
alpha =2;
beta=2;
gamma=  2 * alpha +  beta;
printf("the value of alpha = %d", alpha);
return 0;
}
```

Answer:

```
the value of alpha = 2
```

(c)

```
#include <stdio.h>
#include<math.h>
int main()
{
double theta,  result;
theta = 90;
result = sin(theta);
printf ("The sine 90 degrees is = %lf ", result);
return 0;
}
```

Answer:

```
The sine 90 degrees is = 0.893997
```

- **What is C equivalent expression of $\left[\frac{x}{y}\right]^{n-1}$?**

Answer:

```
pow((x/y), n-1)
```

- **Program 1.9**

  C program to find the square root of a number

```
#include<stdio.h>
#include<math.h>
int main()
{
int a, b;
printf("Enter any number:");
scanf("%d", & a);
b = sqrt (a);
printf("the square root of a number = %d", b);
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 4
```

```
the square root of a number = 2
is outputted on the screen.
```

Suppose if you enter the number 2, the **square root of a number = 1** is outputted on the screen because **int** is used instead of **float**.

- **Note:**

  Since b = sqrt (a) is written

  `#include<math.h>`

  must be included in the above program otherwise **compilation error** will flag on the screen.

i.e., the program:

```
#include<stdio.h>
int main()
{
int a, b;
printf("Enter any number:");
scanf("%d", & a);
b = sqrt (a);
printf("the square root of a number = %d", b);
return 0;
}
```

will flag compilation error on the screen.

If **float** is used instead of **int** then the above program take the form:

```
#include<stdio.h>
```

```
#include<math.h>
int main()
{
float a, b;
printf("Enter any number:");
scanf("%d", & a);
b = sqrt (a);
printf("the square root of a number = %f", b);
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 5
the square root of a number = 2.23
is outputted on the screen.
```

- **Note:**

```
#include<stdio.h>
#include<math.h>
int main()
{
printf("the square root of a number = %f", sqrt (4));
return 0;
}
```

```
|| imply or
> imply greater than
< imply less than
== imply equal to
! imply not
!= imply not equal to
&& imply and
```

```
& imply address
```

- **Program 2.0**

  C program to find the simple interest

```
#include<stdio.h>
int main()
{
int P,T, R, SI;
P = 1000;
T = 2;
R = 3;
SI = P*T*R/100;
printf("the simple interest = %d", SI);
return 0;
}
```

The output on the screen:

```
the simple interest = 60
```

- **Note:**

  If you write **SI = PTR/100;** instead of **`SI=P*T*R/100;`**

  Then compilation error is displayed on the screen because C language does not accept the general expressions.

If you want to supply the values for P, T and R through the key board, then the statements:

```
P = 1000;
T = 2;
```

```
R = 3;
```

should be replaced by the statements:

```
printf("Enter any number:");
scanf("%d", &P);
printf("Enter any number:");
scanf("%d", &T);
printf("Enter any number:");
scanf("%d", &R);
```

i.e., the above program should take the form:

```
#include<stdio.h>
int main()
{
int P,T, R, SI;
printf("Enter principal amount:");
scanf("%d", &P);
printf("Enter time:");
scanf("%d", &T);
printf("Enter rate of interest:");
scanf("%d", &R);
SI = P*T*R/100;
printf("the simple interest = %d", SI);
return 0;
}
```

The output on the screen:

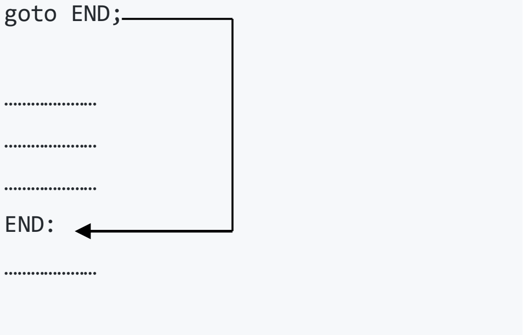
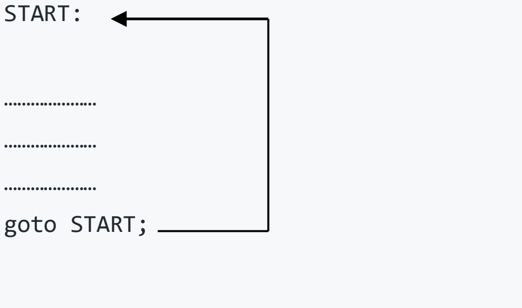
```
Enter principal amount:
```

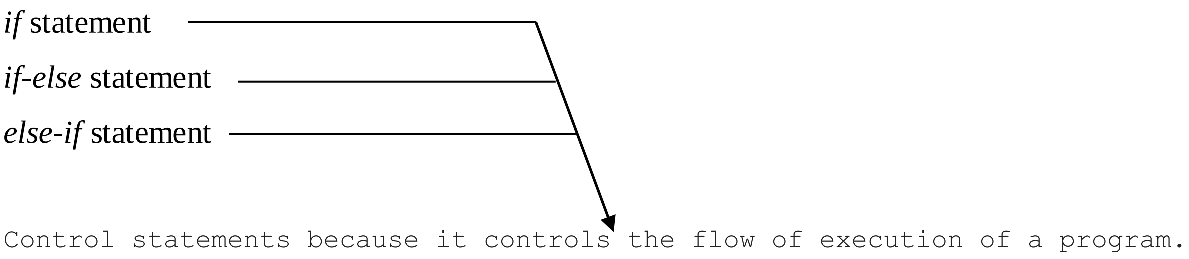
```
If you enter the principal amount 1000
Enter time:
If you enter the time 2
Enter rate of interest:
If you enter the rate of interest 3
the simple interest = 60

will be outputted on the screen.
```

goto statement: Like other languages, C supports an unconditional control statement [**goto**] to transfer the control from one point to another in a C program. The goto is a branching statement and requires a label. The syntax of **goto** statement is as follows:

goto label;

The label can be written anywhere in the C program either before or after the goto statement. For example:

 goto END; .... END: .... The label END: is written after the goto END; statement.	 START: .... goto START; The label START: is written before the goto START; statement.



if statement

if-else statement

else-if statement

`Control statements because it controls the flow of execution of a program.`

- **Program 2.1**

  C program to find whether the person is senior citizen or not

```
#include<stdio.h>
int main()
{
int age;
age=20;
if(age> = 60)
{
printf("senior citizen");
}
if(age<60)
{
printf("not a senior citizen");
}
return 0;
}
```

The output on the screen:

```
not a senior citizen
```

- (age>= 60) means age greater than or equal to 60

If you want to supply the value for age through the key board, then the statement

```
age=20;
```

should be replaced by the statements:

```
printf("Enter age:");
scanf("%d", &age);
```

i.e., the above program should take the form:

```
#include<stdio.h>
int main()
{
int age;
printf("Enter age:");
scanf("%d", &age);
if(age>60)
{
printf("senior citizen");
}
if(age<60)
{
printf("not a senior citizen");
}
return 0;
}
```

The output on the screen:

```
Enter age:
```

```
If you enter the value 60
senior citizen will be outputted on the screen.
Suppose if you enter the value 27
not a senior citizen will be outputted on the screen.
```

- **Program 2.2**

  C program to get marks for 3 subjects and declare the result.

  If the marks >= 35 in all the subjects the student passes else fails.

```
#include<stdio.h>
int main()
{
int M1, M2,M3;
M1 = 38;
M2= 45;
M3 = 67;
if(M1>= 35 && M2>= 35 && M3>= 35)
{
printf("candidate is passed");
}
else
{
printf("candidate is failed");
}
return 0;
}
```

The output on the screen:

```
candidate is passed
```

`>=` imply greater than or equal to and **double ampersand** imply `and`

`(M1>= 35 && M2>= 35 && M3>= 35)` denote the condition and this condition imply M1 is greater than or equal to 35 and M2 is greater than or equal to 35 and M3 is greater than or equal to 35. And if this condition is **TRUE**, then the statement

```
{
printf("candidate is passed");
}
is executed to print the output:
candidate is passed
else the statement
{
printf("candidate is failed");
}
```

is executed to print the output:

```
candidate is failed
```

If you want to supply the integer values for marks M1, M2 and M3 through the **key board**, then the statements:

```
M1 = 38;
M2= 45;
M3 = 67;
```

should be replaced by the statements:

```
printf("Enter any three numbers:");
scanf("%d%d%d", &M1, &M2, &M3);
```

i.e.,

```
#include<stdio.h>
int main()
{
int M1, M2,M3;
printf("Enter any three numbers:");
scanf("%d%d%d", &M1, &M2, &M3);
if(M1>= 35 && M2>= 35 && M3>= 35)
{
printf("candidate is passed");
}
else
{
printf("candidate is failed");
}
return 0;
}
```

The output on the screen:

```
Enter any three numbers:
If you enter three numbers 26, 28, 39
candidate is failed will be outputted on the screen.
```

- **Write a program to check whether a character is an alphabet or not using the function isalpha()**

```
#include <stdio.h>
#include <ctype.h>
int main()
{
```

```
int a =2;
if( isalpha(a) )
{
printf(" the character a  is an alphabet");
}
else
{
printf("the character a  is not an alphabet");
}
return 0;
}
```

The output on the screen:

```
the character a  is not an alphabet
```

```
#include <stdio.h>
#include <ctype.h>
int main()
{
char a = 'b';
if( isalpha(a) )
{
printf(" the character a  is an alphabet");
}
else
{
printf("the character a  is not an alphabet");
}
return 0;
}
```

The output on the screen:

```
the character a  is an alphabet
```

If the statement **`char a = b;`** is written instead of **char a = 'b';** Then the compilation error will be flagged on the display screen.

- **Program 2.3**

  C program to find profit or loss

```
#include<stdio.h>
int main()
{
int CP, SP, loss, profit;
printf("Enter cost price:");
scanf("%d", &CP);
printf("Enter selling price:");
scanf("%d", &SP);
if(SP>CP)
{
printf("profit=%d", SP-CP);
}
else
{
printf("loss =%d", CP-SP);
}
return 0;
}
```

The output on the screen:

```
Enter cost price:
If you enter the cost price 25
Enter selling price:
If you enter the selling price 26
profit = 1 will be outputted on the screen.
```

If the **condition** (SP>CP) is true, then the statement

```
{
printf("profit=%d", SP-CP);
}
```

is executed to print the output:

```
profit = SP-CP (in this case profit = 26-25 =1)
```

else the statement

```
{
printf("loss=%d", CP-SP);
}
```

is executed to print the output:

```
loss = CP-SP
```

- **Program 2.4**

  C program to convert inches into centimeter

```
#include<stdio.h>
int main()
{
float I, C;
I=3.5;
C = 2.54*I;
printf("length in centimeters= %f", C);
return 0;
}
```

The output on the screen:

```
length in centimeters = 8.89
```

- **Note:** float is used instead of int because I = 3.5 if int is used instead of float then the result will not be clearly outputted i.e., instead of 8.89 the **computer displays** only 8. And since float is used instead of int, the *format specifier* %d is replaced by the *format string* %f.

If you want to supply the ***floating-point variable*** for I through the **key board**, then the above program should take the form:

```
#include<stdio.h>
int main()
{
float I, C;
printf("Enter the length in inches:");
scanf("%f", &I);
C = 2.54*I;
printf("length in centimeters= %f", C);
return 0;
}
```

The output on the screen:

```
Enter the length in inches:

If you enter the floating point value or fractional or decimal number for I i.e., 25.5

length in centimeters = 64.9 will be outputted on the screen.
```

Increment operator: This operator is used to increment the value of an integer number by 1. This is represented by '++' [double plus] symbol. This symbol can be placed after the integer variable. For example, if int a = 4; then, `a++ indicate a = a+1.` Thus, the value of a is 5.

Decrement operator: This operator is used to reduce the value of an integer number by 1. This is represented by '--' [double minus] symbol. This symbol can be placed after the integer variable. For example, if int a = 4; then, `a-- indicate a = a-1.` Thus, the value of a is 3.

- **Program 2.5**

  C program to find the incremented and decremented values of two numbers.

```
#include<stdio.h>
int main()
{
int a, b, c, d, e, f;
a = 10;
b=12;
c=a+1;
d=b+1;
e=a-1;
f=b-1;
printf("the incremented value of a =%d", c);
printf("the incremented value of b =%d", d);
printf("the decremented value of a =%d", e);
printf("the decremented value of b =%d", f);
return 0;
}
```

The output on the screen:

```
the incremented value of a = 11 the incremented value of b = 13 the decremented value
of a = 9 the decremented value of b = 11
```

If the statements:

```
printf("the incremented value of a =%d", c);
printf("the incremented value of b =%d", d);
printf("the decremented value of a =%d", e);
printf("the decremented value of b =%d", f);
```

are replaced by the statements:

```
printf("the incremented value of a =%d\n", c);
printf("the incremented value of b =%d\n", d);
printf("the decremented value of a =%d\n", e);
printf("the decremented value of b =%d\n", f);
```

Then the output on the screen is:

```
the incremented value of a = 11
the incremented value of b = 13
the decremented value of a = 9
the decremented value of b = 11
```

- **Note:**

Even if the statements:

```
printf("the incremented value of a =%d\n", c);
printf("the incremented value of b =%d\n", d);
printf("the decremented value of a =%d\n", e);
printf("the decremented value of b =%d\n", f);
```

are replaced by the statements:

```
printf("\n the incremented value of a =%d", c);
printf("\n the incremented value of b =%d", d);
printf("\n the decremented value of a =%d", e);
printf("\n the decremented value of b =%d", f);
```

There will be **no change in the output on the screen**.

If you want to supply the values for a and b through **the key board**, then the above program should take the form:

```
#include<stdio.h>
int main()
{
int a, b, c, d, e, f;
printf("Enter any number:");
scanf("%d", &a);
printf("Enter any number:");
scanf("%d", &b);
c=a+1;
d=b+1;
e=a-1;
f=b-1;
printf("the incremented value of a =%d\n", c);
printf("the incremented value of b =%d\n", d);
printf("the decremented value of a =%d\n", e);
printf("the decremented value of b =%d\n", f);
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 3

the incremented value of a = 3
the incremented value of b = 4
the decremented value of a = 1
the decremented value of b = 2

will be outputted on the screen.
```

Bitwise complement

During bitwise complement operation each zero gets changed to 1 and each one gets changed to 0.

a = 10 and its equivalent binary value is 1010

b = ~a = ~ (1010) = 0101 [which is 1's complement of a]

Consider the binary number 10000100. It is equivalent to 132 in decimal.

Bitwise Left shifting

Before shifting of one bit left

1	0	0	0	0	1	0	0

After one bit shifted to left

0	0	0	0	1	0	0	0

Bitwise right shifting

Before shifting of one bit right

1	0	0	0	0	1	0	0

After one bit shifted to right

0	1	0	0	0	0	1	0

```
#include<stdio.h>
int main()
{
int a, b;
a=128;
b=32;
a=a>>1;
printf("After right-shifting by 1, a =%d\n", a);
b=b<<2;
printf("After left-shifting by 2, b =%d\n", b);
return 0;
}
```

The output on the screen:

```
After right-shifting by 1, a = 64
After left-shifting by 2, b = 128
```

- **Program 2.6**

The percentage marks are entered and the grades are allotted as follows :

percentage>= 60 First Class

percentage>=50 and per <= 60 Second Class

percentage>= 40 and per <= 50 Pass Class

percentage< 40 Fail

Write a C program for the above:

```
#include<stdio.h>
int main()
{
int P;
printf("Enter the percentage:");
scanf("%d", &P);
if(P >= 60)
{
printf("first class");
}
if(P>=50&&P <60)
{
printf("second class");
}
else if(P>=40&&P<=50 )
{
printf("pass class");
}
else
{
printf("fail");
}
return 0;
}
```

The output on the screen:

```
Enter the percentage:
If you enter the percentage 65
first class will be outputted on the screen.
```

- **Program 2.7**

  C program to calculate the discounted price and the total price after discount

  Given:

  - If purchase value is greater than 1000, 10% discount
  - If purchase value is greater than 5000, 20% discount
  - If purchase value is greater than 10000, 30% discount

- **Discounted price**

```
#include<stdio.h>
int main()
{
double PV, dis;
printf("Enter purchased value:");
scanf("%lf", &PV);
if(PV>1000)
{
printf("dis=%lf", PV* 0.1);
}
else if(PV>5000)
{
printf("dis =%lf", PV* 0.2);
}
else
{
printf("dis=%lf", PV* 0.3);
}
return 0;
}
```

The output on the screen:

```
Enter purchased value:
If you enter the purchased value 6500
dis = 1300.000000 will be outputted on the screen.
```

- **Total price**

```
#include<stdio.h>
int main()
{
double PV, total;
printf("Enter purchased value:");
scanf("%lf", &PV);
if(PV<1000)
{
printf("total=%lf", PV - PV* 0.1);
}
else if(PV<5000)
{
printf("total =%lf", PV- PV* 0.2);
}
else
{
printf("total=%lf", PV- PV* 0.3);
}
return 0;
}
```

The output on the screen:

```
Enter purchased value:
If you enter the purchased value 650
total = 585.000000  will be outputted on the screen.
```

- **Now, Combing both the programs (above), we can write:**

```
#include<stdio.h>
int main()
{
double PV, dis, total;
printf("Enter purchased value:");
scanf("%lf", &PV);
if(PV>1000)
{
printf("dis=%lf", PV* 0.1);
printf("total=%lf", PV - PV* 0.1);
}
else if(PV>5000)
{
printf("dis =%lf", PV* 0.2);
printf("total=%lf", PV - PV* 0.1);
}
else
{
```

```
printf("dis=%lf", PV* 0.3);
printf("total=%lf", PV - PV* 0.1);
}
return 0;
}
```

The output on the screen:

```
Enter purchased value:
If you enter the purchased value 850
dis = 85.000000
total = 765.000000
will be outputted on the screen.
```

Program illustrating the ternary operation

```
#include <stdio.h>

int main()
{

int x = 5, y = 10, result;

       max = (x > y) ? x : y;

      printf("Largest number between"
           " %d and %d is %d. ",
           x, y, result);

    return 0;
}
```

The output on the screen:

```
Largest number between 5 and 10 is 10.
```

What is a Loop?

A Loop executes the sequence of statements many times until the stated condition becomes false.

Types of Loops: Depending upon the position of a control statement in a program, a loop is classified into two types:

- *Entry controlled loop* → a condition is checked before executing the body of a loop. It is also called as a pre-checking loop.
- *Exit controlled loop* → a condition is checked after executing the body of a loop. It is also called as a post-checking loop.

- **Program 2.8**
  C program to print the first ten natural numbers using *for* loop statement

```
#include<stdio.h>
int main()
{
int i;
for (i=1; i<=10; i++)
printf("value of i =%d", i);
return 0;
}
```

The output on the screen is:

```
value of i = 1 value of i = 2  value of i= 3  value of i= 4  value of i= 5  value of i=
6 value of i = 7 value of i= 8  value of i = 9  value of i = 10
```

`for (i=1; i<=10; i++)` denote the ***for* loop statement** and the syntax of the

for loop statement is:

`for (initialization; condition; increment)`

Here:

```
i=1 denote initialization (i.e., from where to start)
i<=10 denote the condition (i.e., stop when 10 is reached)
i++ implies increment (which tells the value of i to increase by 1 each time the loop
is executed) and i++ is the same as i+1.
```

The number of iterations required to execute the body of ***for*** loop is computed using the formula:

$$\text{Number of iteration} = \frac{(\text{Final value} - \text{initial value} + \text{step increment})}{\text{step increment}}$$

- **When a *for* loop executes, the following occurs:**

```
i = 1
Is the condition (i<=10) is true?
Yes because i=1
The statement printf("value of i =%d", i); is executed to print the output:
value of i = 1
Now, the value of i is:
i =  1+1 = 2
Is the condition (i<=10) is true?
Yes because i=2
The statement printf("value of i =%d", i); is executed to print the output:
value of i = 2
```

```
Now, the value of i is:
i =  2+1 = 3
Is the condition (i<=10) is true?
Yes because i=3
The statement printf("value of i =%d", i); is executed to print the output:
value of i = 3
Now, the value of i is:
i =  3+1 = 4
Is the condition (i<=10) is true?
Yes because i=4
The statement printf("value of i =%d", i); is executed to print the output:
value of i = 4
Now, the value of i is:
i =  4+1 = 5
Is the condition (i<=10) is true?
Yes because i=5
The statement printf("value of i =%d", i); is executed to print the output:
value of i = 5
Now, the value of i is:
i =  5+1 = 6
Is the condition (i<=10) is true?
Yes because i=6
The statement printf("value of i =%d", i); is executed to print the output:
value of i = 6
Now, the value of i is:
i =  6+1 = 7
Is the condition (i<=10) is true?
Yes because i=7
The statement printf("value of i =%d", i); is executed to print the output:
value of i = 7
Now, the value of i is:
i =  7+1 = 8
Is the condition (i<=10) is true?
Yes because i=8
The statement printf("value of i =%d", i); is executed to print the output:
value of i = 8
Now, the value of i is:
i =  8+1 = 9
Is the condition (i<=10) is true?
Yes because i=9
The statement printf("value of i =%d", i); is executed to print the output:
```

```
value of i = 9
Now, the value of i is:
i =  9+1 = 10
Is the condition (i<=10) is true?
Yes because i=10
The statement printf("value of i =%d", i); is executed to print the output:
value of i = 10
and stop because the condition i<=10 is achieved.
```

If the statement:

```
printf("value of i =%d", i);
```

is replaced by the **statement**:

```
printf("value of i =%d\n", i);
                                  or
printf("\n value of i =%d", i);
```

Then the **output on the screen** is:

```
value of i = 1
value of i = 2
value of i = 3
value of i = 4
value of i = 5
value of i = 6
value of i = 7
value of i = 8
value of i = 9
value of i = 10
```

If the *for* loop statement:

```
for (i=2; i<=10; i++)
```

is written instead of the **statement**:

for(i=1; i<=10; i++), then the **output on the screen** is:

```
value of i = 2  value of i = 3  value of i= 4  value of i= 5  value of i= 6 value of i
= 7 value of i= 8  value of i = 9  value of i= 10
```

If the *for* loop statement:

```
for (i=1; i<10; i++)
```

is written instead of the statement:

```
for (i=1; i<=10; i++)
```

then the output on the screen is:

```
value of i = 1 value of i = 2  value of i= 3  value of i= 4  value of i= 5  value of i=
6 value of i = 7 value of i= 8  value of i = 9
```

- **Note:** the condition i<=10 tells to print till value of i =10 but the **condition** i<10 tells to print till value of i=9.

If the statement:

```
for(i=1; i=10; i++)
```

is written instead of the statement:

```
for(i=1; i<=10; i++)
```

then the output on the screen is:

```
value of i = 10  value of i = 10  value of i = 10  value of  i = 10  value of i= 10
value of i= 10 value of i = 10 value of i= 10  value of i = 10   value of i = 10
value of i = 10   value of i = 10  value of i = 10    value of i = 10   value of i = 10

continues ....
```

Note:

If the statement:

```
printf("value of i =%d", i);
```

is replaced by the statement

```
printf("%d\n", i);
```

Then the **output on the screen** is:

```
1
2
3
4
5
6
7
8
9
10
```

- **What will be the output of the following program :**

```
#include<stdio.h>
int main()
{
int i;
for (i =1; i<=5; i ++)
printf("Linux is not portable\n", i);
return 0;
}
```

Answer:

```
Linux is not portable
Linux is not portable
Linux is not portable
Linux is not portable
Linux is not portable
```

- **C program to print the first ten natural numbers using while loop statement**

The syntax of while loop statement is:

```
while (this is the condition)
{
execute this statement;
}
```

```
#include<stdio.h>
int main()
{
int i = 1;
while (i<=10)
{
printf("%d\n", i++);
}
return 0;
}
```

The output on the screen is:

```
1
2
3
```

```
4
5
6
7
8
9
10
```

`(i<=10)` is the condition and

The statement

```
printf("%d\n", i++);
```

is repeatedly executed as long as a given condition (i<=10) is true.

If the statement:

int i=1;

is replaced by the statement:

int i;

Then the *compilation error* will be displayed on the **console screen** because initialization is not defined i.e., from where to start is not declared.

If the statement:

int i = 1;

is replaced by

```
int i = 0;
```

Then the **output on the screen** is:

```
0
1
2
3
4
5
6
7
8
9
10
```

Similarly if the statement int i = 0; is replaced by the **int i = 7;**

Then the output on the screen is:

```
7
8
9
10
```

- **C program to print first 10 numbers using do while loop statement**

The syntax of do while loop statement is:

```
do
{
execute this statement;
}
while(this is the condition);
```

```
#include<stdio.h>
int main()
```

```
{
int i =1;
do
{
printf("i= %d\n", i++);
} while (i<=10);
return 0;
}
```

The output on the screen is:

```
i=1
i=2
i=3
i=4
i=5
i=6
i=7
i=8
i=9
i=10
```

The statement:

```
printf("i= %d\n", i++);
```

is executed and then condition (i<=10) is checked. If condition (i<=10) is true then

The statement:

```
printf("i= %d\n", i++);
```

is executed again. This process repeats until the given **condition (i<=10)** becomes false.

- ***Why LOOP is USED?***

If loop is not used then the C program to print first 10 numbers should be written as follows:

```
#include<stdio.h>
int main()
{
printf("\n i = 1");
printf("\n i = 2");
printf("\n i = 3");
printf("\n i = 4");
printf("\n i = 5");
printf("\n i = 6");
printf("\n i = 7");
printf("\n i = 8");
printf("\n i = 9");
printf("\n i = 10");
return 0;
}
```

It takes pretty long time to write the code and the execution time is pretty long i.e., Because to reduce the time taken to write the code and to reduce the execution time − loop statement is used.

- **Write a program to print:**

Never test for an error condition you don't know how to handle

5 times using for loop statement.

Answer:

```
#include<stdio.h>
int main()
{
int i;
for (i =1; i<=5; i ++)
printf("Never test for an error condition you don't know how to handle \n");
return 0;
}
```

- **Note:**

For the program:

```
#include<stdio.h>
int main()
{
int i;
for (i=1; i=5; i++)
printf("Linux is not portable");
return 0;
}
```

The output on the screen is:

```
Linux is not portable Linux is not portable Linux is not portable Linux is not portable
Linux is not portable Linux is not portable Linux is not portable Linux is not portable
Linux is not portable Linux is not portable Linux is not portable Linux is not portable
Linux is not portable …. continues
```

- **Program 2.9**

  C program to print the characters from A to Z using for loop, do while loop and while loop statement.

  - **C program to print the characters from A to Z using *for* loop statement:**

```
#include<stdio.h>
int main()
{
char a;
for( a='A'; a<='Z'; a++)
printf("%c\n", a);
return 0;
}
```

The output on the screen:

```
A
B
C
D
E
F
G
H
I
J
K
L
M
N
```

```
O
P
Q
R
S
T
W
X
Y
Z
```

char means the ***data type*** is character.

char → Keyword used to denote the character type data and takes 8-bits for storage. A character constant can be termed as any single character enclosed within a pair of apostrophes `['a','c','2','$','?', etc.]`. A string constant is termed as a sequence of printable ASCII characters placed between double quotes.

Examples: `"computer","keyboard","Hello","Ram", etc.`

The statement

`char a;` imply that we are creating the character a.

Since char a is used. Therefore: the **format specifier** %c should be used instead of %d or %f otherwise error will be flagged on the screen.

If the statement

```
for(a=A; a<=Z; a++)
```

is written instead of the statement

```
for(a='A'; a<='Z'; a++)
```

Then the compilation error will be displayed on the console screen.

- **Which loop to Select?**

Selection of a loop is always a tough task for a programmer, to select a loop do the following steps:

- Analyze the problem and check whether it requires a pre-test or a post-test loop.
- If pre-test is required, use while or for loop.
- If post-test is required, use do-while loop.

- **C program to print the characters from A to Z using while loop statement:**

```
#include<stdio.h>
int main()
{
char a = 'A';
while (a<='Z')
{
printf("%c\n", a++);
}
return 0;
}
```

- **C program to print the characters from A to Z using do while loop statement:**

```
#include<stdio.h>
int main()
{
char a = 'A';
do
{
printf(" %c\n", a++);
} while (a<='Z');
return 0;
}
```

- **Program 3.0**

  C program to print the given number is even or odd.

```
#include<stdio.h>
int main()
{
int a;
printf("Enter any number:");
scanf ("%d", &a);
if(a%2 = = 0)
{
printf("the number is even");
}
else
{
printf("the number is odd");
}
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 4
the number is even will be outputted on the screen.
```

- Mathematical symbol % denote modulus and (a%2 == 0) is the condition and this condition imply: **a divided by 2 yields reminder = 0**.

For example: if you enter the number 4

Then a = 4

Then 4 divided by 2 yields the remainder = 0

Then the statement

```
{
printf("the number is even");
}
```

is executed to print the output:

the number is even

(**Note:** in C language = = implies equal to)

Suppose if you enter the number 3

Then a = 3

Then 3 divided by 2 yields the remainder = 1

Then the statement

```
{
printf("the number is odd");
}
```

is executed to print the output:

the number is odd

- **Program 3.1**

  C program to print the remainder of two numbers

```
#include<stdio.h>
int main()
{
int a, b, c;
printf("Enter any number:");
scanf("%d", &a);
printf("Enter any number:");
scanf("%d", &b);
c = a%b;
printf("the remainder of a and b = %d", c);
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 3
Enter any number:
If you enter the number 2
the remainder of a and b = 1 will be outputted on the screen.
```

Since (a=3 and b=2). Therefore:

3 divided by 2 (i.e., a divided by b) yields the remainder equal to 1

If the statement:

```
printf("the remainder of a and b = %d", c);
```

is replaced by the statement:

```
printf("the remainder of %d and %d = %d", a, b, c);
```

Then the output on the screen is:

```
Enter any number:
If you enter the number 3
Enter any number:
If you enter the number 2
the remainder of 3 and 2 = 1 will be outputted on the screen.
```

- **Program 3.2**

  C program to check the equivalence of two numbers.

```
#include<stdio.h>
int main()
{
int x, y;
printf("Enter any number:");
scanf ("%d", &x);
printf("Enter any number:");
scanf ("%d", &y);
if(x-y==0)
{
printf("the two numbers are equivalent");
}
else
{
printf("the numbers are not equivalent");
}
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 2
the two numbers are equivalent will be outputted on the screen.
```

Since 2−2 is equal to 0 (i.e., x−y = = 0).

Therefore: the statement

{

printf("the two numbers are equivalent");

}

is executed to print the output:

two numbers are equivalent

If you enter the integers 3 and 2

The output on the screen:

the two numbers are not equivalent

Since 3−2 is not equal to 0 (i.e., x−y != 0). Therefore: the statement

```
{
printf("the two numbers are not equivalent");
}
```

is executed to print the output:

two numbers are not equivalent

(as said earlier: in **C language** the symbol != implies not equal to)

- **What is the mistake in the following program:**

```
#include<stdio.h>
int main()
{
int year;
year =1996;
if(year%4==0)
printf("leap year");
else
printf("not a leap year");
return 0;
}
```

Answer:

There is no mistake in the above program. **The output on the screen is:**

leap year

Since year=1996. Therefore:

1996 divided by 4 (i.e., year divided by 4) yields the remainder equal to 0.

The statement

```
printf("leap year");
```

is executed to print the output:

leap year

If the year is = 1995. Then

1995 divided by 4 (i.e., year divided by 4) yields the remainder not equal to 0.

The statement

```
printf("not a leap year");
```

is executed to print the output:

not a leap year

- **Note:** for a year to be leap year, year divided by 4 should yield remainder = zero.

"An algorithm must be seen to be believed."

– **Donald Knuth**

Local variables:

Variable whose existence is known only to the main program or functions are called local variables. Local variables are declared within the main program or a function.

Global variables:

Variables whose existence is known to the both **`main()`** as well as other functions are called global variables. Global variables are declared outside the **`main()`** and other functions.

```
#include<stdio.h>
int x=1; → global variable
main()
{
int y=3; → local variable
}
```

What will be the output of the following programs?

```
#include <stdio.h>
int main () {
```

```
char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
printf("Greeting message: %s\n", greeting );
return 0;
}
```

```
#include <stdio.h>
#define LENGTH 10
#define WIDTH 5
#define NEWLINE '\n'

int main() {
int area;
area = LENGTH * WIDTH;
printf("value of area : %d", area);
printf("%c", NEWLINE);
return 0;
}
```

- **Program 3.3**

  C program to print whether the given number is positive or negative

```
#include<stdio.h>
int main()
{
int a;
a = -35;
if(a>0)
{
printf("number is positive");
}
else
{
printf(" number entered is negative");
}
return 0;
}
```

The output on the screen:

```
number entered is negative
```

Since a = −35. Therefore: **`a is less than 0`** i.e., a < 0 because any negative number is always less than zero.

The statement

```
{
printf("number is negative");
}
```

is executed to print the output:

```
number entered is negative
```

- **Program 3.4**

  C program to print the sum of the first 10 digits using for loop statement

```
#include<stdio.h>
int main()
{
int i, sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
printf("sum of the first 10 digits =%d", sum);
return 0;
}
```

The output on the screen:

```
sum of the first 10 digits = 55
```

- **How the sum of the first 10 digits = 55 is outputted on the screen through the for Loop statement?**

```
i=1 (sum = 0 because the sum is initialized to 0 in the statement int i, sum = 0;)
Is i<=10 true?
Yes, do this
sum = sum + i = 0 +1 =1
Now,
i=2 (sum = 1)
Is i<=10 true?
Yes, do this
sum = sum + i = 1 +2 =3
Now,
i=3 (sum = 3)
Is i<=10 true?
Yes, do this
sum = sum + i = 3 +3 = 6
Now,
i=4 (sum = 6)
Is i<=10 true?
Yes, do this
sum = sum + i = 6 + 4= 10
Now,
i=5 (sum = 10)
Is i<=10 true?
Yes, do this
sum = sum + i = 10 + 5= 15
Now,
i=6 (sum = 15)
Is i<=10 true?
Yes, do this
sum = sum + i = 15 + 6 = 21
Now,
i=7 (sum = 21)
Is i<=10 true?
```

```
Yes, do this
sum = sum + i = 21 + 7 = 28
Now,
i=8 (sum = 28)
Is i<=10 true?
Yes, do this
sum = sum + i = 28 + 8 = 36
Now,
i=9 (sum = 36)
Is i<=10 true?
Yes, do this
sum = sum + i = 36 + 9 = 45
Now,
i=10 (sum = 45)
Is i<=10 true?
Yes, do this
sum = sum + i = 45 + 10 = 55
stops because the condition i<=10 is achieved
```

The statement:

```
{
printf("sum of the first 10 digits =%d", sum);
}
```

is executed to print the output:

```
sum of the first 10 digits = 55
```

If the statement:

```
int i, sum = 0;
```

is replaced by

```
int i, sum = 1;
```

Then the **output on the screen** is:

```
sum of the first 10 digits = 56
```

- **What will be the output if the for loop statement** for(i =1; i<=10; i++) **is replaced by the statement** for(i =2; i<10; i++)**?**

Answer: sum of 10 digits = 44

If the statement

```
int i, sum, sum = 0;
```

is written instead of

```
int i, sum = 0;
```

Then the **compilation error message** will be displayed on the screen (stating that sum is twice declared).

If the *for* loop is ended with a semicolon i.e.,

```
for( i=1; i<=10; i++);
```

Then the **compilation error** will be displayed on the console screen.

- **Program 3.5**

  C program to print the average of the first 10 numbers using for loop statement

```
#include<stdio.h>
int main()
{
```

```
int i, avg, sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
printf("sum of the first 10 numbers =%d", sum);
printf("average of the first 10 numbers =%d", avg);
return 0;
}
```

The output on the screen:

```
sum of the first 10 numbers = 55
average of the first 10 numbers = 5
```

The average of the first10 numbers = 55/10 = 5.5 not 5. But the **output on the screen** is: `average of the first 10 numbers = 5` because int is used instead of float.

If the **data type** float is used i.e.,

```
#include<stdio.h>
int main()
{
float i, avg, sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
printf("sum of the first10 numbers =%f", sum);
printf("average of the first10 numbers = %f", avg);
return 0;
}
```

The output on the screen:

```
sum of the first 10 numbers = 55
average of the first 10 numbers = 5.5
```

- **Program 3.6**

  C program to print the product of the first 10 digits using for loop statement

```
#include<stdio.h>
int main()
{
int i, product = 1;
for( i=1; i<=10; i++)
product = product * i;
printf("the product of the first 10 digits =%d", product);
return 0;
}
```

The output on the screen:

```
the product of the first 10 digits = 3628800
```

▪ **How the product of the first 10 digits = 3628800 is outputted on the screen through the for Loop statement?**

```
i=1 (product = 1 because the product is initialized to 1 in the statement int i,
product = 1;)
Is i<=10 true?
Yes, do this
product = product *  i = 1 * 1 =1
Now,
i=2 (product = 1)
Is i<=10 true?
Yes, do this
product = product *  i = 1 * 2 = 2
Now,
i=3 (product = 2)
Is i<=10 true?
Yes, do this
product = product *  i = 2 * 3 = 6
```

```
Now,
i=4 (product = 6)
Is i<=10 true?
Yes, do this
product = product *  i = 6 * 4 = 24
Now,
i=5 (product =24)
Is i<=10 true?
Yes, do this
product = product *  i = 24 * 5 =120
Now,
i=6 (product =120)
Is i<=10 true?
Yes, do this
product = product *  i = 120 * 6 = 720
Now,
i=7 (product =720)
Is i<=10 true?
Yes, do this
product = product *  i = 720 * 7 = 5040
Now,
i=8 (product =5040)
Is i<=10 true?
Yes, do this
product = product *  i = 5040 * 8 = 40320
Now,
i=9 (product = 40320)
Is i<=10 true?
Yes, do this
product = product *  i = 40320 * 9 = 362880
Now,
i=10 (product = 362880)
Is i<=10 true?
Yes, do this
product = product *  i = 362880 * 10 = 3628800

stops because the condition i<=10 is achieved.
```

The statement:

`printf("the product of the first 10 digits =%d", product);` is executed to display the output:

the product of the first 10 digits = 3628800

If the statement int i, product = 1; is replaced by int i, product = 0;

Then the output on the screen is:

the product of the first 10 digits = 0

If the statement

```
for(i=1; i<=10; i++)
```

is replaced by

```
for(i=5; i<=8; i++)
```

Then the output on the screen is:

the product of the first 10 digits = 1680

- **Program 3.7**

  C Program to print the table of a number using the *for* loop statement

```
#include<stdio.h>
int main()
{
int n, i;
printf("Enter any number:");
scanf("%d", &n);
for( i=1; i<=5; i++)
printf("%d * %d = %d\n", n, i, n*i);
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2 (i.e., n=2)
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
will be outputted on the screen.
```

- **How the execution takes its Way through the for Loop statement**

```
Since you entered the number 2, therefore: n=2.
i=1
Is i<=5 true?
Yes, print this
2 * 1 = 2
using the statement printf("%d * %d = %d\n", n, i, n*i);

Now,
i=2
Is i<=5 true?
Yes, print this
2 * 2 = 4
using the statement printf("%d * %d = %d\n", n, i, n*i);

Now,
i=3
Is i<=5 true?
Yes, print this
2 * 3 = 6
using the statement printf("%d * %d = %d\n", n, i, n*i);

Now,
i=4
Is i<=5 true?
Yes, print this
2 * 4 = 8
using the statement printf("%d * %d = %d\n", n, i, n*i);

Now,
i=5
Is i<=5 true?
Yes, print this
2 * 5 = 10
using the statement printf("%d * %d = %d\n", n, i, n*i);

stop Now because the condition i <=5 is achieved.
```

If the symbol * is replaced by +

i.e.,

```
#include<stdio.h>
int main()
{
int n, a;
printf("Enter any number:");
scanf("%d", &n);
for( i=1; i<=5; i++)
printf("%d + %d = %d\n", n, i, n+ i);
return 0;
}
```

Then the output on the screen is:

```
Enter any number:
If you enter the number 2 (i.e., n=2)

2 + 1 = 3
2 + 2 = 4
2 + 3 = 5
```

```
2 + 4 = 6
2 + 5 = 7

will be outputted on the screen.
```

- **Program 3.8**

  C program:

  If you enter a character M

  Output must be: ch = M

```
#include<stdio.h>
int main()
{
char M;
printf("Enter any character:");
scanf("%c", &M);
printf("ch=%c", M);
return 0;
}
```

The output on the screen:

Enter any character:

If you enter the character M

ch = M will be outputted on the screen.

- **Note:**

```
getchar() function is simplified version of the scanf function
```

If we replace the statement

```
scanf("%c", &M);
```

by the statement:

```
M = getchar();
```

i.e.,

```
#include<stdio.h>
int main()
{
char M;
printf("Enter any character:");
M = getchar();
printf("ch=%c", M);
return 0;
}
```

There will be no change in the output on the screen i.e., **The output on the screen is:**

Enter any character:

If you enter the character K

ch = K will be outputted on the screen.

```
putchar() function is simplified version of the printf function
```

If we replace the statement

```
printf("ch=%c", M);
```

by the statement:

```
putchar (M);
```

i.e.,

```
#include<stdio.h>
```

```
int main()
{
char M;
printf("Enter any character:");
scanf("%c", &M);
putchar (M);
return 0;
}
```

Then there will be no change in the output on the screen i.e., **The output on the screen is:**

```
Enter any character:
If you enter the character M
M will be outputted on the screen.
```

If you replace the statement

```
scanf("%c", &M);
```

by the statement:

```
M = getchar();
```

and the statement

```
printf("ch=%c", M);
```

by the statement:

```
putchar (M);
```

i.e.,

```
#include<stdio.h>
int main()
{
char M;
printf("Enter any character:");
M = getchar();
putchar (M);
return 0;
}
```

The output on the screen:

```
Enter any character:
If you enter the character S
S will be outputted on the screen.
```

- **Program 3.9**

  C program to print the first 5 numbers starting from one together with their squares.

```
#include<stdio.h>
int main()
{
int i;
for( i=1; i<=5; i++)
printf("number=%d its square=%d\n", i , i*i);
return 0;
}
```

The output on the screen:

```
number=1 its square=1
number=2 its square=4
number=3 its square=9
number=4 its square=16
number=5 its square=25
```

- **How the execution takes its way through the for loop statement**

```
i=1
```

```
Is i<=5 true?
Yes, print this
number=1 its square=1
using the statement printf("number=%d its square=%d\n", i , i*i);

Now,
i=2
Is i<=5 true?
Yes, print this
number=2 its square=4
using the statement printf("number=%d its square=%d\n", i , i*i);

Now,
i=3
Is i<=5 true?
Yes, print this
number=3 its square=9
using the statement printf("number=%d its square=%d\n", i , i*i);

Now,
i=4
Is i<=5 true?
Yes, print this
number=4 its square=16
using the statement printf("number=%d its square=%d\n", i , i*i);

Now,
i=5
Is i<=5 true?
Yes, print this
number=5 its square=25
using the statement printf("number=%d its square=%d\n", i , i*i);

stop Now because the condition (i<=5) is achieved.
```

- **Note:**

If the statement

```
printf("number=%d its square=%d\n", i , i*i);
```

is replaced by the statement:

```
printf("\n number=%d/t its square=%d", i , i*i);
```

Then the output on the screen is:

```
number=1      its square=1
number=2     its square=4
number=3     its square=9
number=4     its square=16
number=5     its square=25
```

tab /t is included because to leave space between

```
number=1   and   its square=1
```

Suppose

```
printf("number=%d its square=%d", a , a*a);
```

is replaced by the statement:

```
printf("number=%d\n its square=%d\n", a , a*a);
```

The output on the screen is:

```
number=1
its square=1
number=2
its square=4
number=3
its square=9
number=4
its square=16
number=5
its square=25
```

If you replace the printf statement:

```
printf("number=%d its square=%d", a , a*a);
```

by the statement:

```
printf("number=%d\n, its square=%d\n", a , a*a);
```

i.e., if you place **variable separator** (i.e., comma) between **`number=%d\n`** and **`its square=%d\n`** Then the `compilation error` will be displayed on the screen.

- **Write a program to print the first 10 numbers starting from one together with their squares and cubes?**

Answer:

```
#include<stdio.h>
int main()
```

```
{
int i;
for( i=1; i<=10; i++)
printf("number=%d its square=%d its cube=%d\n", i , i*i, i*i*i);
return 0;
}
```

Rules for switch statement:

- A switch is a decision making construct in 'C.'
- A switch is used in a program where multiple decisions are involved.
- A switch must contain an executable test-expression.
- Each case must include a break keyword.
- Case label must be constants and unique.
- The default is optional.

Pointer → a variable that can hold the address of other variables, arrays, structures, unions and functions that are used in the C program. It contains only the memory location of the variable rather than its content. The purpose of pointer is to save memory space and achieve faster execution time.

Advantages:

- Dynamic memory allocation.
- More compact and efficient coding.
- To return multiple values via functions.
- To point to different data structures.

Operators used with pointers

There are 2 basic operators used with pointers:

- The address operator: & (ampersand)
- The indirection operator: * (asterisk)

The *address operator* gives the address of a variable while the *indirection operator* gives the value of the variable that the pointer is pointing to.

Suppose x is a variable that holds data of type int. We can access the address of x through an address operator [&]. This address can be stored in some variable [say y]. Thus we can relate x and y as

y =&x

y is called a pointer variable because it holds the address of another variable [i.e., x] but not the actual value of x.

There are two integer variables 100 and 640 stored respectively at memory locations 1000 and 1003. Let us assign

x =100;

y =640;

In order to access the value of a variable x via pointer, we must need a variable. Assume that variable be px. The following statement assigns the address of x to px:

px= &x = 1000

In order to access the value of a variable y via pointer, we must need a variable. Assume that variable be py. The following statement assigns the address of y to py:

```
py= &y = 1003
```

x, y, px and py are declared in the program as:

```
int x, y, *px, *py;
```

x and y are integer variables. The asterisk that appears before px and py indicates that the variables are pointer variables.

- **Program 4.0**

  C program to print the sum of two numbers using pointers

If we create an integer variable x by declaring the statement:

```
int x;
```

within the body of the **main** function `int main()` – this variable is stored in the computer memory i.e., this variable occupies a specific location in the space of **computer memory**. And this integer variable x is assigned an address (i.e., &x) to locate its position in the computer memory (like a house in the street is assigned an address to locate its position in the street). Pointers are the variables that represent the address of x in the computer memory i.e., p = &x, where &x imply the **address of x** in the computer memory and p is the pointer variable (which is the variable that represent the address of x in the computer memory). And further if you assign a value to the **variable x** by declaring the statement:

```
x=1;
```

within the body of the main function — this value is stored in the address of x in the computer memory. "*" denote pointer operator and *p denote the pointer (which represent the value stored in the **address of x** in the computer memory).

- **C program to print the address of x and the value assigned to x**

```
#include <stdio.h>
int main()
{
int x, *p;
x = 1;
p = &x;
printf("The address of the variable x =%d", p);
printf("The value of the variable x =%d", *p);
return 0;
}
```

The output on the screen:

```
The address of the variable x = 0x7fffc60478a4
The value of the variable x = 1
```

Since p = &x:

```
*p= *&x
```

The value of the **variable** x = 1 because you have assigned a value to the variable x by declaring the statement:

```
x=1;
```

within the body of the main function.

If the statements:

```
printf("The address of the variable x =%d", p);
printf("The value of the variable x =%d", *p);
```

are replaced by the statement:

```
printf("The address of the variable x =%d and its value =%d", p,*p);
```

i.e.,

```
#include <stdio.h>
int main()
{
int x, *p;
x=1;
p = &x;
printf("The address of the variable x =%d and its value =%d", p,*p);
return 0;
}
```

Then the **output on the screen** is:

```
The address of the variable x = 0x7fffc60478a4and its value = 1
```

```
#include <stdio.h>
int main()
{
int x, y, *p, *q, sum;
printf("Enter any number:");
scanf("%d", &x);
printf("Enter any number:");
scanf("%d", &y);
p = &x;
q = &y;
sum = *p + *q;
printf("Sum of entered numbers = %d\n", sum);
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 3
Sum of entered numbers = 5 will be outputted on the screen.
```

Since *pointer* *p imply the value assigned to the variable x (i.e., 2) through the keyboard and the pointer *q imply the value assigned to the variable y (i.e., 3) **through the keyboard**. Therefore: sum = *p + *q = 2 + 3 = 5 (which will be outputted on the screen).

- **C program to print the product, subtraction and division of two numbers using pointers**

```
#include <stdio.h>
int main()
{
int x, y, *p, *q, product, subtract, div;
printf("Enter any number:");
scanf("%d", &x);
printf("Enter any number:");
scanf("%d", &y);
p = &x;
q = &y;
product = *p * *q;
subtract = *p - *q;
div= *p / *q;
printf("product of entered numbers = %d\n", product);
printf("subtract of entered numbers = %d\n", subtract);
printf("division of entered numbers = %d\n", div);
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 4
Enter any number:
If you enter the number 2
product of entered numbers = 8
subtract of entered numbers = 2
division of entered numbers = 2
will be outputted on the screen.
```

- **C program to find the greatest of two numbers using pointers**

```
#include<stdio.h>
int main()
{
int x, y, *p, *q;
printf("Enter any integer:");
scanf("%d", &x);
printf("Enter any integer:");
scanf("%d", &y);
p = &x;
q = &y;
if(*p>*q)
{
printf("x is greater than y");
}
if(*q>*p)
{
printf("y is greater than x");
}
return 0;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 10
Enter any integer:
If you enter the integer 16
y is greater than x will be outputted on the screen.
```

- **What is the output of the following programs:**

i)

```
#include <stdio.h>
int main()
{
int x;
x=12;
printf("per = %d%", x);
return 0;
}
```

Answer:

```
per=12
```

ii)

```
#include <stdio.h>
int main()
{
int x, t, c;
x =12;
t = 2;
c = x/t;
printf("velocity = %d m/s", c);
return 0;
}
```

Answer:

```
velocity = 6 m/s
```

- **Program 4.1**

C program to print the sum of two numbers using functions

```
#include<stdio.h>
int addition();
int main()
{
int answer;
answer = addition();
printf("The sum of two numbers is: %d\n", answer);
return 0;
}

int addition()
{
int x, y;
printf("Enter any integer:");
scanf("%d", &x);
printf("Enter any integer:");
scanf("%d", &y);
return x+y;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 3
Enter any integer:
If you enter the integer 5
sum of two numbers = 8 will be displayed on the screen.
```

Function	**Purpose**
strlen()	This function is used for finding a length of a string. It returns how many characters are present in a string excluding the NULL character.
strcat(str1, str2)	This function is used for combining two strings together to form a single string. It Appends or concatenates str2 to the end of str1 and returns a pointer to str1.
strcmp(str1, str2)	This function is used to compare two strings with each other. It returns 0 if str1 is equal to str2, less than 0 if str1 < str2, and greater than 0 if str1 > str2.

```
int addition();
```

the statement implies function declaration

int means integer and int addition() implies: addition() should return integer value.

```
int addition()
```

the function to add the entered values (i.e., 3 and 5) and return the result (i.e., 3 + 5 i.e., 8) to the statement:

printf("sum of two numbers = %d", answer); to make provision to display the output:

```
sum of two numbers = 8
```

```
{
int x, y;
printf("Enter any integer:");
scanf("%d", &x);
printf("Enter any integer:");
scanf("%d", &y);
return x+y;
}
```

the body of the function int addition()

```
answer = addition();
```

the function call i.e., this statement calls the function:

addition()

to add the entered values (i.e., 3 and 5) and return the result (i.e., 3 + 5 i.e., 8) to the statement:

```
printf("sum of two numbers = %d", answer);
```

to make provision to display the output:

```
sum of two numbers = 8
```

on the screen.

In the statement:

```
printf("sum of two numbers=%d", answer);
```

the format string %d indicates that the value to be displayed at that point in the string i.e., after the statement:

```
sum of two numbers =
```

needs to be taken from the result returned by the function `int addition()`.

- **C program to print the product of two numbers using functions**

```
#include<stdio.h>
int multiplication();
int main()
{
int answer;
answer = multiplication();
printf("The product of two numbers is: %d\n", answer);
return 0;
}

int multiplication()
{
int x, y;
printf("Enter any integer:");
scanf("%d", &x);
printf("Enter any integer:");
scanf("%d", &y);
return x*y;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 3
Enter any integer:
If you enter the integer 5
product of two numbers = 15 will be outputted on the screen.
```

- **C program to print the greatest of two numbers using functions**

```
#include<stdio.h>
int largest();
int main()
{
int answer;
answer = largest();
printf("The largest of two numbers is: %d\n", answer);
return 0;
}

int largest()
{
int x, y;
printf("Enter any integer:");
scanf("%d", &x);
printf("Enter any integer:");
scanf("%d", &y);
if(x>y)
return x;
if(y>x)
return y;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 3
Enter any integer:
If you enter the integer 5
largest of two numbers= 5 will be outputted on the screen.
```

- **C program to print the greatest of three numbers using functions**

```
#include<stdio.h>
int largest();
int main()
{

int answer;
answer = largest();
printf("largest of three numbers=%d", answer);
return 0;
}
int largest()
{
int x, y, z;
printf("Enter any integer:");
scanf("%d", &x);
printf("Enter any integer:");
scanf("%d", &y);
printf("Enter any integer:");
scanf("%d", &z);
if(x>y&& x>z)
return x;
if(y>x&& y > z)
return y;
if(z>x && z>y)
return z;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 3
Enter any integer:
If you enter the integer 5
Enter any integer:
If you enter the integer 10
largest of three numbers = 10 will be outputted on the screen.
```

- **C program to print the square of the number using functions**

```
#include<stdio.h>
int square();
int main()
{

int answer;
answer = square();

printf("square of the given number=%d", answer);
}
int square()
{
int x;
printf("Enter any integer:");
scanf("%d", &x);
return x*x;
}
```

The output on the screen is:

Enter any integer:
If you enter an integer 5
square of the number = 25 will be outputted on the screen.

- **What is the output of the following program:**

```
#include<stdio.h>
int main()
{
int x;
x=6;
printf("The address of x = %d", &x);
return 0;
}
```

Answer:

```
The address of x = -604171156
```

- **Program 4.2**

if-else statement provides a way for selecting any one of the 2 possible alternatives. And, ***nested-if*** allow us to select one of the many alternatives but it is time consuming. To overcome this, the switch case statement is used. **Switch (case)** allows the user to make decision from the number of choices i.e., **from the number of cases.**

For example:

```
#include<stdio.h>
int main()
{
char ch;
printf("Enter any character:");
scanf("%c", &ch);
switch(ch)
{
case 'R':
printf("Red");
break;
case 'W':
printf("White");
break;
case 'Y':
printf("Yellow");
break;
case 'G':
printf("Green");
break;
default:
printf("Error");
break;
} ──────────→ End of switch
return 0;
} ──────────→ End of main()
```

The output on the screen:

```
Enter any character:
If you enter a character R
Red will be outputted on the screen.
```

`switch(ch)` allow us to make decision from the number of choices i.e., from the number of cases

case 'R':

case 'W':

case 'Y':

case 'G':

Since we have entered the **character** R (which corresponds to case 'R':)

The statement

```
printf("Red");
```

is executed to display the output:

Red

on the screen.

Suppose you enter a **character** K

Then the output on the screen is:

```
Error
```

(Entered character K does not correspond to any of the cases:

case 'R':

case 'W':

case 'Y':

case 'G':

Therefore the statement:

`printf("Error");`

is executed to display the output:

Error

on the screen).

The ***break*** statement denotes the end of a particular case and thereby the switch statement is terminated. The case ***default*** is executed, when the value of an expression is not matched with any of the cases.

If the statements:

```
case 'R':
printf("Red");
break;
case 'W':
printf("White");
break;
case 'Y':
printf("Yellow");
break;
case 'G':
printf("Green");
break;
default:
printf("Error");
break;
```

are replaced by the statements:

```
case 'R':
printf("Red");
case 'W':
printf("White");
case 'Y':
printf("Yellow");
break;
case 'G':
printf("Green");
break;
```

```
default:
printf("Error");
break;
```

Then the **output on the screen** is:

```
Red
White
Yellow
```

i.e., the output will be printed till **yellow** even though you have entered the **character R**.

Arrays

↓

Ordered list of homogeneous data elements

- Elements may be of data type int, float, char or double.
- All the elements are stored in consecutive memory locations [on RAM].

In an array named odd, the individual items are as shown below:

odd[1]	odd[2]	odd[3]	odd[4]	odd[5]

where:

1, 2, 3, 4 and 5 are subscripts

odd[1] : the first element in an array odd.

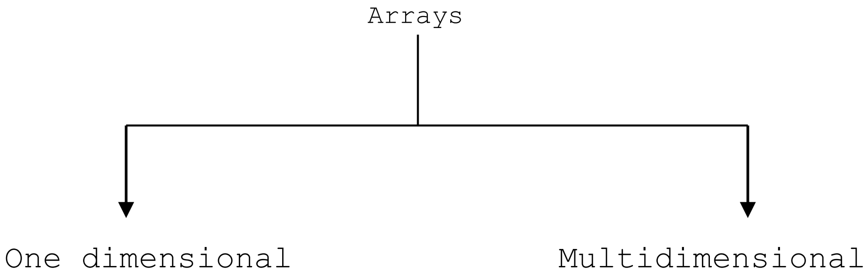
odd[2] : the second element in an array odd.

odd[3] : the third element in an array odd.

odd[4] : the fourth element in an array odd.

odd[5] : the fifth element in an array odd.

In general, odd[i] denote the i^{th} element of an odd array.



- **One dimensional array:** Number of subscripts is 1
- **Multidimensional array:** Number of subscripts is more than 1

array1[] → one-dimensional array

array2[] [] → two-dimensional array

array3[] [] [] → three-dimensional array, and so on.

- **Program 4.3**

  C program to print the output:

  Element [0] = 16

  Element [1] = 18

  Element [2] = 20

  Element [3] = 25

  Element [4] = 36

  using arrays:

```
#include<stdio.h>
int main()
{
```

```
int i;
int num [5] = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
printf("\n Element [%d] = %d", i, num[i]);
return 0;
}
```

The output on the screen:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
```

The statement:

```
int num [5] = {16, 18, 20, 25, 36};
```

imply that we are creating an integer array (and the name of array is num) consisting of 5 values (i.e., 16, 18, 20, 25, 36) of the same **data type int**.

The number of values between the braces { } cannot be larger than the number of values that we declare for the array between square brackets [ ].

There are 5 integers i.e., 16, 18, 20, 25, 36 within the braces { }, so 5 is written within the square brackets [ ].

If there were 6 integers i.e., 16, 18, 20, 25, 36, 42 within the braces { }, then 6 must be written within the square brackets [ ].

- **Note:** With the declaration int num [5], computer creates 5 memory cells with name num[0], num[1], num[2], num[3], num[4].

And since:

```
int num [5] = {16, 18, 20, 25, 36};
```

the values 16, 18, 20, 25, 36 are stored in num[0], num[1], num[2], num[3], num[4] respectively.

- **How the execution takes its way through the *for* loop statement**

```
i=0
Is i<5 true?
Yes, print this
Element [0] = 16
using the statement:
                       printf("\n Element [%d] = %d", i, num[i])
format string %d in the square brackets indicates that the value to be displayed at
that point in the string i.e., within the square brackets [ ] needs to be taken from a
variable (which is i i.e., i=0) and the format string %d after the statement (\n
Element [%d] = ) indicates that the value to be displayed at that point in the string
i.e., after the statement (\n Element [%d] = ) needs to be taken from a variable (which
is stored in num[i] i.e., num[0] i.e., 16).

Now,
i=1
Is i<5 true?
Yes, print this
Element [1] = 18
using the statement:
                       printf("\n Element [%d] = %d", i, num[i])
format string %d in the square brackets indicates that the value to be displayed at
that point in the string i.e., within the square brackets [ ] needs to be taken from a
variable (which is i i.e., i=1) and the format string %d after the statement (\n
Element [%d] = ) indicates that the value to be displayed at that point in the string
i.e., after the statement (\n Element [%d] = ) needs to be taken from a variable (which
is stored in num[i] i.e., num[1] i.e., 18).

Now,
i=2
```

```
Is i<5 true?
Yes, print this
Element [2] = 20
using the statement:
                       printf("\n Element [%d] = %d", i, num[i])
format string %d in the square brackets indicates that the value to be displayed at
that point in the string i.e., within the square brackets [ ] needs to be taken from a
variable (which is i i.e., i=2) and the format string %d after the statement (\n
Element [%d] = ) indicates that the value to be displayed at that point in the string
i.e., after the statement (\n Element [%d] = ) needs to be taken from a variable (which
is stored in num[i] i.e., num[2] i.e., 20).

Now,
i=3
Is i<5 true?
Yes, print this
Element [3] = 25
using the statement:
                       printf("\n Element [%d] = %d", i, num[i])
format string %d in the square brackets indicates that the value to be displayed at
that point in the string i.e., within the square brackets [ ] needs to be taken from a
variable (which is i i.e., i=3) and the format string %d after the statement (\n
Element [%d] = ) indicates that the value to be displayed at that point in the string
i.e., after the statement (\n Element [%d] = ) needs to be taken from a variable (which
is stored in num[i] i.e., num[3] i.e., 25).

Now,
i=4
Is i<5 true?
Yes, print this
Element [4] = 36
using the statement:
                       printf("\n Element [%d] = %d", i, num[i])
Stop because the condition i<5 is achieved.
```

Format string %d in the square brackets indicates that the value to be displayed at that point in the string i.e., with the square brackets [ ] needs to be taken from a variable (which is i i.e., i=4) and the format string %d after the statement (\n Element [%d] =) indicates that the value to be displayed at that point in the string i.e., after the statement (\n Element [%d] =) needs to be taken from a variable (which is stored in num[i] i.e., num[4] i.e., 36).

Suppose the statement:

```
printf("\n Element [%d] = %d", i, num[i]);
```

is replaced by the statement:

```
printf("\n Element [%d] = %d", i, num[0]);
```

Then the **output on the screen**:

```
Element [0] = 16
Element [1] = 16
Element [2] = 16
Element [3] = 16
Element [4] = 16
```

Suppose the statement:

```
printf("\n Element [%d] = %d", i, num[i]);
```

is replaced by the statement:

```
printf("\n Element [%d] = %d", i, num[1]);
```

The output on the screen:

```
Element [0] = 18
Element [1] = 18
Element [2] = 18
Element [3] = 18
Element [4] = 18
```

Suppose the statement:

```
printf("\n Element [%d] = %d", i, num[i]);
```

is replaced by the statement:

```
printf("\n Element [%d] = %d", i, num[2]);
```

The output on the screen:

```
Element [0] = 20
Element [1] = 20
Element [2] = 20
Element [3] = 20
Element [4] = 20
```

Suppose the statement:

```
printf("\n Element [%d] = %d", i, num[i]);
```

is replaced by the statement:

```
printf("\n Element [%d] = %d", i, num[3]);
```

The output on the screen:

```
Element [0] = 25
Element [1] = 25
Element [2] = 25
Element [3] = 25
Element [4] = 25
```

Suppose the statement:

```
printf("\n Element [%d] = %d", i, num[i]);
```

is replaced by the statement:

```
printf("\n Element [%d] = %d", i, num[4]);
```

The output on the screen:

```
Element [0] = 36
Element [1] = 36
Element [2] = 36
Element [3] = 36
Element [4] = 36
```

If the condition:

i<5

is replaced by the condition:

i<=5

Then the **output on the screen** is:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
Element [5] = 3656
```

- 3656 is the number stored in the memory i.e., any number stored in the memory will be displayed.

If the statement:

int num [5] = {16, 18, 20, 25, 36}; is replaced by the statement:

int num [i] = {16, 18, 20, 25, 36};

Then the **compilation error** will be displayed on the screen because there are 5 elements within the braces {} not i elements.

- **Note:**

  - **C program to print the sum of the elements in array.**

```
#include<stdio.h>
int main()
{
int i, sum = 0;
int num [5] = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
sum = sum + num[i];
printf("Sum of the Elements in the array = %d", sum);
return 0;
}
```

The output on the screen:

Sum of the Elements in the array = 115

i.e., 16 + 18 + 20 + 25 + 36 = 115

- **How the Execution takes its way through the for loop statement**

```
i=0 (sum = 0)
Is i<5 true?
Yes, do this
sum = sum + num[i] = sum + num[0] = 0 +16 =16
```

```
Now,
i=1 (sum = 16)
Is i<5 true?
Yes, do this
sum = sum + num[i] = sum + num[1] = 16 +18 =34

Now,
i=2 (sum = 34)
Is i<5 true?
Yes, do this
sum = sum + num[i]  = sum + num[2] = 34 +20 =54

Now,
i=3 (sum = 54)
Is i<5 true?
Yes, do this
sum = sum + num[i] = sum + num[3] = 54 +25 =79

Now,
i=5 (sum = 79)
Is i<5 true?
Yes, do this
sum = sum + num[i] = sum + num[5] = 79 + 36 =115
stop because the condition i<5 is achieved

The statement:
printf("Sum of the Elements in the array  = %d", sum); is executed to display the
output:
Sum of the Elements in the array = 115
on the screen.

If the statement:
int i, sum = 0;
is replaced by int i, sum = 1;
Then The output on the screen:
Sum of the Elements in the array = 116
```

- **C program to print the average of the elements in array**

```
#include<stdio.h>
int main()
{
int i, avg, sum = 0;
int num [5] = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
sum = sum + num [i];
avg = sum/5;
printf("Sum of the Elements in the array = %d", sum);
printf("average of the elements in the array= %d", avg);
return 0;
}
```

The output on the screen:

```
Sum of the Elements in the array = 115
average of the elements in the array = 23
```

Write a program to print:

Einstein [0] = E

Einstein [1] = I

Einstein [2] = N

Einstein [3] = S

Einstein [4] = T

Einstein [5] = E

Einstein [6] = I

Einstein [7] = N

using arrays

Answer:

```
#include<stdio.h>
int main()
{
int i;
char name [8] = {' E' , ' I', ' N', ' S', ' T ', ' E', ' I', ' N'};
for(i=0; i<8; i++)
printf("\n Element [%d] = %c", i, name[i]);
return 0;
}
```

- **Note:**

If the format string %d is used instead of %c i.e., if the statement:

```
printf("\n Element [%d] = %c", name[i], name[i]);
```

is written instead of the statement:

```
printf("\n Element [%c] = %c", name[i], name[i]);
```

Then the **output on the screen** is:

```
Element [69] = E
Element [73] = I
Element [78] = N
Element [83] = S
Element [84] = T
Element [69] = E
Element [73] = I
Element [78] = N
```

- **What will be the output of the following programs?**

i)

```
#include <stdio.h>
#include <math.h>
int main()
{
printf("%f", cbrt(27));
return 0;
}
```

Answer:

3.000

ii)

```
#include <stdio.h>
int main()
{
char i;
char body [4] = {'b', 'o', 'd', 'y'};
for(i=0; i<4; i++)
printf("\n body[%c] = %c", body[i] , body[i]);
return 0;
}
```

Answer:

```
body [b] = b
body [o] = o
body [d] = d
body [y] = y
```

iii)

```
#include <stdio.h>
```

```
#include <malloc.h>
int main()
{
int x=2;
printf("%d", malloc(200*sizeof(x)));
return 0;
}
```

Answer:

```
8183824
```

- **What is the mistake in the following program:**

```
#include<stdio.h>
int main()
{
int i;
int num [] = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
printf("\n Element [%d] = %d", i, num[i]);
return 0;
}
```

Answer: There is no mistake in the above program. The **output on the screen** is:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
```

- **Program 4.3**

  C program to print the output:

```
Name of the book = B
Price of the book = 135.00
Number of pages = 300
Edition = 8
```

  using structures

```
#include<stdio.h>
int main()
{
struct book {
char name;
float price;
int pages;
int edition;
};

struct book b1;
b1.name = 'B';
b1.price = 135.00;
b1.pages = 300;
b1.edition = 8;
printf("\n Name of the book = %c", b1.name);
printf("\n Price of the book = %f", b1.price);
printf("\n Number of pages = %d", b1.pages);
printf("\n Edition of the book = %d", b1.edition);
return 0;
}
```

The output on the screen:

```
Name of the book = B
Price of the book = 135.00
Number of pages = 300
Edition of the book = 8
```

The statement:

```
struct book {
char name;
float price;
int pages;
int edition;
};
```

imply the structure definition i.e., we are defining a structure (and the data type name of the structure is book) and it consists of elements:

- **name** (which is of data type char), **price** (which is of data type float), **pages** (which is of data type int), **edition** (which is of data type int) – which are placed within the body of the structure.

The statement:

```
struct book b1;
```

imply the structure variable declaration (where b1 denote the structure variable)

- **Why structure variable b1 is declared or defined?**

In order to assign the values to the elements within the body of the structure, each element must be linked with structure variable with dot operator or period operator or member accessibility operator.

- **For example:** name is the element which must be linked with structure variable b1 with dot operator to assign a value B to the element “name”.

Format string %c (corresponding to the **data type** char) in the statement:

```
printf("\n Name of the book = %c", b1.name);
```

indicates that the value to be displayed at that point in the string i.e., after the statement (\n Name of the book =) needs to be taken from b1.name.

The statement:

```
printf("\n Name of the book = %c", b1.name);
```

make provision to print the output:

Name of the book = B

on the screen.

format string %f (corresponding to the data type float) in the statement:

```
printf("\n Price of the book = %f", b1.price);
```

indicates that the value to be displayed at that point in the string i.e., after the statement (\n Price of the book =) needs to be taken from b1.price.

The statement:

printf("\n Price of the book = %f", b1.price);

make provision to print the output:

Price of the book = 135.00

on the screen.

format string %d (corresponding to the **data type** int) in the statement:

```
printf("\n Number of pages = %d", b1.pages);
```

indicates that the value to be displayed at that point in the string i.e., after the statement (\n Number of pages =) needs to be taken from b1.pages.

The statement:

```
printf("\n Number of pages = %d", b1.pages);
```

make provision to print the output:

Number of pages = 300

on the screen.

format string %d (corresponding to the **data type** int) in the statement:

```
printf("\n Edition of the book = %d", b1.edition);
```

indicates that the value to be displayed at that point in the string i.e., after the statement (\n Edition of the book =) needs to be taken from b1.edition.

The statement:

printf("\n Edition of the book = %d", b1.edition);

make provision to print the output:

Edition of the book = 8

on the screen.

- **What will be output of the following programs?**

A)

```
#include<stdio.h>
```

```
struct book {
char name;
float price;
int pages;
int edition;
};
int main()
{
struct book b1;
b1.name = 'B';
b1.price = 135.00;
b1.pages = 300;
b1.edition = 8;
printf("\n Name of the book = %c", b1.name);
printf("\n Price of the book = %f", b1.price);
printf("\n Number of pages = %d", b1.pages);
printf("\n Edition of the book = %d", b1.edition);
}
```

Answer:

```
Name of the book = B
Price of the book = 135.000000
Number of pages = 300
Edition of the book = 8
```

B)

```
#include <stdio.h>
int main(){

for( ; ; ) {
printf("This loop will run forever.\n");
}
return 0;
}
```

Answer:

```
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever. ......... continues
```

C)

```
#include<stdio.h>
int main()
{
char  ch [5];
printf( "Enter the name: ");
scanf("%s", &ch);
printf( "the name you entered = %s", ch);
return 0;
}
```

Answer:

Enter the name:

If you enter the name Dennis

the name you entered = **Denni** will be outputted on the screen.

Instead of Dennis, only Denni will be displayed on the screen because of the statement

```
char  ch [5];
```

The statement:

```
char ch [5];
```

make provision only for 5 lettered name to be displayed on the screen.

If the statement:

```
char ch [5];
```

is replaced by the statement

```
char  ch [6];
```

Then the output on the screen is:

```
Enter the name:
If you enter the name Dennis
the name you entered = Dennis will be outputted on the screen.
```

- **Note:** %s implies the format specifier for string.

- **Program 4.4**

  Continue and break statements:

  i)

```
#include <stdio.h>
int main()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
continue;
}

printf("%d\n ", i);
}
return 0;
}
```

Output on the screen:

```
1
2
4
5
```

- **Note:**

```
i = 1
Is the condition (i<=5) is true?
Yes because i=1
The statement printf("%d\n ", i); is executed to print the output:
1
Now, the value of i is:
i =  1+1 = 2
Is the condition (i<=5) is true?
Yes because i=2
The statement printf("%d\n ", i); is executed to print the output:
2
Now, the value of i is:
i =  2+1 = 3
```

```
Is the condition (i<=5) is true?
Yes because i=3
The statement printf("%d\n ", i); is not executed to print the output:
3

Because of the statement:

if (i==3)
{
continue;
}

// Execution skips //

Now, the value of i is:
i =  3+1 = 4
Is the condition (i<=5) is true?
Yes because i=4
The statement printf("%d\n ", i); is executed to print the output:
4
Now, the value of i is:
i =  4+1 = 5
Is the condition (i<=5) is true?
Yes because i=5
The statement printf("%d\n ", i);  is executed to print the output:
5
and stop because the condition i<=5 is achieved.
```

ii)

```
#include <stdio.h>
int main()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
break;
}

printf("%d\n ", i);
}
return 0;
}
```

Output on the screen:

```
1
2
```

- **Note:**

```
i = 1
Is the condition (i<=5) is true?
Yes because i=1
The statement printf("%d\n ", i); is executed to print the output:
1
Now, the value of i is:
i =  1+1 = 2
Is the condition (i<=5) is true?
Yes because i=2
The statement printf("%d\n ", i); is executed to print the output:
2
Now, the value of i is:
i =  2+1 = 3
Is the condition (i<=5) is true?
Yes because i=3
The statement printf("%d\n ", i); is not executed to print the output:

Because of the statement:

if (i==3)
{
break;
}
```

```
The for loop:
for (i=1; i<=5; i++)

is immediately terminated (even before the condition i<=5 is achieved) and program
execution stops.
```

- **The goto statement:**

```
#include <stdio.h>
int main()
{
int i;
for(i=1; i<=5; i++)
{
if(i==3)
{
goto HAI;
}
printf("\n %d ",i);
}
HAI : printf("\n Linux");
}
```

Output on the screen:

```
1
2
Linux
```

- **Note:**

```
i = 1
Is the condition (i<=5) is true?
Yes because i=1
The statement printf("\n %d ",i); is executed to print the output:
```

```
1
Now, the value of i is:
i =  1+1 = 2
Is the condition (i<=5) is true?
Yes because i=2
The statement printf("\n %d ",i); is executed to print the output:
2
Now, the value of i is:
i =  2+1 = 3
Is the condition (i<=5) is true?
Yes because i=3
The statement printf("%d\n ", i); is not executed to print the output:
3

Rather
The statement printf("\n Linux"); is executed to print the output:

Linux

Because of the statement:

if(i==3)
{
goto HAI;
}

The for loop:
for (i=1; i<=5; i++)

is immediately terminated (even before the condition i<=5 is achieved) and program
execution stops.
```

- **Program 4.5**

  C program to convert the upper case letter to lower case letter

```
#include<stdio.h>
int main()
{
char ch = 'A';
char b = tolower(ch);
printf("upper case letter %c is converted to lower case letter %c", ch, b);
return 0;
}
```

Output on the screen:

upper case letter A is converted to lower case letter a

If you want to enter the **character through the keyboard**, then the above program should take the form:

```
#include<stdio.h>
int main()
{
char ch;
printf("Enter any character:");
scanf("%c", &ch);
char b = tolower(ch);
printf("upper case letter %c is converted to lower case letter %c", ch, b);
return 0;
}
```

Output on the screen:

```
Enter any character:
If you enter the character C
upper case letter C is converted to lower case letter c will be outputted on the
screen.
```

- **Program 4.6**

  C program to convert the lower case letter to upper case letter

```
#include<stdio.h>
int main()
{
char ch = 'a';
char b = toupper(ch);
printf("lower case letter %c is converted to upper case letter %c", ch, b);
return 0;
}
```

Output on the screen:

```
lower case letter a is converted to upper case letter A
```

If you want to enter the character through the **keyboard**, then the above program should take the form:

```
#include<stdio.h>
int main()
{
char ch;
printf("Enter any character:");
scanf("%c", &ch);
char b = toupper(ch);
printf("lower case letter %c is converted to upper case letter %c", ch, b);
return 0;
}
```

Output on the screen:

Enter any character:
If you enter the character h
lower case letter h is converted to upper case letter H will be outputted on the screen.

- **Program 4.7**

C program to test whether the entered character is upper case letter or not

```
#include<stdio.h>
int main()
{
char ch = 'a';
if(isupper(ch))
printf("you have entered the upper case letter");
else
printf("you have entered the lower case letter");
return 0;
}
```

Output on the screen:

you have entered the lower case letter

If the statement:

char ch = 'a'; is replaced by the statement:

char ch = 'A';

Then the **output on the screen** is:

```
you have entered the upper case letter
```

- **Program 4.8**

  C program to test whether the entered character is lower case letter or not

```
#include<stdio.h>
int main()
{
char ch = 'a';
if(islower(ch))
printf("you have entered the lower case letter");
else
printf("you have entered the upper case letter");
return 0;
}
```

Output on the screen:

```
you have entered the lower case letter
```

Functions responsible for dynamic memory management:

Function	Purpose
malloc	Allocates the memory of requested size and returns the pointer to the first byte of allocated space.
calloc	Allocates the space for elements of an array. Initializes the elements to zero and returns a pointer to the memory.
realloc	It is used to modify the size of previously allocated memory space.
Free	Frees or empties the previously allocated memory space.

- **Program 4.9**

  C program to print the value of tan inverse x (i.e., the value of $\tan^{-1}x$)

```
#include<stdio.h>
#include<math.h>
int main()
{
```

```
int x = 20;
printf("the value of tan inverse x = %f", atan(x));
return 0;
}
```

Output on the screen:

```
the value of tan inverse x = 1.520838
```

- **Program 5.0**

  C program to print the value of tan inverse $\frac{x}{y}$ (i.e., the value of $\tan^{-1}\frac{x}{y}$)

```
#include<stdio.h>
#include<math.h>
int main()
{
int x,y;
x = 20;
y =20;
printf("the value of tan inverse x/y = %f",  atan2(x, y));
return 0;
}
```

Output on the screen:

```
the value of tan inverse x/y = 0.785398
```

- **Program 5.1**

  C program to print the value of fmod(x, y)

```
#include<stdio.h>
#include<math.h>
int main()
```

```
{
float x = 20.500000;
float y =20.799999;
printf("the remainder of %f divided by %f is %f", x, y, fmod(x, y));
return 0;
}
```

Output on the screen:

the remainder of 20.500000 divided by 20.799999 is 20.500000

- **Program 5.2**

  C program to print the value of ~x

```
#include<stdio.h>
int main()
{
int x, y;
x = 205;
y=~x;
printf("the value of y is:%d", y);
return 0;
}
```

Output on the screen:

```
the value of y is:-206
```

If the statement:

y=~x; is replaced by the statement:

```
y= -(~x);
```

Then the **output on the screen** is:

```
the value of y is: 206
```

- **Program 5.3**

C program to print the ASCII (American Standard Code for Information Interchange) value of the entered character

```c
#include<stdio.h>
int main()
{
char ch ='A';
printf("the ASCII value of ch is: %d", ch);
return 0;
}
```

Output on the screen:

```
the ASCII value of ch is: 65
```

If the statement:

```c
printf("the ASCII value of ch is: %d", ch);
```

is replaced by the statement:

```c
printf("the ASCII value of ch is: %c", ch);
```

Then the **output on the screen** is:

```
the ASCII value of ch is: A
```

- **What will be the output of the following programs:**

i)

```
#include<stdio.h>
int main()
{
int i;
int num [5] ={16,18,19,20,21};
for(i=0;i<5;i++)
printf("\n Element = %d", num[i] +1);
return 0;
}
```

Answer:

```
Element = 17
Element = 19
Element = 20
Element = 21
Element = 22
```

ii)

```
#include<stdio.h>
int main()
{
int i = 54;
int y = i<<1;
printf("The value of y = %d", y);
return 0;
}
```

Answer:

```
The value of y = 108
```

If the statement:

i<<1 is replaced by the statement: i<<2

Then the **output on the screen** is:

The value of y = 216

- **Note:**

    - i<<1 implies 54 * 2 = 108
    - i<<2 implies 54 * 4 = 216
    - i<<3 implies 54 * 6 = 324
    - i<<4 implies 54 * 8 = 432

iii)

```
#include<stdio.h>
int main()
{
int i = 54;
int y = i>>1;
printf("The value of y = %d", y);
return 0;
}
```

Answer:

The value of y = 27

If the statement:

i>>1 is replaced by the statement: i>>2

Then the **output on the screen** is:

```
The value of y = 13
```

- **Note:**
  - i>>1 implies $\frac{54}{2} = 27$
  - i>>2 implies $\frac{54}{4} = 13$
  - i>>3 implies $\frac{54}{6} = 9$
  - i>>4 implies $\frac{54}{8} = 6$

<< **implies:** left shift operator

>> **implies:** right shift operator

- **Program 5.4**

  C program to print the length of the entered character (i.e., to print the length of the string)

```
#include<stdio.h>
#include<string.h>
int main()
{
char ch[4];
printf("Enter any word: ");
scanf("%c", &ch);
printf("The length of the string = %d", strlen(ch));
return 0;
}
```

Output on the screen:

```
Enter any word:
If you enter the word dog
The length of the string = 3
```

will be displayed on the console screen because **there are three letters** in the word dog.

Suppose if you enter the word tech

The length of the string = 4

will be displayed on the console screen because **there are four letters** in the word tech.

- **Program 5.5**

  C program to print the factorial of the entered number

```
#include<stdio.h>
int main()
{
int i, n, fact=1 ;
printf("Enter any number:");
scanf("%d", &n);
for(i=1; i<=n; i++)
fact = fact *i;
printf("\n Entered number is: %d", n);
printf("\n The factorial of the entered number %d is: %d", n, fact);
return 0;
}
```

Output on the screen:

```
Enter any number:
If you enter the number 2
Entered number is: 2
The factorial of the entered number 2 is: 2
will be displayed on the screen.
```

Suppose if you enter the number 4

Entered number is: 4

The factorial of the entered number 4 is: 24

will be displayed on the screen.

Storage class	**Purpose**
auto	It is a default storage class.
extern	It is a global variable.
static	It is a local variable which is capable of returning a value even when control is transferred to the function call.
register	It is a variable which is stored inside a Register.

- **What will be the output of the following program:**

```
#include <stdio.h>
int main()
{
printf("\nLinux \' linux ");
printf("\nLinux \? linux ");
return0;
}
```

Answer:

```
Linux ' linux
Linux ? linux
```

```
#include<stdio.h>
#include<stdlib.h>
int main () {
printf("linux\n");
exit (0);
printf("php\n");
return 0;
}
```

Answer:

```
linux
```

- **Note:** `exit(0)` is useful for terminating a program upon having discovered some error which prevents the program from continuing to execute normally. The ***header file*** for `exit(0);` is **stdlib.h.**

- **What will be the output of the following program?**

```
#include<stdio.h>
main()
{
int a, b, c;
a=2;
b=2;
c = a ^ b;
printf( " the value of c = %d", c);
}
```

Answer:

0

symbol ^ denote XOR operator.

2^3	2^2	2^1	2^0
8	4	2	1

Since a = 2

2^3	2^2	2^1	2^0
8	4	2	1
0	0	1	0

Since b = 2

2^3	2^2	2^1	2^0
8	4	2	1
0	0	1	0

a ^ b

0	0	1	0
0	0	1	0
0	0	0	0

2^3	2^2	2^1	2^0
8	4	2	1
0	0	1	0
0	0	1	0
0	0	0	0

0 × 8 + 0 × 4 + 0×2 + 0 × 1 = 0

a ^ b = 0

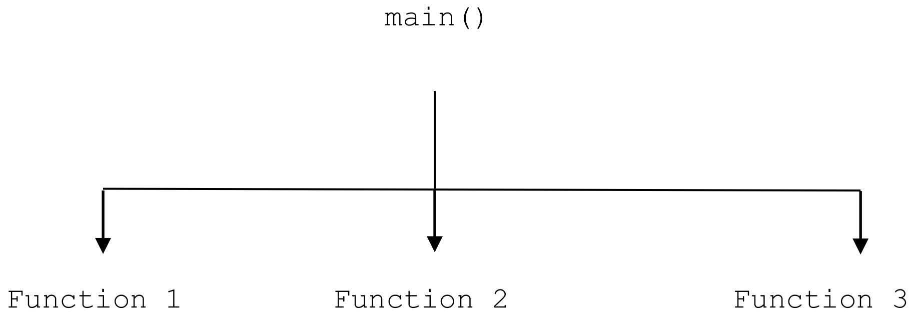
To create a file in a 'C' program following syntax is used:

```
FILE *fp;
fp = fopen ("file_name", "mode");
```

```
#include <stdio.h>
int main() {
FILE *fp;
fp = fopen ("data.txt", "w");
}
```

What is a Function in C?

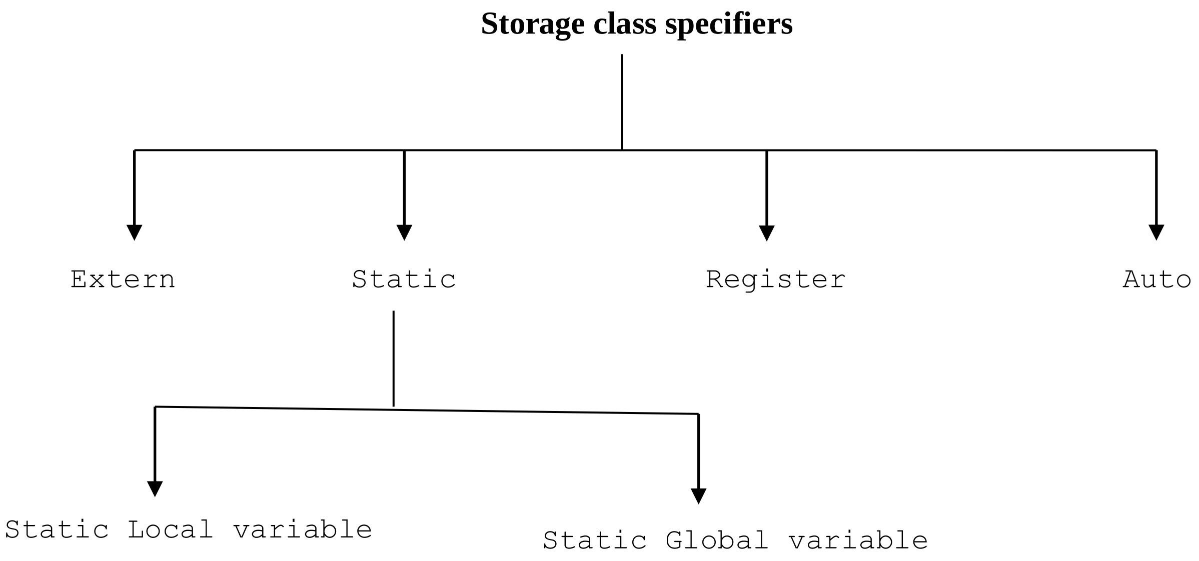
Function in C is a reusable block of code that makes a program easier to understand, test and can be easily modified without changing the calling program. Functions divide the code and modularize the program for better and effective results. In short, a larger program is divided into various subprograms which are called as *functions*.



In 'C' programming functions are divided into three activities such as:

- Function declaration
- Function definition
- Function call

x	**y**	**x & y**	**x | y**	**x ^ y**
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0



```c
#include <stdio.h>
extern int x = 32;
int b = 8;
int main() {
   auto int a = 28;
   extern int b;
   printf("The value of auto variable: %d\n", a);
   printf("The value of extern variables x and b: %d, %d\n",x,b);
   x = 15;
   printf("The value of modified extern variable x: %d\n", x);
   return 0;
}
```

Output on the screen:

```
The value of auto variable: 28
The value of extern variables x and b: 32, 8
The value of modified extern variable x: 15
```

	value of x
`x=7;`	7
`x= x<<1;`	14
`x= x<<3;`	112
`x=x<<2;`	192
`x= x>>1;`	96
`x= x>>2;`	24

128 | 9

```
10000000   [128 in binary]

00000011   [3 in binary]

|          [bitwise OR]
________________________
10000011
________________________
```

127^120

```
01111111 [127 in binary]

01111000 [120 in binary]

^        [bitwise XOR]
________________________
00000111
________________________
```

C / C++ categorize statements into:

- **Selection [***if and switch***]**
- **Iteration [***while, for, do-while***]**
- **Jump [***break, continue, goto and return***]**
- **Label [** *case and default***]**
- **Expression [** *statements composed of a valid expression***]**
- **Block [***blocks of code. Each block begins with { and ends with } and referred to as compound statements***]**

Suppressing input:

```
scanf("%d %*d %d", &a, &b, &c);
```

Input

a=15, b=50, c=55

Here 15 is stored in a, 50 skipped and 55 is stored in the b, since no data is available for c so it has garbage value. The character * is called the suppression character.

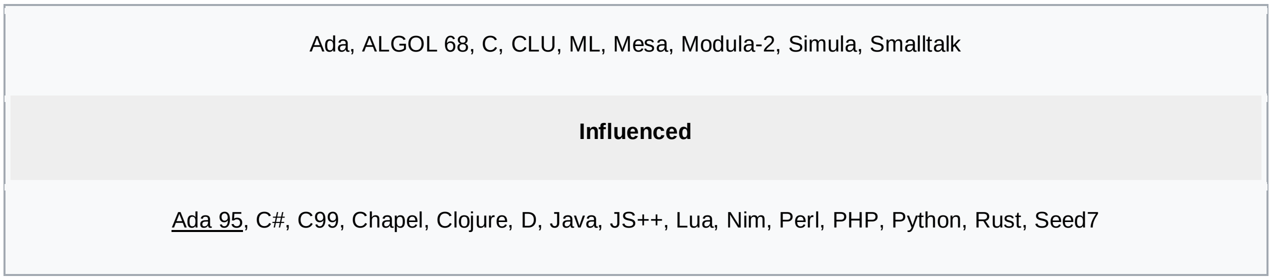
The infinite loop

```
for( ; ; )

printf("This Loop will run forever.\n";)
```

```
for( ; ; ) {
ch = getchar(); ──────────▶ get a character
if(ch=='B')
break; → Exit the loop
}
printf("you entered B";)

// This loop will run until the user types a Letter B at the keyword.
```

C++ Programming

Paradigms	Multi-paradigm: procedural, functional, object-oriented, generic
Family	C
Designed by	Bjarne Stroustrup
Developer	ISO/IEC JTC1 (Joint Technical Committee 1) / SC22 (Subcommittee 22) / WG21 (Working Group 21)
First appeared	1985; 35 years ago
Stable release	C++17 (ISO/IEC 14882:2017) / 1 December 2017; 2 years ago
Preview release	C++20
Typing discipline	Static, nominative, partially inferred
OS	Most major
Filename extensions	.C, .cc, .cpp, .cxx, .c++, .h, .hh, .hpp, .hxx, .h++
Website	isocpp.org
Major implementations	
GCC, LLVM Clang, Microsoft Visual C++, Embarcadero C++Builder, Intel C++ Compiler, IBM XL C++, EDG	
Influenced by	



Ada, ALGOL 68, C, CLU, ML, Mesa, Modula-2, Simula, Smalltalk

Influenced

Ada 95, C#, C99, Chapel, Clojure, D, Java, JS++, Lua, Nim, Perl, PHP, Python, Rust, Seed7

An Object-oriented (**Programming methodology** that views a computer program as a combination of variables, functions, and data structures called *objects*) high level cross-platform general-purpose case-sensitive language developed by a Danish computer scientist **Bjarne Stroustrup** (at Bell Labs circa 1980) as an **extension of the C language** for a specific purpose [C was developed for programming Operating Systems]. Initially named **C with classes** which later named C ++ in 1983. As a **successor of C language**, C++ has been certified as a **99.9 percent pure standard** and possesses exceptional performance, efficiency and flexibility of use compared to C language. It well deserves the widely acknowledged nickname "***Swiss Pocket Knife of Languages***."

Advantages: C++ is extremely fast and has the power and extensibility to write large-scale programs and runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX. Most famous software has their backbone in C++ and many programming languages depend on C++'s performance and reliability in their implementation [Examples include: JVM, JavaScript interpreters and Web frameworks]. C++ [*a superset of C*] is said to use static typing when type checking is performed during compile-time as opposed to run-time.

Limitation: Some tasks can be implemented in C++, though not very quickly. For example: designing GUI screens for applications. Other programming languages like VB, Python have GUI design elements built into them. Therefore, they are better suited for GUI type of task.

Uses: Used in the development of new programming languages [*C#, Java, JavaScript, Perl, PHP, Python and Verilog*], Apple Macintosh, PC running Windows, operating systems and Adobe Systems (like Photoshop, Acrobat etc), softwares for MRI machines, high-end CAD/CAM systems. C++ fully supports most important features of object-oriented programming including the four pillars of **object-oriented development**:

- Encapsulation
- Data hiding
- Inheritance
- Polymorphism

Inheritance: The ability of a class (sub class) to derive properties and characteristics from another class (super class) is called Inheritance. Inheritance is one of the most important features of Object Oriented Programming.

- **Sub Class:** The class that inherits properties from another class is called Sub class or Derived Class.
- **Super Class:** The class whose properties are inherited by sub class is called Base Class or Super class.

```
// My First C++ Program → Single line Comment

#include<iostream> →  preprocessor command

int main() → main function. Execution of C++ program begins from main()

{ →  Beginning of the main function

std::cout<<"Hello, world!"; →  output statement

return 0; → terminates the execution of the main function

} →  End of the main function

/* you can still use C style comment */
```

Process of C++ program execution:

Every 'C++' program follows a basic structure. A C++ program:

```
#include<iostream> //Pre-processor directive
int main() //main function declaration
{
std::cout<<"Hello, world!"; // output the string on a display screen
return 0; //terminating function
}
```

is written using **Text Editor** , such as [**Notepad** (in case of Windows Operating System), *vim* or vi (in case of Linux Operating System)] and saved with **[.cpp] Extension** – *for example, hello.cpp*. File Saved with [.cpp] extension is called **Source Program** or Source Code. C++ Source code with **[.cpp] Extension** is sent to preprocessor first. The preprocessor generates an expanded source code:

```
The contents of <iostream> would be pasted at the location of #include<iostream>
int main()
{
std::cout<<"Hello, world!";
return 0;
}
```

Expanded source code is given as input to compiler where the expanded source program is compiled (i.e., the program is entirely read and translated to instructions the computer can understand i.e., machine understandable / readable language i.e., to **machine code sequence** of 0s and 1s). If the C++ compiler finds any error during compilation, it provides information about the error to the programmer.

The programmer has to review code and re-edit the program. After re-editing program, **Compiler** again check for any error. If program is error-free then it is sent to assembler [where the code is assembled and converted into object code. Now a simple**.obj file** is generated].

The object code is sent to linker (where the object code is linked with appropriate libraries). Then it is converted into a single executable code. A simple**.exe file** is generated.

The executable code is sent to loader (where the executable code is loaded into memory and then it is executed). After execution, output:

```
Hello, world!
```

is displayed on the console screen.

#include<iostream> → preprocessor statement

This statement begins with # symbol and is also called preprocessor directive. This statement directs the preprocessor to include a section of standard C++ code [header iostream] that supports C++ style input output operations [<iostream> is to C++ what stdio.h is to C, allows to perform standard input and output operations - such as writing the output of this program "Hello, world!" to the console screen]. There is no .h extension to the name iostream. This is because that <iostream> is one of the modern style headers defined by standard C++.

- **iostream** means input output screen (i→ input, o → output, stream → screen) and iostream comprises input output functions like cout, cin etc. - note: **cin** is a input function (**cin** means console input) and **cout** is a output function (cout means console output)
- and it is included into the C++ program by writing the statement **#include<iostream>**). The statement #include tell the compiler to include the contents of the file iostream before compilation.
- If a program is written without the statement

  ```
  #include<iostream>
  ```

  then the C++ compiler can't compile and a compilation error is displayed on the screen (because C++ compiler fails to recognize the functions such as **cin** and **cout**).

main()→ main function

The function named main is a special function in all C++ programs; it is the function called when the C++ program is executed. As the name itself indicates this is the main function of every C++ Program. Parentheses () indicate a function and the word **main** indicate the name of the function.

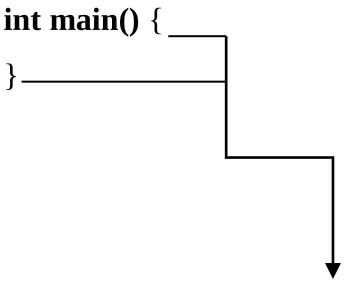
`main()` **implies:** main function. Execution of C++ program begins from `main()`. No C++ program is executed without `main()`. It is case sensitive language: only lower case letters (or small letters) must be used and should not be enclosed by a semicolon. There must be one and only one `main()` function in every C++ program.

We can represent the main function in various forms, such as:

- `main()`
- `int main()`
- `void main()`
- `main(void)`
- `void main(void)`
- `int main(void)`

As we know **C++ is Platform dependent language**. So the Operating system needs to know when the program execution ends. So when there is value returns from the main function, the Operating System get to know that the program execution is over. **`int main()` implies:** `main()` should return integer value.

- If the main function **returns 0 to the operating system**, then the program has completed execution successfully.
- If the main function **returns 1 to the operating system**, then the program has not completed execution successfully.



int main() {
}

body of the main function within which the sequence of instructions to the computer to perform specific operations in the form of statements [input-output statements, arithmetic statements, control statements and other statements] are **written and executed**.

The left curly brace

```
{
```

implies: the beginning of the main function

and the right curly brace

```
}
```

implies: the end of the main function

These braces can also be used to indicate the beginning and end of user-defined functions and compound statements.

`return 0;` → implies the exit status of execution of the program i.e., at this point, main function returns back the control of the computer to the operating system since the **execution is terminated** at this point and once a **return statement** i.e., **`return 0;`** is executed, no further instructions within the main function are executed.

For example:

```
#include<iostream>
int main()
{
std::cout<<"Hello, world!";
return 0;
std::cout<<"Hello, world!";
}
```

```
#include<iostream>
using namespace std;
int main()
{
cout<<"Hello, world!";
return 0;
cout<<"Hello, world!";
}
```

Output on the screen:

```
Hello, world!
```

`;` → implies **semicolon** or **statement terminator** [*a delimiter of the declaration*] → All C++ statements must end with a semicolon character. One of the most common syntax errors in C++ is forgetting to end a statement with a semicolon. A program is a well-defined set of instructions and each well-defined instruction (in the form of a statement) is ended by a semicolon (which is **C++ language punctuation** − like a period in English i.e., in an English paragraph each sentence is ended by a full stop which tells that one sentence ends and another begins, semicolon implies the end of one logical entity − that one instruction (or statement) ends and another begins).

`// My First C++ Program` → Comment

A good programmer who writes codes understood by a human is better than a programmer who generates codes understood only by the machine. So, it is highly recommended to insert comments. Comment is explanatory note on some instruction. Two slash signs indicate that the rest of the line is a comment inserted by the programmer but which has no effect on the behavior of the C++ program. Comment statement is not compiled and executed.

cout→ implies **the standard way of displaying output on the screen and** output function of the C++ language which makes provision to print the output:

```
Hello, world!
```

on the console screen. The statement

```
std::cout<<"Hello, world!";
```

has three parts:

- First, std::cout which identifies the standard character output device (usually, this is the computer screen).
  - std → standard
  - :: → scope resolution operator
  - cout → console output

  std::cout **basically means**: look in standard library and get cout function.

- Second, the insertion operator (<<), which indicates that what follows is inserted into std::cout. The symbol << implies: output the text:

  Hello, world!

  on the console screen using the **cout** function.

- Finally, a sentence enclosed by the double quotation marks ("Hello world!"), is the content inserted into the standard output.

If the statement:

```
using std::cout;
```

is added below the statement:

```
#include<iostream>
```

then the **program** takes the form:

```
#include<iostream>
using std::cout;
int main()
{
cout<<"Hello, crazy world!";
return 0;
}
```

i.e., no need to include `std::` in the statement:

```
std::cout<<"Hello, crazy world!";
```

C++ does not have strict rules on indentation:

```
int main() { std::cout << " Hello World!\n"; } → No ERROR
```

C++ Keywords:

auto	break	case	char	const
double	else	enum	extern	float
int	long	register	return	short
struct	switch	typedef	union	unsigned

continue	default	do
for	goto	if
signed	sizeof	static
void	volatile	while

A list of 30 Keywords in C++ Language which are not available in C language:

asm	dynamic_cast	namespace
explicit	new	static_cast
operator	template	friend
this	inline	public
delete	mutable	protected
typeid	typename	using

reinterpret_cast	bool
false	catch
private	class
throw	const_cast
true	try
virtual	wchar_t

```
int main()
{
int i, j;
i=10;
j=i*2;
return j;
} → NO ERROR

int main()
{
int i;
i=10;
int j;
j=i*2;
return j;
} → ERROR
```

C++ data types

Type	Keyword
Boolean	bool
Character	char
Integer	int
Floating point	float
Double floating point	double
Valueless	void
Wide character	wchar_t

```
#include <iostream>
using namespace std;

int main() {
   cout << "Size of char: " << sizeof(char) << endl;
   cout << "Size of int: " << sizeof(int) << endl;
   cout << "Size of short int: " << sizeof(short int) << endl;
   cout << "Size of long int: " << sizeof(long int) << endl;
   cout << "Size of float: " << sizeof(float) << endl;
   cout << "Size of double: " << sizeof(double) << endl;
   cout << "Size of wchar_t: " << sizeof(wchar_t) << endl;

   return 0;
}
```

The output on the screen:

```
Size of char: 1
Size of int: 4
Size of short int: 2
Size of long int: 4
Size of float: 4
Size of double: 8
Size of wchar_t: 4
```

- **Program 1.1**

  C++ program to print the word "hello Bill Gates" on screen

```
#include<iostream>
using std::cout;
int main()
{
cout<<"hello Bill Gates";
return 0;
}
```

The output on the screen:

```
hello Bill Gates
```

- **Program 1.2**

  C++ program to print

```
  *
*****
*****
*****
*****
```

on screen

```
#include<iostream>
using std::cout;
int main()
{
cout<<"\n    *     ";
cout<<"\n  *****   ";
cout<<"\n  *****   ";
cout<<"\n  *****   ";
cout<<"\n  *****   ";
return 0;
}
```

The output on the screen:

```
                                           *
                                         *****
                                         *****
                                         *****
                                         *****
```

If **new line** `\n` is not included in the above program then the output on the screen is:

```
*********************
```

- `endl` can be used instead of **`\n`**:

```
#include<iostream>
using std::cout;
int main()
{
cout<<"        *     "<< endl;
cout<<"      *****   "<< endl;
cout<<"      *****   "<< endl;
cout<<"       *****   "<< endl;
cout<<"       *****   "<< endl;
return 0;
}
```

The output on the screen:

```
// Error ⟶ endl was not declared in this scope //
```

If the above program is rewritten:

```
#include<iostream>
using std::cout;
using std::endl;
int main()
{
cout<<"       *     "<< endl;
cout<<"     *****   "<< endl;
cout<<"     *****   "<< endl;
cout<<"     *****   "<< endl;
cout<<"     *****   "<< endl;
return 0;
}
```

The output on the screen:

```
                                           *
                                         *****
                                         *****
                                         *****
                                         *****
```

The single statement:

```
using namespace std;
```

can be used instead of the statements:

```
using std::cout;
using std::endl;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
cout<<"       *     "<< endl;
cout<<"     *****   "<< endl;
cout<<"     *****   "<< endl;
cout<<"     *****   "<< endl;
cout<<"     *****   "<< endl;
return 0;
}
```

The output on the screen:

```
  *
*****
*****
*****
*****
```

C++'s Predefined Streams

Stream	Meaning	Default device
cin	Standard input	Keyboard
cout	Standard output	Screen
cerr	Standard error output	Screen
clog	Buffered version of cerr	Screen

Streams cin, cout and cerr correspond to C's stdin, stdout and stderr.

Token	Token type
E	VARIABLE
+	DELIMITER
101	NUMBER
–	DELIMITER
(	DELIMITER
K	VARIABLE
*	DELIMITER
1	NUMBER
null	null
)	DELIMITER
/	DELIMITER

C++ Manipulators

- **boolalpha:** Alphanumerical bool values (function)
- **showbase:** Show numerical base prefixes (function)

- **showpoint:** Show decimal point (function)
- **showpos:** Show positive signs (function)
- **skipws:** Skip whitespaces (function)
- **unitbuf:** Flush buffer after insertions (function)
- **uppercase:** Generate upper-case letters (function)
- **noboolalpha:** No alphanumerical bool values (function)
- **noshowbase:** Do not show numerical base prefixes (function)
- **noshowpoint:** Do not show decimal point (function)
- **noshowpos:** Do not show positive signs (function)
- **noskipws:** Do not skip whitespaces (function)
- **nounitbuf:** Do not force flushes after insertions (function)
- **nouppercase:** Do not generate upper case letters (function)
- **dec:** Use decimal base (function)
- **hex:** Use hexadecimal base (function)
- **oct:** Use octal base (function)
- **fixed:** Use fixed floating-point notation (function)
- **scientific:** Use scientific floating-point notation (function)
- **internal:** Adjust field by inserting characters at an internal position (function)
- **left:** Adjust output to the left (function)
- **right:** Adjust output to the right (function)
- **ws:** Extract whitespaces (function)
- **endl:** Insert newline and flush (function)
- **ends:** Insert null character (function)
- **flush:** Flush stream buffer (function)
- **setiosflags:** Set format flags (function)
- **resetiosflags:** Reset format flags (function)

- **setbase:** Set basefield flag (function)
- **setfill:** Set fill character (function)
- **setprecision:** Set decimal precision (function)
- **setw:** Set field width (function)

```
#include <iostream>
int main () {
  int n = -77;
  std::cout.width(6); std::cout << std::internal << n << '\n';
  std::cout.width(6); std::cout << std::left << n << '\n';
  std::cout.width(6); std::cout << std::right << n << '\n';
  return 0;
}
```

The output on the screen:

```
-   77
-77
   -77
```

- **Write a program to print the following outputs:**

(a)

*

```
****

*******

****

*
```

(b)

```
***************

* *

* Hello World! *

* *

***************
```

(c)

```
Braces come in pairs!

Comments come in pairs!

All statements end with a semicolon!

Spaces are optional!

Must have a main function!

C++ is done mostly in lowercase. It's a case-sensitive language
```

Answers:

(a)

```
#include<iostream>
using namespace std;
int main()
{
cout<<"\n           *     ";
cout<<"\n         ****    ";
cout<<"\n        *******  ";
cout<<"\n         ****    ";
cout<<"\n           *     ";
return 0;
}
```

(b)

```
#include<iostream>
using namespace std;
int main()
{
cout<<"\n           ****************         ";
cout<<"\n                 * *                ";
cout<<"\n           * Hello World! *         ";
cout<<"\n                 * *                ";
cout<<"\n           ****************         ";
return 0;
}
```

(c)

```
#include<iostream>
using namespace std;
int main()
{
cout<<"\n Braces come in pairs!";
cout<<"\n Comments come in pairs!";
cout<<"\n All statements end with a semicolon!";
cout<<"\n Spaces are optional!";
cout<<"\n Must have a main function!";
cout<<"\n C++ is done mostly in lowercase. It's a case-sensitive language";
return 0;
}
```

An old-style C++ program:

```
#include<iostream.h>
int main()
{
return 0;
}

// It includes the file iostream.h, not the header <iostream>. No namespace statement
is present.
```

A modern-style C++ program that uses the new style headers and a namespace:

```
#include<iostream>
Using namespace std;
int main()
{
return 0;
}

// It includes C++ style header and specifies a namespace.
```

Data Type	Initializer
int	0
char	'\0'
float	0
double	0
pointer	NULL

- **Program 1.3**

  C++ program to find the area of a circle

```
#include<iostream>
using namespace std;
main()
{
int r, area;
r = 2;
area = 3.14 * r * r;
cout<<"The area of the circle = "<< area;
return 0;
}
```

The output on the screen:

```
The area of the circle = 12
```

If the statement:

```
float r, area;
```

is used instead of

```
int r, area;
```

i.e.,

```
#include<iostream>
int main()
{
float r, area;
r = 2;
area = 3.14 * r * r;
cout<<"The area of the circle = "<< area;
return 0;
}
```

Then the **output on the screen**:

```
The area of the circle = 12.56
```

If you want to supply the *value for* r through the **key board**, then the statement

```
float r = 2;
```

should be replaced by the statements

```
cout<<"Enter any number:";
cin>>r;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
float r, area;
cout<<"Enter any number:";
cin>>r;
area = 3.14 * r * r;
cout<<"The area of the circle = "<< area;
return 0;
}
```

The output on the screen:

```
Enter any number:
If you the number 2
The area of the circle = 12.56

will be outputted on the screen.
```

- **`cout`** is an output function and **`cin`** is an input function.

The statement:

```
cout<<"Enter any number:";
```

make provision to print the text

`Enter any number:`

on the screen.

- `cin>> r;` is to C++ what `scanf("%d", &r);` is to C

If you write the statement:

```
area = 3.14 * r ^ 2;
```

instead of

```
area = 3.14 * r * r;
```

Then ***compilation error*** will be displayed on the console screen because like in *C Language* – there is no operator for performing exponentiation operation – so the statement

```
area = 4 * 3.14 * r ^ 2;
```

is invalid.

- cout and cin are not part of **C++ language** but they are part of **iostream file.** Hence the statement #include<iostream> should be included in the C++ program otherwise **cout** and **cin** will not work and the **compilation error** will be displayed on the console screen.
- **Right shift operator** >> denote stream extraction operator (extract data entered through the keyboard).
- **Left shift operator** << denote stream insertion operator (insert data into an output screen)
- << and >> are termed **overloaded operators** and the file iostream defines these operators.

As told earlier: when you enter an integer for x through the **keyboard**, this integer will be stored in the computer memory. If you yearn to know the ***storage size*** of the integer in **computer memory** (i.e., space occupied by the entered integer in the computer memory), you need to appeal to the following program:

```
#include<iostream>
using namespace std;
```

```
int main()
{
int x;
x=10;
cout<<"size of r =  "<< sizeof(r);
return 0;
}
```

The output on the screen:

```
size of x = 4
```

i.e., integer entered for r i.e., 10 has occupied a space of 4 bytes in the computer memory.

C++ Type Qualifiers

- **const:** Objects of type const cannot be changed by your program during execution.
- **volatile:** The modifier volatile tells the compiler that a variable's value may be changed in ways not explicitly specified by the program.
- **restrict:** A pointer qualified by restrict is initially the only means by which the object it points to can be accessed. Only C99 adds a new type qualifier called restrict.

- **Write a program to print the circumference of the circle (given r = 2.5)**

Answer:

```
#include<iostream>
using namespace std;
int main()
{
float r, area;
r = 2.5;
circumference = 2* 3.14 * r;
cout<<"The circumference of the circle =  "<< circumference;
return 0;
}
```

- **Write a program to print the area of the rectangle (given l = 2.5 and b = 3)**

Answer:

```
#include<iostream>
using namespace std;
int main()
{
float l, b, area;
l = 2.5;
b = 3;
area = 1*b;
cout<<"The area of the rectangle =  "<< area;
return 0;
}
```

- **Program 4.6**

  C++ program to find the sum of two numbers

```
#include<iostream>
using namespace std;
int main()
{
int a, b, sum;
a=1;
b=2;
sum = a + b;
cout<<"the sum of a and b = "<< sum;
return 0;
}
```

The output on the screen:

```
the sum of a and b = 3
```

If you assign the **floating point values** 1.5 and 2.6 *for* a and b, then the statement:

```
int a, b, sum;
```

should be replaced by the statement

```
float a, b, sum;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
float a, b, sum;
a=1.5;
b=2.6;
sum = a + b;
cout<<"the sum of a and b = "<< sum;
return 0;
}
```

The output on the screen:

```
the sum of a and b = 4.1
```

The statement:

```
cout<<"the sum of a and b = "<< sum;
```

make provision to print the output:

```
the sum of a and b = 4.1
```

on the console screen. And if the **statement**:

```
cout<<"the sum of a and b = "<< sum;
```

is omitted from the **C ++ program**, then the program will be successfully executed but there will be *no display of the output* on the console screen.

If you want to supply the values *for* a and b through the key board, then the statements:

```
a=1.5;
b=2.6;
```

should be replaced by the statements:

```
cout<<"Enter any two numbers:";
cin>>a;
cin>>b;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
float a, b, sum;
cout<<"Enter any two numbers:";
cin>>a;
cin>>b;
sum = a+ b;
cout<<"the sum of a and b = "<< sum;
return 0;
}
```

The output on the screen:

```
Enter any two numbers:
If you enter two numbers 2.9 & 3.6
the sum of a and b = 6.5
will be outputted on the screen.
```

The statement:

```
cout<<"Enter any two numbers:";
```

make provision to print

```
Enter any two numbers:
```

on the screen and the statements:

```
cin>>a;
cin>>b;
```

make provision to read the two numbers 2.9 and 3.6 entered through the **keyboard** and store them in the computer memory.

If the statements:

```
cout<<"Enter any two numbers:";
cin>>a;
cin>>b;
```

are replaced by the statements:

```
cout<<"Enter any number:";
cin>>a;
cout<<"Enter any number:";
cin>>b;
```

Then the **output on the screen** is:

```
Enter any number:
If you enter the number 2.9
Enter any number:
If you enter the number 3.6
the sum of a and b = 6.5
will be outputted on the screen.
```

If the statement:

```
cout<<"the sum of a and b = "<< sum;
```

is replaced by the statement:

```
cout<< a <<" + "<< b <<" = "<< sum;
```

Then the **output**:

2.9 + 3.6 = 6.5

will be displayed on the console screen.

- **What will be the output of the following program:**

```
#include<iostream>
using namespace std;
int a = 5;
int main()
{
int a =2;
cout<< a;
return 0;
}
```

Answer:

2

- 2 is a **local variable** (variable declared within the body of the **main function**)

The statement:

```
int a = 2;
```

imply: **local variable** declaration.

- 5 is a **global variable** (variable declared outside the body of the main function)

The statement:

```
int a = 5;
```

imply: global variable declaration.

If the statement:

```
cout<< a;
```

is replaced by the statement:

```
cout<< :: a; (where :: denote scope resolution operator)
```

i.e.,

```
#include<iostream>
using namespace std;
int a = 5;
int main()
{
int a =2;
cout<< ::a;
return 0;
}
```

Then the **output on the screen** is:

5

i.e., **global variable** will be outputted on the screen.

If the same program is written in **C language**

i.e.,

```
#include<stdio.h>
int a = 5;
int main()
{
int a =2;
print("%d", ::a);
return 0;
}
```

Then the **compilation error** will be outputted on the screen because ***scope resolution operator*** is not defined in the C language (i.e., C does not hold **scope resolution operator**).

while loop Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
for loop Execute a sequence of statements multiple times and abbreviates the code that manages the loop variable.
do...while loop Like a 'while' statement, except that it tests the condition at the end of the loop body.
nested loops You can use one or more loop inside any another 'while', 'for' or 'do..while' loop.

- **Whether the following program will be successfully outputted or not:**

```
#include<iostream>
using namespace std;
int main()
{
int a, b, c;
a=3;
b=2;
c= a+b;
cout<<" sum of two numbers = 6"<< c;
return 0;
}
```

Answer:

Yes, the output on the screen is:

```
sum of two numbers = 65
```

- **Program 4.7**

  C ++ program to convert the temperature in Celsius to Fahrenheit

```
#include<iostream>
using namespace std;
int main()
{
float C, F;
C=38.5;
F = 9*C/5 +32;
cout<<"temperature in Fahrenheit= "<< F;
return 0;
}
```

The output on the screen:

```
temperature in Fahrenheit = 101.3
```

If you want to supply a value **16 digits** after decimal point i.e., 36.5555555555555555 for C, then the statement:

```
double C, F;
```

```
should be used instead of the statement:

float C, F;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
double C, F;
C=38.5555555555555555;
F = 9*C/5 +32;
cout<<"temperature in Fahrenheit= "<< F;
return 0;
}
```

If you want to supply the value for C through the key board, then the statement:

`C=38.5;`

should be replaced by the statements:

```
cout<<"Enter any number:";
cin>>C;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
float C, F;
cout<<"Enter any number:";
cin>>C;
F = 9*C/5 +32;
cout<<"temperature in Fahrenheit= "<< F;
return 0;
}
```

The output on the screen:

```
Enter any number:

If you enter the number 23.6
temperature in Fahrenheit = 74.48

will be outputted on the screen.
```

break statement Terminates the **loop** or **switch** statement and transfers execution to the statement immediately following the loop or switch.
continue statement Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.
goto statement Transfers control to the labeled statement. Though it is not advised to use goto statement in your program.

- **Program 4.8**

C++ program to find the product of two numbers

```
#include<iostream>
using namespace std;
int main()
```

```
{
int a, b, product;
a=1;
b=2;
product = a * b;
cout<<"the product of a and b = "<< product;
return 0;
}
```

The output on the screen:

```
the product of a and b = 2
```

If you want to insert a **10 digit number** *for* a and b i.e.,

```
a=1000000000
b=3000000000
```

, then the statement:

```
int a, b, product;
```

should be replaced by the statement:

```
long int a, b, product;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
long int a, b, product;
a=1;
b=2;
product = a * b;
cout<<"the product of a and b = "<< product;
return 0;
}
```

The output on the screen:

```
the product of a and b = 3000000000000000000
```

If you want to supply the **integer values** ***for*** a and b through the key board, then the statements:

a=1;

b=2; should be replaced by the statements:

```
cout<<"Enter any two numbers:";
cin >> a;
cin >> b;
```

i.e.,

```
#include<iostream>
using namespace std;
int main ()
{
int a, b, product;
cout<<"Enter any two numbers:";
cin>>a;
cin>>b;
product = a* b;
cout<<"the product of a and b = "<< product;
return 0;
}
```

The output on the screen:

```
Enter any two numbers:

If you enter two numbers 2 & 3

the product of a and b = 6

will be outputted on the screen.
```

The numeric expressions are composed of the following terms:

- Numbers
- Operators +, −, /, *,^,%, =
- Parentheses
- Variables

- **Program 4.9**

  C++ program to find the square of a number

```
#include<iostream>
using namespace std;
int main()
{
int a, b;
a=2;
b = a * a;
cout<<"the square of a = "<< b;
return 0;
}
```

The output on the screen:

```
the square of a = 4
```

If you want to supply the integer value for a through the **key board**, then the statement:

```
a=2;
```

should be replaced by the statements:

```
cout<<"Enter any number:";

cin>>a;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
int a, b;
cout<<"Enter any number:";
cin>>a;
b = a * a;
cout<<"the square of a = "<< b;
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter a number 3
the square of a = 9
will be outputted on the screen.
```

- **Write a program to print the cube of a number**

Answer:

```
#include<iostream>
using namespace std;
int main()
{
int a, b;
cout<<"Enter any number:";
cin>>a;
b = a * a*a;
cout<<"the cube of a = "<< b;
return 0;
}
```

- **Write a program to print the force applied to the mass m.**

Answer:

```
#include<iostream>
using namespace std;
int main()
{
int m, a, F;
cout<<"Enter the mass:";
cin>>m;
cout<<"Enter acceleration:";
cin>>a;
F = m * a;
cout<<"the force applied to the mass =  "<< F;
return 0;
}
```

if statement An 'if' statement consists of a boolean expression followed by one or more statements.
if...else statement An 'if' statement can be followed by an optional 'else' statement, which executes when the boolean expression is false.
switch statement A 'switch' statement allows a variable to be tested for equality against a list of values.
nested if statements You can use one 'if' or 'else if' statement inside another 'if' or ‘else if’ statement(s).
nested switch statements You can use one 'switch' statement inside another 'switch' statement(s).

- **Program 5.0**

C ++ program to find the greatest of two numbers using *if* - *else* statement

The syntax of *if* - *else* statement is:

```
if (this condition is true)
{
print this statement;
}
else
{
print this statement;
}
```

```
#include<iostream>
using namespace std;
int main()
{
int a, b;
a = 2;
b = 3;
if(a>b)
{
cout<<"a is greater than b";
}
else
{
cout<<"b is greater than a";
}
return 0;
}
```

The output on the screen:

```
b is greater than a
```

- **Program 5.1**

  C++ program to find the greatest of three numbers using *else-if* statement

  The syntax of *else-if* statement:

```
if (this condition is true)

{

print this statement;

}

else if (this condition is true)
```

```
{
print this statement;
}
else
{
print this statement;
}
```

```
#include<iostream>
using namespace std;
int main()
{
int a, b, c;
cout<<"Enter any number:";
cin>>a;
cout<<"Enter any number:";
cin>>b;
cout<<"Enter any number:";
cin>>c;
if(a>b&&a>c)
{
cout<< a<<" is greater than"<< b<<" and "<<c;
}
else if (b>a&&b>c)
{
cout<< b<<" is greater than"<< a <<" and "<<c;
}
else
{
cout<< c<<" is greater than"<< b<<" and "<< a;
}

return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 3
Enter any number:
If you enter the number 4
4 is greater than 3 and 2 will be outputted on the screen.
```

- **What will be the output of the following program?**

```
#include <iostream>
int main()
{
int a, b;
a=2;
b=2;
if(a>b || a==b)
cout<<"a is greater than or equal to b";
else
cout<<"b is greater than a";
return 0;
}
```

Answer:

```
a is greater than or equal to b
```

- symbol || denote OR i.e., a>b || a == b denote **a** *is greater than or* **a is equal to b.**

- **Program 5.2**

  C ++ program to find the average of 10 numbers

```
#include<iostream>
using namespace std;
int main()
{
int N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, X;
cout<<"Enter any 10 numbers:";
cin>>N1;
cin>>N2;
cin>>N3;
cin>>N4;
cin>>N5;
cin>>N6;
cin>>N7;
cin>>N8;
cin>>N9;
cin>>N10;
X = (N1 + N2 + N3 + N4 + N5 + N6 + N7 + N8 + N9 + N10) /10;
cout<<"the average of 10 numbers = "<< X;
return 0;
}
```

The output on the screen:

```
Enter any 10 numbers:

If you enter ten numbers 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10

the average of 10 numbers = 5

will be outputted on the screen.
```

C++ equivalent *mathematical expression* is same as C equivalent *mathematical expression*

For example:

Mathematical expression	C equivalent expression	C++ equivalent expression
$\log_{10}x + bx$	log10 (x) + b * x	log10 (x) + b * x

- **Program 5.3**

C ++ program to find the square root of a number

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
int a, b;
cout<<"Enter any number:";
cin>> a;
b = sqrt (a);
cout<<"the square root of a number = "<<  b;
return 0;
}
```

The output on the screen:

```
Enter any number:

If you enter the number 16

the square root of a number = 4

will be outputted on the screen.
```

Since b = sqrt(a) is written:

the statement: #include<cmath> must be included in the above program because **cmath file** defines the mathematical functions like sqrt().

If the statement: #include<cmath> is not included in the above program:

```
#include<iostream>
using namespace std;
int main()
{
int a, b;
cout<<"Enter any number:";
cin>> a;
b = sqrt (a);
cout<<"the square root of a number = "<<  b;
return 0;
}
```

Then the ***compilation error*** will be displayed on the console screen.

Note:

```
#include<math.h> is used in C
whereas #include<cmath> is used in C ++
```

- **Write a program to print the cube root of a number:**

Answer:

```
#include<iostream>
#include<cmath>
```

```
using namespace std;
int main()
{
cout<<"the cube root of a number =  "<< cbrt (8);
return 0;
}
```

- **Program 5.4**

  C++ program to find the simple interest

```
#include<iostream>
using namespace std;
int main()
{
int P,T, R, SI;
P = 1000;
T = 2;
R = 3;
SI = P*T*R/100;
cout<<"the simple interest = "<< SI;
return 0;
}
```

The output on the screen:

```
the simple interest = 60
```

If you want to supply the **integer values** *for* P, T and R through the **key board**, then the statements:

```
P = 1000;
```

```
T = 2;
R = 3;
```

should be replaced by the statements:

```
cout<<"Enter principal amount:";
cin>>P;
cout<<"Enter time:";
cin>>T;
cout<<"Enter rate of interest:";
cin>>R;
```

i.e., the above program should take the form:

```
#include<iostream>
using namespace std;
int main()
{
int P,T, R, SI;
cout<<"Enter principal amount:";
cin>>P;
cout<<"Enter time:";
cin>>T;
cout<<"Enter rate of interest:";
cin>>R;
SI = P*T*R/100;
cout<<"the simple interest = "<<SI;
return 0;
}
```

The output on the screen:

```
Enter principal amount:
```

```
If you enter the principal amount 1000
Enter time:
If you enter the time 2
Enter rate of interest:
If you enter the rate of interest 3
the simple interest = 60
will be outputted on the screen.
```

- **Program 5.5**

  C++ program to find the senior citizen

```
#include<iostream>
using namespace std;
int main()
{
int age;
age=20;
if(age > = 60)
{
cout<<"senior citizen";
}
if(age<60)
{
cout<<"not a senior citizen";
}
return 0;
}
```

The output on the screen:

```
not a senior citizen
```

- (age > = 60) **means:** age greater than or equal to 60.

If you want to supply the value for age through the key board, then the statement:

```
age = 20;
```

should be replaced by the statements:

```
cout<<"Enter age:";
cin>>age;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
int age;
cout<<"Enter age:";
cin>>age;
if(age>60)
{
cout<<"senior citizen";
}
if(age<60)
{
cout<<"not a senior citizen";
}
return 0;
}
```

The output on the screen:

```
Enter age:

If you enter the age 60

senior citizen
```

```
will be outputted on the screen.
```

Suppose if you enter the age 31

```
not a senior citizen
```

will be outputted on the screen.

- **Program 5.6**

  C ++ program to get marks for 3 subjects and declare the result.

  If the marks >= 35 in all the subjects the student passes else fails.

```
#include<iostream>
using namespace std;
int main()
{
int M1, M2, M3;
M1 = 38;
M2= 45;
M3 = 67;
if(M1 >= 35 && M2>= 35 && M3>= 35)
{
cout<<"candidate is passed";
}
else
{
cout<<"candidate is failed";
}
return 0;
}
```

The output on the screen:

```
candidate is passed
```

If you want to supply the integer values *for marks* M1, M2 and M3 through the **key board**, then the statements:

```
M1 = 38;
M2= 45;
M3 = 67;
```

should be replaced by the statements:

```
cout<<"Enter any three marks:";
cin>> M1;
cin>> M2;
cin>> M3;
```

i.e.,

```
#include<iostream>
int main()
{
int M1, M2, M3;
cout<<"Enter any three marks:";
cin>> M1;
cin>> M2;
cin>> M3;
if(M1 >= 35 && M2>= 35 && M3>= 35)
{
cout<<"candidate is passed";
}
else
{
```

```
cout<<"candidate is failed";
}
return 0;
}
```

The output on the screen:

```
Enter any three numbers:

If you enter three numbers 26, 28, 39

candidate is failed

will be outputted on the screen.
```

- **Program 5.7**

  C ++ program to find profit or loss

```
#include<iostream>
using namespace std;
int main()
{
int CP, SP, loss, profit;
cout<<"Enter cost price:";
cin >> CP;
cout<<"Enter selling price:";
cin>>SP;
if ( SP > CP )
{
cout<<"profit= "<< SP-CP;
}
else
{
cout<<"loss = "<< CP-SP;
}
return 0;
}
```

The output on the screen:

```
Enter cost price:
If you enter the cost price 25
Enter selling price:
If you enter the selling price 26
profit = 1
will be outputted on the screen.
```

- **Program 5.8**

  C++ program to convert inches into centimeter

```
#include<iostream>
using namespace std;
int main()
{
float I, C;
I=3.5;
C = 2.54*I;
cout<<"length in centimeters = "<< C;
return 0;
}
```

The output on the screen:

```
length in centimeters = 8.89
```

- **Note:** float is used instead of int because I = 3.5 if int is used instead of float then the result will not be clearly outputted i.e., instead of 8.89 the computer displays only 8.

If you want to supply the value *for* I through the **key board**, then the above program should take the form:

```
#include<iostream>
using namespace std;
int main()
{
float I, C;
cout<<"Enter the length in inches:";
cin >> I;
C = 2.54*I;
cout<<"length in centimeters= "<< C;
return 0;
}
```

The output on the screen:

```
Enter the length in inches:
```

If you enter the value for I i.e., 25.5

```
length in centimeters = 64.9 will be outputted on the screen.
```

- **Program 5.9**

  C++ program to find the incremented and decremented values of two numbers

```
#include<iostream>
using namespace std;
int main()
```

```
{
int a, b, c, d, e, f;
a = 10;
b=12;
c=a+1;
d=b+1;
e=a-1;
f=b-1;
cout<<"the incremented value of a = "<< c;
cout<<"the incremented value of b = "<< d;
cout<<"the decremented value of a = "<< e;
cout<<"the decremented value of b = "<< f;
return 0;
}
```

The output on the screen:

```
the incremented value of a = 11 the incremented value of b = 13 the decremented value
of a = 9 the decremented value of b = 11
```

If the statements:

```
cout<<"the incremented value of a = "<< c;
cout<<"the incremented value of b = "<< d;
cout<<"the decremented value of a = "<< e;
cout<<"the decremented value of b = "<< f;
```

are replaced by the statements:

```
cout<<"\n the incremented value of a = "<< c;
cout<<"\n the incremented value of b = "<< d;
cout<<"\n the decremented value of a = "<< e;
cout<<"\n the decremented value of b = "<< f;
```

Then the **output on the screen** is:

```
the incremented value of a = 11
the incremented value of b = 13
the decremented value of a = 9
the decremented value of b = 11
```

If the statements:

```
cout<<"the incremented value of a = "<< c;
cout<<"the incremented value of b = "<< d;
cout<<"the decremented value of a = "<< e;
cout<<"the decremented value of b = "<< f;
```

are replaced by the statements:

```
cout<<"the incremented value of a = "<< c << endl;
cout<<"the incremented value of b = "<< d << endl;
cout<<"the decremented value of a = "<< e << endl;
cout<<"the decremented value of b = "<< f << endl;
```

Then the **output on the screen**:

```
the incremented value of a = 11
the incremented value of b = 13
the decremented value of a = 9
the decremented value of b = 11
```

If you want to supply the values *for* a ***and*** b through the **key board**, then the above program should take the form:

```
#include<iostream>
```

```
using namespace std;
int main()
{
int a, b, c, d, e, f;
cout<<"Enter any number:";
cin>> a;
cout<<"Enter any number:";
cin>> b;
c=a+1;
d=b+1;
e=a-1;
f=b-1;
cout<<"\n the incremented value of a = "<< c;
cout<<"\n the incremented value of b = "<< d;
cout<<"\n the decremented value of a = "<< e;
cout<<"\n the decremented value of b = "<< f;
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 3

the incremented value of a = 3
the incremented value of b = 4
the decremented value of a = 1
the decremented value of b = 2

will be outputted on the screen.
```

- **What will be the output of the following program:**

```
#include<iostream>
using namespace std;
int main()
{
float T1, T2, A;
cout<<"Enter any number:";
cin >>T1;
cout<<"Enter any number:";
cin >>T2;
A = (T1 + T2) / 2;
cout<<"the average temperature of the day = "<< A;
return 0;
}
```

Answer:

```
Enter any number:
If you enter the number:
2
Enter any number:
If you enter the number:
3
the average temperature of the day = 2.5
will be displayed on the console screen.
```

- **Program 6.0**

  The percentage marks are entered and the grades are allotted as follows:

  percentage >= 60 First Class

  percentage >=50 and per <= 60 Second Class

  percentage >= 40 and per <= 50 Pass Class

  percentage < 40 Fail

Write a C++ program for the above:

```
#include<iostream>
using namespace std;
main()
{
int P;
cout<<"Enter the percentage:";
cin>>P;
if(P >= 60)
{
cout<<"first class";
}
if(P>=50&&P <60)
{
cout<<"second class";
}
if(P>=40&&P<=50 )
{
cout<<"pass class";
}
if(P<40)
{
cout<<"fail";
}
return 0;
}
```

The output on the screen:

```
Enter the percentage:

If you enter the percentage 35

fail

will be outputted on the screen.
```

- **Program 6.1**

C++ program to calculate the discounted price and the total price after discount

Given:

- If purchase value is greater than 1000, 10% discount
- If purchase value is greater than 5000, 20% discount
- If purchase value is greater than 10000, 30% discount

- **discounted price**

```
#include<iostream>
using namespace std;
int main()
{
double PV, dis;
cout<<"Enter purchased value:";
cin>>PV;
if(PV>1000)
{
cout<<"dis= "<< PV* 0.1;
}
else if(PV>5000)
{
cout<<"dis= "<< PV* 0.2;
}
else
{
cout<<"dis= "<< PV* 0.3;
}
return 0;
}
```

The output on the screen:

```
Enter purchased value:
If you enter the purchased value 6500
dis = 1300.000000
will be outputted on the screen.
```

- **total price**

```
#include<iostream>
using namespace std;
int main()
{
double PV, total;
cout<<"Enter purchased value:";
scanf("%lf", &PV;
if(PV<1000)
{
cout<<"total= "<< PV - PV* 0.1;
}
else if(PV<5000)
{
cout<<"total = "<< PV- PV* 0.2;
}
else
{
cout<<"total= "<< PV- PV* 0.3;
}
return 0;
}
```

The output on the screen:

```
Enter purchased value:
If you enter the purchased value 650
```

```
total = 585.000000
will be outputted on the screen.
```

- **Now, Combing both the programs (above), we can write:**

```
#include<iostream>
using namespace std;
int main()
{
double PV, dis, total;
cout<<"Enter purchased value:";
cin>>PV;
if(PV>1000)
{
cout<<"dis= "<< PV* 0.1;
cout<<"total= "<< PV - PV* 0.1;
}
else if(PV>5000)
{
cout<<"dis = "<< PV* 0.2;
cout<<"total= "<< PV - PV* 0.1;
}
else
{
cout<<"dis= "<< PV* 0.3;
cout<<"total= "<< PV - PV* 0.1;
}
return 0;
}
```

The output on the screen:

```
Enter purchased value:
```

```
If you enter the purchased value 850

dis = 85.000000

total = 765.000000

will be outputted on the screen.
```

- **Program 6.2**

  C++ program to print the first ten natural numbers using *for* loop statement

```
#include<iostream>
using namespace std;
int main()
{
int i;
for (i=1; i<=10; i++)
cout<<"value of i = "<< i;
return 0;
}
```

The output on the screen is:

```
value of i = 1 value of i = 2 value of i = 3  value of i = 4  value of i= 5  value of
i= 6 value of i = 7 value of i= 8  value of i= 9  value of i= 10
```

- **When *for loop* executes, the following occurs:**

```
i = 1
Is the condition (i<=10) is true?
Yes because i=1
The statement cout<<"value of i = "<< i; is executed to print the output:
```

```
value of i = 1
Now, the value of i is:
i = 1+1 = 2
Is the condition (i<=10) is true?
Yes because i=2
The statement cout<<"value of i = "<< i; is executed to print the output:
value of i = 2
Now, the value of i is:
i = 2+1 = 3
Is the condition (i<=10) is true?
Yes because i=3
The statement cout<<"value of i = "<< i; is executed to print the output:
value of i = 3
Now, the value of i is:
i = 3+1 = 4
Is the condition (i<=10) is true?
Yes because i=4
The statement cout<<"value of i = "<< i; is executed to print the output:
value of i = 4
Now, the value of i is:
i = 4+1 = 5
Is the condition (i<=10) is true?
Yes because i=5
The statement cout<<"value of i = "<< i; is executed to print the output:
value of i = 5
Now, the value of i is:
i = 5+1 = 6
Is the condition (i<=10) is true?
Yes because i=6
The statement cout<<"value of i = "<< i; is executed to print the output:
value of i = 6
Now, the value of i is:
i = 6+1 = 7
Is the condition (i<=10) is true?
Yes because i=7
The statement cout<<"value of i = "<< i; is executed to print the output:
value of i = 7
Now, the value of i is:
i = 7+1 = 8
Is the condition (i<=10) is true?
Yes because i=8
```

```
The statement cout<<"value of i = "<< i; is executed to print the output:
value of i = 8
Now, the value of i is:
i = 8+1 = 9
Is the condition (i<=10) is true?
Yes because i=9
The statement cout<<"value of i = "<< i; is executed to print the output:
value of i = 9
Now, the value of i is:
i = 9+1 = 10
Is the condition (i<=10) is true?
Yes because i=10
The statement cout<<"value of i = "<< i; is executed to print the output:
value of i = 10
and stop because the condition i<=10 is achieved.
```

If the statement:

```
cout<<"value of i = "<< i;
```

is replaced by the statement:

```
cout<<"\n value of i = "<< i;
```

Then the output on the screen is:

```
value of i = 1
value of i = 2
value of i = 3
value of i = 4
value of i = 5
value of i = 6
value of i = 7
value of i = 8
value of i = 9
value of i = 10
```

If the *for loop* statement:

```
for (i=2; i<=10; i++)
```

is written instead of the statement:

```
for (i=1; i<=10; i++)
```

then the output on the screen is:

```
value of i = 2 value of i  = 3 value of i= 4 value of i= 5 value of i= 6 value of i = 7
value of i= 8 value of i = 9 value of i= 10
```

If the *for loop* statement:

`for(i=1; i<10; i++)`

is written instead of the statement:

`for (i=1; i<=10; i++)`, then the output on the screen is:

```
value of i = 1 value of i  = 2 value of i= 3 value of i= 4 value of i= 5 value of i= 6
value of i = 7 value of i= 8 value of i = 9
```

- **Note:** the condition i<=10 tells to print till value of i=10 but the condition i<10 tells to print till value of i=9.

If the statement:

```
for(i=1; i=10; i++)
```

is written instead of the statement:

```
for(i=1; i<=10; i++)
```

then the output on the screen is:

```
value of i = 10  value of i = 10  value of i = 10  value of  i = 10  value of i= 10
value of i= 10 value of i = 10 value of i= 10  value of i = 10   value of i = 10
value of i = 10   value of i = 10  value of i = 10    value of i = 10   value of i = 10

continues ....
```

- **Note:**

If the statement:

```
cout<<"value of i = "<< i;
```

is replaced by the statement:

```
cout<<"\n "<< i;
```

Then the output on the screen is:

```
1

2

3

4

5

6

7

8

9

10
```

Multi-dimensional arrays C++ supports multidimensional arrays. The simplest form of the multidimensional array is the two-dimensional array.
Pointer to an array You can generate a pointer to the first element of an array by simply specifying the array name, without any index.
Passing arrays to functions You can pass to the function a pointer to an array by specifying the array's name without an index.
Return array from functions C++ allows a function to return an array.

- **What will be the output of the following program:**

```
#include<iostream>
using namespace std;
int main()
{
int i;
for (i =1; i<=5; i ++)
cout<<"\n Linux is not portable";
return 0;
}
```

Answer:

```
Linux is not portable
Linux is not portable
Linux is not portable
Linux is not portable
Linux is not portable
```

- **C++ program to print the first ten natural numbers using *while loop* statement**

  The syntax of *while loop* statement is:

```
while (this is the condition)

{

execute this statement;

}
```

```
#include<iostream>
using namespace std;
int main()
{
int i = 1;
while (i<=10)
{
cout<<"\n "<< i++;
}
return 0;
}
```

The output on the screen is:

```
1
2
3
4
5
6
7
8
9
10
```

(i<=10) is the condition and

The statement

```
cout<<"\n "<< i++;
```

is repeatedly executed as long as a given condition (i<=10) is true.

If the statement:

```
int i=1;
```

is replaced by the statement:

```
int i;
```

Then the **compilation error** will be displayed on the console screen because initialization is not defined [i.e., from where to start is not declared].

If the statement:

```
int i = 1;
```

is replaced by **`int i = 0;`**

Then the output on the screen is:

```
0
1
2
3
4
5
6
7
8
9
10
```

Similarly if the statement `int i = 0;` is replaced by the `int i = 7;`

Then the output on the screen is:

```
7
```

```
8

9

10
```

- **C++ program to print first 10 numbers using *do while loop* statement**

  The syntax of do while loop statement is:

  do

  {

  execute this statement;

  }

  while(this is the condition);

```
#include<iostream>
using namespace std;
int main()
{
int i =1;
do
{
cout<<" \n i= "<< i++;
} while (i<=10);
return 0;
}
```

The output on the screen is:

```
i=1
i=2
i=3
i=4
i=5
i=6
```

```
i=7
i=8
i=9
i=10
```

The statement:

`cout<<" \ni= "<< i++;`

is executed and then condition `(i<=10)` is checked. If condition `(i<=10)` is true then

The statement:

```
cout<<" \ni= "<< i++;
```

is executed again. This process repeats until the given **condition (i<=10)** becomes false.

- **Write a program to print**

  When in doubt use brute force

  100 times using for loop statement.

Answer:

```
#include<iostream>
using namespace std;
int main()
{
int i;
for(i=0; i<=99; i++)
cout<<"\n When in doubt use brute force";
return 0;
}
```

- **Program 6.3**

C++ program to print the characters from A to Z using *for loop*, *do while loop* and *while loop* statement.

- **C ++ program to print the characters from A to Z using *for loop* statement:**

```
#include<iostream>
using namespace std;
int main()
{
char a;
for( a='A'; a<='Z'; a++)
cout<<" \n"<< a;
return 0;
}
```

The output on the screen:

```
A
B
C
D
E
F
G
H
I
```

```
J
K
L
M
N
O
P
Q
R
S
T
W
X
Y
Z
```

If the statement: `for( a=A; a<=Z; a++)` is written instead of the statement `for( a='A'; a<='Z'; a++)` i.e., A is used instead of 'A' and Z is used instead of 'Z', then the **compilation error** will be displayed on the screen.

- **C ++ program to print the characters from A to Z using while loop statement:**

```
#include<iostream>
using namespace std;
int main()
{
char a = 'A';
while (a<='Z')
{
cout<<" \n"<< a++;
}
return 0;
}
```

- **C ++ program to print the characters from A to Z using do while loop statement:**

```
#include<iostream>
using namespace std;
int main()
{
char a = 'A';
do
{
cout<<" \n"<< a++;
} while (a<='Z');
return 0;
}
```

- **Program 6.4**

C++ program to print the given number is even or odd.

```
#include<iostream>
using namespace std;
int main()
{
int a;
cout<<"Enter any number:";
cin>>a;
if(a%2 = = 0)
{
cout<<"the number is even";
}
else
{
cout<<"the number is odd";
}
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 6
the number is even
will be outputted on the screen.
```

- ### Program 6.5

C++ program to print the remainder of two numbers

```
#include<iostream>
using namespace std;
int main()
{
int a, b, c;
cout<<"Enter any number:";
cin>>a;
cout<<"Enter any number:";
cin>>b;
c = a % b;
cout<<"the remainder of a and b = "<< c;
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 3
Enter any number:
If you enter the number 2
the remainder of a and b = 1

will be outputted on the screen.
```

If the statement:

```
cout<<"the remainder of a and b = "<< c;
```

is replaced by the statement:

```
cout <<" the remainder of "<<a <<"and"<< b <<"=  "<< c;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
int a, b, c;
cout<<"Enter any number:";
cin>>a;
cout<<"Enter any number:";
cin>>b;
c = a % b;
cout <<" the remainder of "<<a <<"and"<< b <<"=  "<< c;
return 0;
}
```

The output on the screen:

```
Enter any number:

If you enter the number 3

Enter any number:

If you enter the number 2

the remainder of 3 and 2 = 1

will be outputted on the screen.
```

strcpy(s1, s2); Copies string s2 into string s1.
strcat(s1, s2); Concatenates string s2 onto the end of string s1.
strlen(s1); Returns the length of string s1.
strcmp(s1, s2); Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
strchr(s1, ch); Returns a pointer to the first occurrence of character ch in string s1.
strstr(s1, s2); Returns a pointer to the first occurrence of string s2 in string s1.

- **Program 6.6**

  C++ program to check equivalence of two numbers

```
#include<iostream>
using namespace std;
int main()
{
int x, y;
cout<<"Enter any number:";
cin>>x;
cout<<"Enter any number:";
cin>>y;
if(x-y==0)
{
cout<<"the two numbers are equivalent";
}
else
{
cout<<"the number are not equivalent";
}
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2
Enter any number:
If you enter the number 2
the two numbers are equivalent
will be outputted on the screen.
```

- **Program 6.7**

  C ++ program to print whether the given number is positive or negative.

```
#include<iostream>
using namespace std;
int main()
{
int a;
a = -35;
if(a>0)
{
cout<<"number is positive";
}
else
{
cout<<" number entered is negative";
}
return 0;
}
```

The output on the screen:

```
number entered is negative
```

- **Program 6.8**

C++ program to print the sum of the first 10 numbers using *for loop* statement

```
#include<iostream>
using namespace std;
int main()
{
int i, sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
cout<<"sum of the first10 numbers = "<< sum;
return 0;
}
```

The output on the screen:

```
sum of the first 10 digits = 55
```

- **How the sum of the first 10 digits = 55 is outputted on the screen through the for Loop statement?**

```
i=1 (sum = 0 because the sum is initialized to 0 in the statement int i, sum = 0;)
Is i<=10 true?
Yes, do this
sum = sum + i = 0 +1 =1
Now,
i=2 (sum = 1)
Is i<=10 true?
Yes, do this
sum = sum + i = 1 +2 =3
Now,
i=3 (sum = 3)
Is i<=10 true?
Yes, do this
sum = sum + i = 3 +3 = 6
Now,
i=4 (sum = 6)
Is i<=10 true?
Yes, do this
sum = sum + i = 6 + 4= 10
Now,
i=5 (sum = 10)
Is i<=10 true?
Yes, do this
sum = sum + i = 10 + 5= 15
Now,
i=6 (sum = 15)
Is i<=10 true?
Yes, do this
sum = sum + i = 15 + 6 = 21
Now,
i=7 (sum = 21)
Is i<=10 true?
Yes, do this
sum = sum + i = 21 + 7 = 28
Now,
i=8 (sum = 28)
Is i<=10 true?
Yes, do this
sum = sum + i = 28 + 8 = 36
Now,
i=9 (sum = 36)
Is i<=10 true?
Yes, do this
sum = sum + i = 36 + 9 = 45
Now,
i=10 (sum = 45)
Is i<=10 true?
Yes, do this
sum = sum + i = 45 + 10 = 55
stops because the condition i<=10 is achieved
```

The statement:

```
cout<<"sum of the first 10 digits = "<< sum;
```

is executed to print the output:

```
sum of the first 10 digits = 55
```

If the statement:

`int i, sum = 0;`

is replaced by `int i, sum = 1;`

Then the output on the screen is:

```
sum of the first 10 digits = 56
```

- **What will be the output if the for loop statement** `for(i=1; i<=10; i++)` **is replaced by the statement** `for(i=2; i<10; i++)`**?**

Answer: sum of 10 digits = 44

If the statement `int i, sum, sum = 0;` is written instead of `int i, sum = 0;`

Then the *compilation error message* will be displayed on the screen (stating that sum is twice declared).

If the *for loop* is ended with a semicolon i.e.,

```
for( i=1; i<=10; i++);
```

Then the ***compilation error*** will be displayed on the console screen.

- **Program 6.9**

  C++ program to print the average of the first 10 numbers using *for loop* statement

```
#include<iostream>
using namespace std;
int main()
{
int i, avg, sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
cout<<"sum of the first 10 numbers = "<< sum;
cout<<"average of the first 10 numbers = "<< avg;
return 0;
}
```

The output on the screen:

```
sum of the first 10 numbers = 55
average of the first 10 numbers = 5
```

The average of the first10 numbers = 55/10 = 5.5 not 5. But the **output on the screen** is:

```
average of the first 10 numbers = 5
```

because int is used instead of float.

If the ***data type*** float is used i.e.,

```
#include<iostream>
using namespace std;
int main()
{
float i, avg, sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
cout<<"sum of the first 10 numbers = "<< sum;
cout<<"average of the first 10 numbers = "<< avg;
return 0;
}
```

The output on the screen:

```
sum of the first 10 numbers = 55
average of the first 10 numbers = 5.5
```

- **Program 7.0**

  C++ program to print the product of the first 10 digits using *for loop* statement

```
#include<iostream>
using namespace std;
int main()
{
```

```
int i, product = 1;
for( i=1; i<=10; i++)
product = product * i;
cout<<"the product of the first 10 digits =%d", product;
return 0;
}
```

The output on the screen:

```
the product of the first 10 digits = 3628800
```

- **How the product of the first 10 digits = 3628800 is outputted on the screen through the for Loop statement?**

```
i=1 (product = 1 because the product is initialized to 1 in the statement int i,
product = 1;)
Is i<=10 true?
Yes, do this
product = product *  i = 1 * 1 =1
Now,
i=2 (product = 1)
Is i<=10 true?
Yes, do this
product = product *  i = 1 * 2 = 2
Now,
i=3 (product = 2)
Is i<=10 true?
Yes, do this
product = product *  i = 2 * 3 = 6
Now,
i=4 (product = 6)
Is i<=10 true?
Yes, do this
product = product *  i = 6 * 4 = 24
Now,
i=5 (product =24)
```

```
Is i<=10 true?
Yes, do this
product = product *  i = 24 * 5 =120
Now,
i=6 (product =120)
Is i<=10 true?
Yes, do this
product = product *  i = 120 * 6 = 720
Now,
i=7 (product =720)
Is i<=10 true?
Yes, do this
product = product *  i = 720 * 7 = 5040
Now,
i=8 (product =5040)
Is i<=10 true?
Yes, do this
product = product *  i = 5040 * 8 = 40320
Now,
i=9 (product = 40320)
Is i<=10 true?
Yes, do this
product = product *  i = 40320 * 9 = 362880
Now,
i=10 (product = 362880)
Is i<=10 true?
Yes, do this
product = product *  i = 362880 * 10 = 3628800

stops because the condition i<=10 is achieved.
```

The statement:

```
cout<<"the product of the first 10 digits = "<< product;
```

is executed to display the output:

```
the product of the first 10 digits = 3628800
```

If the statement:

`int i, product = 1;` is replaced by `int i, product = 0;`

Then the output on the screen is:

```
the product of the first 10 digits = 0
```

If the statement:

`for(i=1; i<=10; i++)` is replaced by

```
for(i=5; i<=8; i++)
```

Then the output on the screen is:

```
the product of the first 10 digits = 1680
```

- **Program 7.1**

  C++ Program to print the table of a number using the *for loop* statement

```
#include<iostream>
using namespace std;
int main()
{
```

```
int n, i;
cout<<"Enter any number:";
cin>>n;
for( i=1; i<=5; i++)
cout<< n <<"  *   "<< i <<"  =  "<< n*i;
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 2 (i.e., n=2)
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
will be outputted on the screen.
```

- **How the execution takes its Way through the *for Loop* statement**

```
Since you entered the number 2, therefore: n=2.
i=1
Is i<=5 true?
Yes, print this
2 * 1 = 2
using the statement cout<< n <<"  *   "<< i <<"  =  "<< n*i;

Now,
i=2
Is i<=5 true?
Yes, print this
2 * 2 = 4
using the statement cout<< n <<"  *   "<< i <<"  =  "<< n*i;

Now,
i=3
Is i<=5 true?
Yes, print this
2 * 3 = 6
using the statement cout<< n <<"  *   "<< i <<"  =  "<< n*i;

Now,
i=4
Is i<=5 true?
Yes, print this
2 * 4 = 8
using the statement cout<< n <<"  *   "<< i <<"  =  "<< n*i;

Now,
i=5
Is i<=5 true?
Yes, print this
2 * 5 = 10
using the statement cout<< n <<"  *   "<< i <<"  =  "<< n*i;

stop Now because the condition i<=5 is achieved.
```

If the symbol * is replaced by +

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
int n, a;
cout<<"Enter any number:";
cin>>n;
for( i=1; i<=5; i++)
cout<< n <<"  +   "<< i <<"  =  "<<  n + i <<endl;
return 0;
}
```

Then the **output on the screen** is:

```
Enter any number:

If you enter the number 2 (i.e., n=2)

2 + 1 = 3

2 + 2 = 4

2 + 3 = 5

2 + 4 = 6

2 + 5 = 7

will be outputted on the screen.
```

time_t time(time_t *time);

This returns the current calendar time of the system in number of seconds elapsed since January 1, 1970. If the system has no time, .1 is returned.

char *ctime(const time_t *time);

This returns a pointer to a string of the form *day month year hours:minutes:seconds year\n\0*.

struct tm *localtime(const time_t *time);

This returns a pointer to the **tm** structure representing local time.

clock_t clock(void);

This returns a value that approximates the amount of time the calling program has been running. A value of .1 is returned if the time is not available.

char * asctime (const struct tm * time);

This returns a pointer to a string that contains the information stored in the structure pointed to by time converted into the form: day month date hours:minutes:seconds year\n\0

struct tm *gmtime(const time_t *time); This returns a pointer to the time in the form of a tm structure. The time is represented in Coordinated Universal Time (UTC), which is essentially Greenwich Mean Time (GMT).
time_t mktime(struct tm *time); This returns the calendar-time equivalent of the time found in the structure pointed to by time.
double difftime (time_t time2, time_t time1); This function calculates the difference in seconds between time1 and time2.
size_t strftime(); This function can be used to format date and time in a specific format.

- **Program 7.2**

  C++ program:

  If you enter a character M

  Output must be: ch = M

```
#include<iostream>
using namespace std;
int main()
{
char M;
cout<<"Enter any character:";
cin>>M;
cout<<"ch= "<< M;
return 0;
}
```

The output on the screen:

```
Enter any character:

If you enter the character M

ch = M

will be outputted on the screen.
```

- **Note:**

If we replace the statement: `cin>>M;` by the statement:

`M = getchar();`

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
char M;
cout<<"Enter any character:";
M = getchar();
cout<<"ch= "<< M;
return 0;
}
```

There will be no change in the output on the screen i.e., The output on the screen is:

```
Enter any character:

If you enter the character K

ch = K

will be outputted on the screen.
```

If we replace the statement: `cout<<"ch= "<< M;` by the statement:

`putchar(M);`

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
char M;
cout<<"Enter any character:";
cin>>M;
putchar(M);
return 0;
}
```

There will be **no change in the output** on the screen i.e., The output on the screen is:

```
Enter any character:
If you enter the character M
M will be outputted on the console screen.
```

If we replace the statement: `cin>>M;` by the statement:

`M = getchar();`

and the statement:

```
cout<<"ch= "<< M;
```

by the statement:

`putchar(M);` i.e.,

```
#include<iostream>
using namespace std;
int main()
{
char M;
cout<<"Enter any character:";
M = getchar();
putchar(M);
return 0;
}
```

The output on the screen:

```
Enter any character:
If you enter the character S
S will be outputted on the screen.
```

- **Write a program to print the absolute value of a number**

Answer:

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
int a, b;
a= - 2;
b= abs(a);
cout<<" absolute value of a = "<< b<< endl;
return 0;
}
```

The output on the screen:

```
absolute value of a = 2
```

- **Program 7.2**

  C ++ program to print the first 5 numbers starting from one together with their squares

```
#include<iostream>
using namespace std;
int main()
{
int i;
for( i=1; i<=5; i++)
cout<<"\n number = "<< i <<"its square =  "<< i*i;
return 0;
}
```

The output on the screen:

```
number=1 its square=1
number=2 its square=4
number=3 its square=9
number=4 its square=16
number=5 its square=25
```

- **How the execution takes its way through the *for loop* statement**

```
i=1
Is i<=5 true?
```

```
Yes, print this
number=1 its square=1
using the statement cout<<"\n number = "<< i <<"its square =  "<< i*i;

Now,
i=2
Is i<=5 true?
Yes, print this
number=2 its square=4
using the statement cout<<"\n number = "<< i <<"its square =  "<< i*i;

Now,
i=3
Is i<=5 true?
Yes, print this
number=3 its square=9
using the statement cout<<"\n number = "<< i <<"its square =  "<< i*i;

Now,
i=4
Is i<=5 true?
Yes, print this
number=4 its square=16
using the statement cout<<"\n number = "<< i <<"its square =  "<< i*i;

Now,
i=5
Is i<=5 true?
Yes, print this
number=5 its square=25
using the statement cout<<"\n number = "<< i <<"its square =  "<< i*i;

stop Now because the condition (i<=5) is achieved.
```

- **Note:**

If the statement

```
cout<<"\n number = "<< i <<"its square = "<< i*i;
```

is replaced by the statement:

```
cout<<"\n number =   "<< i <<"\t its square =  "<< i*i;
```

Then the **output on the screen** is:

```
number=1     its square=1
number=2     its square=4
number=3     its square=9
number=4     its square=16
number=5     its square=25
```

tab /t is included because to leave space between

number=1   and   its square=1

Suppose cout<<"\n number =   "<< i <<"\t its square = "<< i*i; is replaced by the statement

```
cout<<"\n number =   "<< i <<"\n its square =  "<< i*i;
```

Then the **output on the screen** is:

```
number=1
its square=1
number=2
its square=4
```

```
number=3
its square=9
number=4
its square=16
number=5
its square=25
```

- **Write a program to print the first 10 numbers starting from one together with their squares and cubes?**

Answer:

```
#include<iostream>
using namespace std;
int main()
{
int i;
for( i=1; i<=10; i++)
cout<<"number =  "<< i <<" its square =  "<< i*i <<" its cube =  "<< i*i*i<< endl;
return 0;
}
```

- **Program 7.3**

  C ++ program to print the address of x and the value assigned to x

```
#include<iostream>
using namespace std;
int main()
{
int x, *p;
cout<<"Enter any integer:";
cin>>x;
p = &x;
cout<<"The address of the variable x = "<< p;
cout<<"The value of the variable x = "<< *p;
return 0;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 1
The address of the variable x = 0x7fffc60478a4
The value of the variable x = 1
will be outputted on the screen.
```

If the statements:

```
cout<<"The address of the variable x = "<< p;
cout<<"The value of the variable x = "<< *p;

are replaced by the statement:

cout<<"The address of the variable x = "<< p <<"its value = "<< *p;
```

i.e.,

```
#include<iostream>
using namespace std;
int main()
{
int x, *p;
cout<<"Enter any integer:";
cin >> x;
p = &x;
cout<<"The address of the variable x = "<< p <<"its value = "<< *p;
return 0;
}
```

Then the **output on the screen** is:

The address of the variable x = 0x7fff78508cc4 its value = 2

C++ program to print the sum of two numbers using pointers

```
#include<iostream>
using namespace std;
int main()
{
int x, y, *p, *q, sum;
cout<<"Enter any number:";
cin >> x;
cout<<"Enter any number:";
cin >> y;
p = &x;
q = &y;
sum = *p + *q;
cout<<"\n sum of entered numbers =  "<< sum;
return 0;
}
```

The output on the screen:

```
Enter any number:

If you enter the number 4

Enter any number:

If you enter the number 3
```

```
sum of entered numbers = 7

will be outputted on the screen.
```

- **C++ program to print the product, subtraction and division of two numbers using pointers**

```
#include<iostream>
using namespace std;
int main()
{
int x, y, *p, *q, product, subtract, div;
cout<<"Enter any number:";
cin>> x;
cout<<"Enter any number:";
cin>> y;
p = &x;
q = &y;
product = *p * *q;
subtract = *p - *q;
div= *p / *q;
cout<<"\n product of entered numbers =  "<< product;
cout<<"\n subtract of entered numbers =  "<< subtract;
cout<<"\n division of entered numbers =  "<< div;
return 0;
}
```

The output on the screen:

```
Enter any number:
If you enter the number 4
Enter any number:
If you enter the number 2
product of entered numbers = 8
```

```
subtract of entered numbers = 2
division of entered numbers = 2

will be displayed on the screen.
```

- **C++ program to find the greatest of two numbers using pointers**

```
#include<iostream>
using namespace std;
int main()
{
int x, y, *p, *q;
cout<<"Enter any integer:";
cin>> x;
cout<<"Enter any integer:";
cin>> y;
p = &x;
q = &y;
if(*p>*q)
{
cout<<"x is greater than y";
}
else
{
cout<<"y is greater than x";
}
return 0;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 10
Enter any integer:
```

```
If you enter the integer 16
y is greater than x
will be outputted on the screen.
```

- **What is the output of the following programs:**

A)

```
#include <iostream>
using namespace std;
int main()
{
int x;
x=12;
cout<<"per = "<< x;
return 0;
}
```

Answer:

```
per=12
```

B)

```
#include <iostream>
using namespace std;
int main()
{
int x, t, c;
x=12;
t=2;
c = x/t;
cout<<"velocity =  "<< c <<"m/s";
return 0;
}
```

Answer:

```
velocity = 6 m/s
```

- **Program 7.4**

C++ program to print the sum of two numbers using functions

```
#include<iostream>
using namespace std;
int addition();
int main()
{
int answer;
answer = addition();
cout<<"The sum of two numbers is: "<<answer;
return 0;
}

int addition()
{
int x, y;
cout<<"Enter any integer:";
cin>>x;
cout<<"Enter any integer:";
cin>>y;
return x+y;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 3
Enter any integer:
If you enter the integer 5
sum of two numbers = 8
will be displayed on the screen.
```

- **C++ program to print the product of two numbers using functions**

```
#include<iostream>
using namespace std;
int multiplication();
int main()
{
int answer;
answer = multiplication();
cout<<"The product of two numbers is: "<<answer;
return 0;
}

int multiplication()
{
int x, y;
cout<<"Enter any integer:";
cin>>x;
cout<<"Enter any integer:";
cin>>y;
return x*y;
}
```

The output on the screen:

```
Enter any integer:
```

If you enter the integer 3

Enter any integer:

If you enter the integer 5

product of two numbers = 15

will be outputted on the screen.

- **C++ program to print the greatest of two numbers using functions**

```
#include<iostream>
using namespace std;
int largest();
int main()
{
int answer;
answer = largest();
cout<<"The largest of two numbers is: "<<answer;
return 0;
}

int largest()
{
int x, y;
cout<<"Enter any integer:";
cin>>x;
cout<<"Enter any integer:";
cin>>y;
if(x>y)
return x;
if(y>x)
return y;
}
```

The output on the screen:

```
Enter any integer:
If you enter the integer 3
Enter any integer:
If you enter the integer 5
largest of two numbers= 5
will be outputted on the screen.
```

- **C++ program to print the greatest of three numbers using functions**

```
#include<iostream>
using namespace std;
int largest();
int main()
{
int answer;
answer = largest();
cout<<"largest of three numbers= "<< answer;
return 0;
}
int largest()
{
int x, y, z;
cout<<"Enter any integer:";
cin>>x;
cout<<"Enter any integer:";
cin>>y;
cout<<"Enter any integer:";
cin>>z;

if(x>y&& x>z)
return x;
if(y>x&& y > z)
return y;
if(z>x && z>y)
return z;
}
```

The output on the screen:

```
Enter any integer:

If you enter the integer 3

Enter any integer:

If you enter the integer 5

Enter any integer:

If you enter the integer 10

largest of three numbers = 10

will be outputted on the screen.
```

- **C++ program to print the square of the number using functions**

```
#include<iostream>
using namespace std;
int square(;
int main()
{
int answer;
answer = square();
cout<<"square of the given number = "<< answer;
return 0;
}
int square()
{
int x;
cout<<"Enter any integer:";
cin>>x;
return x*x;
}
```

The output on the screen:

```
Enter any integer:
If you enter an integer 5
square of the number = 25
will be outputted on the screen.
```

- **What is the output of the following program:**

```
#include<iostream>
using namespace std;
int main()
{
int x;
x=6;
cout<<"The address of x = "<<&x;
return 0;
}
```

Answer:

```
The address of x = 0x7ffd80d2c06c
```

<iostream> This file defines the **cin**, **cout**, **cerr** and **clog** objects, which correspond to the standard input stream, the standard output stream, the un-buffered standard error stream and the buffered standard error stream, respectively.
<iomanip> This file declares services useful for performing formatted I/O with so-called parameterized stream manipulators, such as **setw** and **setprecision**.
<fstream>

This file declares services for user-controlled file processing. We will discuss about it in detail in File and Stream related chapter.

- **Program 7.5**

  C++ has an inbuilt multiple branch selection statement [called ***switch***] which successively tests the value of an expression against a list of integer or character constants. When a match is found, the statement sequences associated with that constant are executed. `Switch(case)` allows us to make decision from the number of choices i.e., from the number of cases.

For example:

```
#include<iostream>
using namespace std;
int main()
{
char ch;
cout<<"Enter any character:";
cin>>ch;
switch(ch)
{
case 'R':
cout<<"Red";
break;
case 'W':
cout<<"White";
break;
case 'Y':
cout<<"Yellow";
break;
case 'G':
cout<<"Green";
break;
default:
cout<<"Error";
break;
}
return 0;
}
```

The output on the screen:

```
Enter any character:

If you enter a character R

Red

will be outputted on the screen.
```

If the statements:

```
case 'R':
cout<<"Red";
break;
case 'W':
cout<<"White";
break;
case 'Y':
cout<<"Yellow";
break;
case 'G':
cout<<"Green";
break;
default:
cout<<"Error";
break;
```

are replaced by the statements:

```
case 'R':
cout<<"Red";
case 'W':
cout<<"White";
```

```
case 'Y':
cout<<"Yellow";
break;
case 'G':
cout<<"Green";
break;
default:
cout<<"Error";
break;
```

Then the **output on the screen** is:

```
Red

White

Yellow
```

i.e., the output will be printed till yellow even though you have entered the **character R**.

- In C, a switch can have at least 257 case statements. C++ recommends that at least 16,384 case statements be supported.

Why do we need arrays?

Arrays provide a more convenient way of storing variables of a similar data type together instead of storing them separately. Each value of the array will be accessed separately.

- **Program 7.6**

  C++ program to print the output:

  ```
  Element [0] = 16

  Element [1] = 18
  ```

```
Element [2] = 20

Element [3] = 25

Element [4] = 36
```

using arrays:

```
#include<iostream>
using namespace std;
main()
{
int i;
int num [5] = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
cout<<"Element ["<< i <<" ] = "<<  num[i] << endl;
return 0;
}
```

The output on the screen:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
```

Suppose the statement:

```
cout<<"Element ["<< i <<" ] = "<<  num[i] << endl;
```

is replaced by the statement:

```
cout<<"Element ["<< i <<" ] = "<<  num[0] << endl;
```

Then the **output on the screen**:

```
Element [0] = 16

Element [1] = 16

Element [2] = 16

Element [3] = 16

Element [4] = 16
```

Suppose the statement:

```
                cout<<"Element ["<< i <<" ] = "<<  num[i] << endl;
```

is replaced by the statement:

```
                cout<<"Element ["<< i <<" ] = "<<  num[1] << endl;
```

The output on the screen:

```
Element [0] = 18
Element [1] = 18
Element [2] = 18
Element [3] = 18
Element [4] = 18
```

Suppose the statement:

```
cout<<"Element ["<< i <<" ] = "<<  num[i] << endl;
```

is replaced by the statement:

```
cout<<"Element ["<< i <<" ] = "<<  num[2] << endl;
```

The output on the screen:

```
Element [0] = 20
Element [1] = 20
Element [2] = 20
Element [3] = 20
Element [4] = 20
```

Suppose the statement:

```
cout<<"Element ["<< i <<" ] = "<<  num[i] << endl;
```

is replaced by the statement:

```
cout<<"Element ["<< i <<" ] = "<<  num[3] << endl;
```

The output on the screen:

```
Element [0] = 25

Element [1] = 25

Element [2] = 25

Element [3] = 25

Element [4] = 25
```

Suppose the statement:

`cout<<"Element ["<< i <<" ] = "<<  num[i] << endl;` is replaced by the statement:

```
cout<<"Element ["<< i <<" ] = "<<  num[4] << endl;
```

The output on the screen:

```
Element [0] = 36
Element [1] = 36
Element [2] = 36
Element [3] = 36
Element [4] = 36
```

If the condition:

`i<5`

is replaced by the condition:

`i<=5`

Then the **output on the screen** is:

```
Element [0] = 16
Element [1] = 18
Element [2] = 20
Element [3] = 25
Element [4] = 36
Element [5] = 3656

3656 is the number stored in the memory i.e., any number stored in the memory will be
displayed.
```

If the statement:

`int num [5] = {16, 18, 20, 25, 36};` is replaced by the statement:

```
int num [i] = {16, 18, 20, 25, 36};
```

Then the ***compilation error*** will be displayed on the screen because there are 5 elements within the braces {} not `i` elements.

Class Member Functions

A member function of a class is a function that has its definition or its prototype within the class definition like any other variable.

Class Access Modifiers

A class member can be defined as public, private or protected. By default members would be assumed as private.

Constructor & Destructor

A class constructor is a special function in a class that is called when a new object of the class is created. A destructor is also a special function which is called when created object is deleted.

Copy Constructor

The copy constructor is a constructor which creates an object by initializing it with an object of the same class, which has been created previously.

Friend Functions

A **friend** function is permitted full access to private and protected members of a class.

Inline Functions

With an inline function, the compiler tries to expand the code in the body of the function in place of a call to the function.

this Pointer

Every object has a special pointer **this** which points to the object itself.

Pointer to C++ Classes

A pointer to a class is done exactly the same way a pointer to a structure is. In fact a class is really just a structure with functions in it.

Static Members of a Class

Both data members and function members of a class can be declared as static.

- **C++ program to print the sum of the elements in array.**

```
#include<iostream>
using namespace std;
int main()
{
int i, sum = 0;
int num [5] = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
sum = sum + num[i];
cout<<"Sum of the Elements in the array = "<< sum;
return 0;
}
```

The output on the screen:

```
Sum of the Elements in the array = 115
i.e., 16 + 18 + 20 + 25 + 36 = 115
```

If the statement:

```
int i, sum = 0;
```

is replaced by `int i, sum = 1;`

Then The **output on the screen**:

```
Sum of the Elements in the array = 116
```

- **C++ program to print the average of the elements in array**

```
#include<iostream>
using namespace std;
int main()
{
int i, avg, sum = 0;
int num [5] = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
sum = sum + num [i];
avg = sum/5;
cout<<"Sum of the Elements in the array = "<< sum;
cout<<"average of the elements in the array= "<< avg;
return 0;
}
```

The output on the screen:

```
Sum of the Elements in the array = 115

average of the elements in the array = 23
```

- **Write a program to print:**

Einstein [0] = E

Einstein [1] = I

Einstein [2] = N

Einstein [3] = S

Einstein [4] = T

Einstein [5] = E

Einstein [6] = I

Einstein [7] = N

using arrays

Answer:

```
#include<iostream>
using namespace std;
int main()
{
int i;
char name [8] = {'E' , 'I', 'N', 'S', 'T', 'E', 'I', 'N'};
for(i=0; i<8; i++)
cout<<"Element ["<< i <<" ] = "<< name[i] << endl;
return 0;
}
```

- **What will be the output of the following programs?**

i)

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
cout<<""<< cbrt(27);
return 0;
}
```

Answer:

3

ii)

```
#include <iostream>
using namespace std;
int main()
{
char i;
char body [4] = {'b', 'o', 'd', 'y'};
for(i=0; i<4; i++)
cout<<"\n body ["<<body[i] <<" ] = "<< body[i] << endl;
return 0;
}
```

Answer:

body [b] = b

body [o] = o

body [d] = d

body [y] = y

iii)

```
#include <iostream>
#include <malloc.h>
using namespace std;
int main()
{
int x=2;
cout<<""<< malloc (200*sizeof(x));
return 0;
}
```

Answer:

8183824

- **Program 7.7**

C++ program to print the output:

```
Name of the book = B

Price of the book = 135.00

Number of pages = 300

Edition = 8
```

using structures

```
#include<iostream>
using namespace std;
int main()
{
struct book {
char name;
float price;
int pages;
int edition;
};
struct book b1= {'B', 135.00, 300, 8};
cout<<"Name of the book =  "<< b1.name<< endl;
cout<<"Price of the book = "<< b1.price<<endl;
cout<<"Number of pages = "<< b1.pages<<endl;
cout<<"Edition of the book = "<< b1.edition<< endl;
return 0;
}
```

The output on the screen:

```
Name of the book = B

Price of the book = 135.00

Number of pages = 300

Edition of the book = 8
```

- **What will be output of the following programs?**

A)

```
#include<iostream>
using namespace std;
struct book {
char name;
float price;
int pages;
int edition;
};
int main()
{
struct book b1;
b1.name = 'C';
b1.price = 135.00;
b1.pages = 300;
b1.edition = 8;
cout<<"Name of the book = bulgarian "<<  b1.name << endl;
cout<<"\n Price of the book = "<< b1.price;
cout<<"\n Number of pages = "<< b1.pages<<endl;
cout<<"\n Edition of the book = "<< b1.edition;
}
```

Answer:

```
Name of the book = B

Price of the book = 135.000000

Number of pages = 300

Edition of the book = 8
```

B)

```
#include <iostream>
using namespace std;
int main()
{

for( ; ; ) {
cout<<"This loop will run forever.\n";
}

return 0;
}
```

Answer:

```
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.

......... continues
```

- **Program 7.8**

  Continue and break statements:

  i)

```
#include <iostream>
using namespace std;
int main()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
continue;
}

cout<<"\n "<< i;
}
return 0;

}
```

Output on the screen:

```
1

2

4
```

5

ii)

```
#include <iostream>
using namespace std;
int main()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
break;
}
cout<<"\n "<< i;
}
return 0;
}
```

Output on the screen:

```
1
2
```

Access	**public**	**protected**	**private**
Same class	yes	yes	yes
Derived classes	yes	yes	no
Outside classes	yes	no	no

- **Program 7.9**

C++ program to convert the upper case letter to lower case letter

```
#include<iostream>
using namespace std;
int main()
{
char ch = 'A';
char b = tolower(ch);
cout<<" upper case letter "<< ch <<" is converted to lower case letter "<< b;
return 0;
}
```

Output on the screen:

```
upper case letter A is converted to lower case letter a
```

If you want to enter the ***character*** through the keyboard, then the above program should take the form:

```
#include<iostream>
using namespace std;
int main()
{
char ch;
cout<<"Enter any character:";
cin>>ch;
char b = tolower(ch);
cout<<" upper case letter "<< ch <<" is converted to lower case letter "<< b;
return 0;
}
```

Output on the screen:

```
Enter any character:
If you enter the character C
```

```
upper case letter C is converted to lower case letter c will be outputted on the
screen.
```

- **Program 8.0**

  C++ program to convert the lower case letter to upper case letter

```
#include<iostream>
using namespace std;
int main()
{
char ch = 'a';
char b = toupper(ch);
cout<<" lower case letter "<<ch<<" is converted to upper case letter "<<b;
return 0;
}
```

Output on the screen:

```
lower case letter a is converted to upper case letter A
```

If you want to enter the character **through the keyboard**, then the above program should take the form:

```
#include<iostream>
using namespace std;
int main()
{
char ch;
cout<<"Enter any character:";
cin>>ch;
char b = toupper(ch);
cout<<" lower case letter "<<ch<<" is converted to upper case letter "<<b;
return 0;
}
```

Output on the screen:

```
Enter any character:

If you enter the character h

lower case letter h is converted to upper case letter H

will be outputted on the screen.
```

Following is the list of operators which can be overloaded:

+	-	*	/	%	^
&	|	~	!	,	=
<	>	<=	>=	++	--
<<	>>	==	!=	&&	||
+=	-=	/=	%=	^=	&=
|=	*=	<<=	>>=	[]	()
->	->*	new	new []	delete	delete []

Following is the list of operators, which cannot be overloaded:

::	.*	.	?:

- **Program 8.1**

C++ program to test whether the entered character is upper case letter or not

```
#include<iostream>
using namespace std;
int main()
{
char ch = 'a';
if(isupper(ch))
cout<<"you have entered the upper case letter";
else
cout<<"you have entered the lower case letter";
return 0;
}
```

Output on the screen:

```
you have entered the lower case letter
```

If the statement:

```
char ch = 'a';
```

is replaced by the statement:

```
char ch = 'A';
```

Then the output on the screen is:

```
you have entered the upper case letter
```

- **Program 8.2**

  C++ program to test whether the entered character is lower case letter or not

```
#include<iostream>
using namespace std;
int main()
{
char ch = 'a';
if(islower(ch))
cout<<"you have entered the lower case letter";
else
cout<<"you have entered the upper case letter";
return 0;
}
```

Output on the screen:

```
you have entered the lower case letter
```

- **Program 8.3**

  C++ program to print the value of tan inverse x (i.e., the value of $\tan^{-1}$ x)

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
int x = 20;
cout<<"the value of tan inverse x = "<< atan(x);
return 0;
}
```

Output on the screen:

```
the value of tan inverse x = 1.520838
```

- **Program 8.4**

C++ program to print the value of tan inverse $\frac{x}{y}$ (i.e., the value of $\tan^{-1}\frac{x}{y}$)

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
int x, y;
x = 20;
y =20;
cout<<"the value of tan inverse x/y = "<< atan2(x,y);
return 0;
}
```

Output on the screen:

```
the value of tan inverse x/y = 0.785398
```

- **Program 8.5**

C++ program to print the value of fmod(x, y)

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
float x = 20.500000;
float y =20.799999;
cout<<" the remainder of "<<x <<" divided by "<<y <<" is: "<< fmod(x,y);
return 0;
}
```

Output on the screen:

```
the remainder of 20.500000 divided by 20.799999 is 20.500000
```

- **Program 8.6**

  C++ program to print the value of ~x

```
#include<iostream>
using namespace std;
int main()
{
int x, y;
x = 205;
y=~x;
cout<<"the value of y is: "<< y;
return 0;
}
```

Output on the screen:

```
the value of y is:-206
```

If the statement:

`y=~x;` is replaced by the statement:

`y= －(~x);`

Then the output on the screen is:

```
the value of y is: 206
```

What will be the output of the following programs:

i)

```
#include<iostream>
using namespace std;
int main()
{
int i = 54;
int y = i<<1;
cout<<"The value of y = "<< y;
return 0;
}
```

Answer:

```
The value of y = 108
```

If the statement:

`i<<1` is replaced by the statement: `i<<2`

Then the **output on the screen** is:

```
The value of y = 216
```

ii)

```
#include<iostream>
using namespace std;
int main()
```

```
{
int i = 54;
int y = i>>1;
cout<<"The value of y = "<< y;
return 0;
}
```

Answer:

The value of y = 27

If the statement:

i>>1 is replaced by the statement: i>>2

Then the **output on the screen** is:

```
The value of y = 13
```

- **Program 8.7**

  C++ program to print the length of the entered character (i.e., to print the length of the string)

```
#include<iostream>
#include<string.h>
using namespace std;
int main()
{
```

```
char ch[4];
cout<<"Enter any word: ";
cin>>ch;
cout<<"The length of the string = "<< strlen(ch);
return 0;
}
```

Output on the screen:

```
Enter any word:
If you enter the word dog

The length of the string = 3

will be displayed on the console screen because there are three letters in the word
dog.
```

Suppose if you enter the **word** tech

The length of the string = 4

will be displayed on the console screen because there are four letters in the word tech.

std::exception An exception and parent class of all the standard C++ exceptions.
std::bad_alloc This can be thrown by **new**.

std::bad_cast This can be thrown by **dynamic_cast**.
std::bad_exception This is useful device to handle unexpected exceptions in a C++ program.
std::bad_typeid This can be thrown by **typeid**.
std::logic_error An exception that theoretically can be detected by reading the code.
std::domain_error This is an exception thrown when a mathematically invalid domain is used.
std::invalid_argument This is thrown due to invalid arguments.
std::length_error This is thrown when a too big std::string is created.
std::out_of_range This can be thrown by the 'at' method, for example a std::vector and std::bitset<>::operator[]().
std::runtime_error An exception that theoretically cannot be detected by reading the code.
std::overflow_error This is thrown if a mathematical overflow occurs.
std::range_error This is occurred when you try to store a value which is out of range.
std::underflow_error This is thrown if a mathematical underflow occurs.

- **Program 8.8**

  C++ program to print the factorial of the entered number

```
#include<iostream>
using namespace std;
int main()
{
int i, n, fact=1 ;
cout<<"Enter any number:";
cin>>n;
for(i=1; i<=n; i++)
fact = fact *i;
cout<<"\n Entered number is: "<< n;
cout<<"\n The factorial of the entered number"<<n<<"is:"<< fact;
return 0;
}
```

Output on the screen:

```
Enter any number:
If you enter the number 2
Entered number is: 2
The factorial of the entered number 2 is: 2
will be displayed on the screen.
```

Suppose if you enter the number 4

```
Entered number is: 4
The factorial of the entered number 4 is: 24
```

will be displayed on the screen.

Linux

Developer	Community Linus Torvalds
Written in	C, Assembly language
OS family	Unix-like
Working state	Current
Source model	Open source
Initial release	September 17, 1991; 28 years ago
Marketing target	Cloud computing, embedded devices, mainframe computers, mobile devices, personal computers, servers, supercomputers
Available in	Multilingual
Platforms	Alpha, ARC, ARM, C6x, H8/300, Hexagon, Itanium, m68k, Microblaze, MIPS, NDS32, Nios II, OpenRISC, PA-RISC, PowerPC, RISC-V, s390, SuperH, SPARC, Unicore32, x86, XBurst, Xtensa
Kernel type	Monolithic
Userland	GNU[a]
Default user interface	Unix shell
License	GPLv2 and others (the name "Linux" is a trademark[b])
Official website	www.linuxfoundation.org

What is Linux and why is it so popular?

Whether you know it or not you are already using Linux (the best-known and most-used open source operating system) every day. From supercomputers to smartphones, the Linux operating system is everywhere. As an operating system, Linux is a family of open source Unix-like software based on the Linux kernel - that sits underneath all of the other software on a computer, receiving requests from those programs and relaying these requests to the computer's hardware. With regard to careers, it is becoming increasingly valuable to have Linux skills rather than just knowing how to use Windows. In general, Linux is harder to manage than Windows, but offers more flexibility and configuration options.

Every desktop computer uses an operating system. The most popular operating systems in use today are: Windows, Mac OS, and LINUX. Linux is the best-known notoriously reliable and highly secure open source portable operating system -- very much like UNIX -- that has become very popular over the last several years -- created as a task done for pleasure by Linus Torvalds -- computer science student at the University of Helsinki in Finland -- in the early 1990s and later developed by more than a thousand people around the world.

Linux is fast, free and easy to use, that sits underneath all the other software on a computer − runs your computer -- handling all interactions between you and the hardware i.e., whether you're typing a letter, calculating a money budget, or managing your food recipes on your computer, the Linux operating system (similar to other Operating Systems, such as Windows XP, Windows 7, Windows 8, and Mac OS X) provides the essential air that your computer breathes.

Linux is the most important technology advancement of the twenty-first century and Licensed under the General Public License (GPL) that Linux uses ensures that the software will always be open to anyone and whose source code is open and available for any user to check, which makes it easier to find and repair vulnerabilities and it power the laptops, development machines and servers at Google, Facebook, Twitter, NASA, and New York Stock Exchange, just to name a few. Linux has many more features to amaze its users such as: Live CD/USB, Graphical user interface (X Window System) etc.

Why LINUX?

Although Microsoft Windows (which is the most likely the victim of viruses and malware) has made great improvements in reliability in recent years, it considered less reliable than Linux. Linux is notoriously reliable and secure and it is free from constant battling viruses and malware (which may affect your desktops, laptops, and servers by corrupting files, causing slow downs, crashes, costly repairs and taking over basic functions of your operating system) – and it keep yourself free from licensing fees i.e., zero cost of entry ... as in free. You can install Linux on as many reliable computer ecosystems on the planet as you like without paying a cent for software or server licensing. While Microsoft Windows usually costs between $99.00 and $199.00 USD for each licensed copy and fear of losing data.

Below are some examples of where Linux is being used today:

- Android phones and tablets
- Servers
- TV, Cameras, DVD players, etc.
- Amazon
- Google
- U.S. Postal service
- New York Stock Exchange

Linux Operating System has primarily three components:

- **Kernel**

Kernel is the core part of Linux Operating System and interacts directly with hardware. It is responsible for all major activities of the Linux operating system.

- **System Library**

System libraries are special programs using which application programs accesses Kernel's features.

- **System Utility**

System Utility programs are responsible to do specialized tasks.

Important features of Linux Operating System:

- Portable
- Open Source
- Multi-User
- Multiprogramming
- Hierarchical File System
- Security

Now Linux (successfully being used by several millions of users worldwide) has grown passed the stage where it was almost exclusively an academic system, useful only to a handful of people with a technical background. It provides more than the operating system: there is an entire infrastructure supporting the chain of effort of creating an operating system, of making and testing programs for it, of bringing everything to the users, of supplying maintenance, updates and support and customizations, runs on different platforms including the Intel and Alpha platform. Today, Linux is ready to accept the challenge of a fast-changing world to do various types of operations, call application programs etc. Since the hiring focus is shifting more and more toward DevOps type skills, a Linux skill set will be the types of things that will make you very deployable.

The command-line interface is one of the nearly all well built trademarks of Linux. There exists an ocean of Linux commands, permitting you to do nearly everything you can be under the impression of doing on your Linux operating system. Although, this to the end of time creates a problem: by all of so copious commands accessible to manage, you don't comprehend where and at which point to fly learning them, especially when you are learner. If you are facing this problem, and are peering for a painless method to begin your command line journey in Linux, you've come to the right place, we will launch you to a hold of well liked and helpful Linux commands.

Description:

Display system date and time.

Command:

```
date
```

Description:

Display calendar.

Command:

```
cal
```

Description:

Display date, time and calendar.

Command:

```
date & cal
```

Description:

Display August month 2016 year calendar.

Command:

```
cal 8 2016
```

Description:

Used to clear the terminal window.

Command:

```
clear
```

Description:

Exit from the terminal window.

Command:

```
exit
```

Description:

Display free and used system memory.

Command:

```
free
```

Description:

Display free and used system memory in bytes.

Command:

```
free  -b
```

Description:

Display free and used system memory in kilobytes.

Command:

```
free  -k
```

Description:

Display free and used system memory in megabytes.

Command:

```
free  -m
```

Description:

Change user password.

Command:

```
passwd
```

Description:

Power-off the machine.

Command:

```
shutdown
```

Description:

Power-off the machine immediately.

Command:

```
shutdown  -h now
```

Description:

Power-off the machine after 10 minutes.

Command:

```
shutdown  -h +10
```

Description:

Print current working directory.

Command:

```
echo $PWD
```

Description:

Print previous working directory.

Command:

```
echo $OLDPWD
```

Description:

Executes the 11th command in command history.

Command:

```
!11
```

Description:

Reveals your command history.

Command:

```
history
```

Description:

Power off or reboot the Operating system.

Command:

```
sudo reboot
```

Description:

Display the IP address of the host.

Command:

```
ip address
```

Description:

List the size of files and directories.

Command:

```
ls  -s
```

Description:

View mounted file systems.

Command:

```
mount
```

Description:

Display the information of disk usage of files and directories.

Command:

```
du
```

Description:

Tells you how long the system has been running.

Command:

```
uptime
```

Description:

Set current date as 02 Nov 1988.

Command:

```
date -- set 1998-11-02
```

Description:

Set current time as 12:11:02 IST.

Command:

```
date -- set 12:11:02
```

Description:

View and change the configuration of the network interfaces on the system.

Command:

```
ifconfig
```

Description:

Lists files.

Command:

```
ls
```

Description:

Report the process information.

Command:

```
ps
```

Description:

Display disk usage.

Command:

```
df
```

Description:

Display disk usage in gigabytes, megabytes, or kilobytes.

Command:

```
df  -H
```

Description:

Delete every file and every directory.

Command:

```
rm  -r *
```

Description:

Provides a quick overview of the currently running processes.

Command:

```
top
```

Description:

The system performs an immediate reboot.

Command:

```
reboot
```

Description:

Terminate processes without having to log out or reboot.

Command:

```
kill
```

Description:

Change the current working directory.

Command:

```
cd
```

Description:

Create a new session on the system.

Command:

```
login
```

Description:

List open files.

Command:

```
lsof
```

Description:

List USB devices.

Command:

```
lsusb
```

Description:

Check the status of the network services.

Command:

```
service network status
```

Description:

Start the network service.

Command:

```
service network start
```

Description:

Stop the network service.

Command:

```
service network stop
```

Description:

Restart the network service.

Command:

```
service network restart
```

Description:

Report information about the users currently on the machine and their processes.

Command:

```
w
```

Description:

Display the current directory.

Command:

```
pwd
```

Description:

Displays CPU architecture information (such as number of CPUs, threads, cores, sockets, and more).

Command:

```
lscpu
```

Description:

Displays the number of processing units available to the current process.

Command:

```
nproc
```

Description:

The system performs an immediate reboot.

Command:

```
init 6
```

Description:

Power-off the machine.

Command:

```
init 0
```

Description:

List files by date.

Command:

```
ls -lrt
```

Description:

Report information about storage devices such as hard disks, flash drives etc.

Command:

```
lsblk
```

Description:

Show exit status of previous command.

Command:

```
echo $?
```

Description:

Lists a few useful info commands.

Command:

```
info
```

Description:

Prints current year's calendar.

Command:

```
cal -y
```

Description:

Check the status of all the services.

Command:

```
service --status-all
```

Description:

Display time in hh:mm:ss.

Command:

```
date +%T
```

Description:

Tells when the user last logged on and off and from where.

Command:

```
last  -1 username
```

Description:

Sort files and directories by extension name.

Command:

```
ls  -X
```

Description:

Display the manual for the pwd command.

Command:

```
man pwd
```

Description:

Displays information about running processes in the form of a tree.

Command:

```
pstree
```

Description:

Resets your terminal.

Command:

```
reset
```

Description:

Displays What date is it this Friday.

Command:

```
date -d fri
```

Description:

Displays the size of each individual file.

Command:

```
du  -a
```

Description:

Display information about the Advanced configuration and power Interface.

Command:

```
acpi
```

Description:

Takes you two folders back.

Command:

```
cd ../..
```

Description:

Takes you to the previous directory.

Command:

```
cd -
```

Description:

Displays a list of shell built-in commands.

Command:

```
help
```

Description:

Lists your last logins.

Command:

```
last yourusername
```

Description:

Create a new directory called myfiles.

Command:

```
mkdir myfiles
```

Description:

Remove the directory myfiles.

Command:

```
rmdir myfiles
```

Description:

Disable password for a specific user "root1".

Command:

```
passwd -d root1
```

Description:

Switch to user "root1".

Command:

```
sudo su root1
```

Description:

Exit from the terminal window.

Command:

```
logout
```

Description:

Creates a user "root1".

Command:

```
useradd "root1"
```

Description:

Assign password to user "root1".

Command:

```
passwd "root1"
```

Description:

Repeats the last command.

Command:

```
!!
```

Description:

Display Who you are logged in as.

Command:

```
whoami
```

Description:

Display the login name of the current user.

Command:

```
logname
```

Description:

Report the name of the kernel.

Command:

```
uname
```

Description:

Print the kernel version.

Command:

```
uname  -v
```

Description:

Print the operating system.

Command:

```
uname  -o
```

Description:

Report the machine hardware name.

Command:

```
uname  -m
```

Description:

Print version information and exit.

Command:

```
uname  --version
```

Description:

Print the kernel release.

Command:

```
uname  -r
```

Description:

Report the network node hostname.

Command:

```
uname  -n
```

Description:

Display all port connections (both TCP and UDP).

Command:

```
netstat  -a
```

Description:

Display only TCP (Transmission Control Protocol) port connections.

Command:

```
netstat  -at
```

Description:

Display only UDP (User Datagram Protocol) port connections.

Command:

```
netstat  -au
```

Description:

Display all active listening ports.

Command:

```
netstat  -I
```

Description:

Display all active listening TCP ports.

Command:

```
netstat  -It
```

Description:

Display all active listening UDP ports.

Command:

```
netstat  -lu
```

Description:

Reveal all the information about the current user (user id, username, group id, group name etc.).

Command:

```
id
```

Description:

Reveal all the information about the user "root1" (user id, username, group id, group name etc.).

Command:

```
id  root1
```

Description:

Print the machine's architecture.

Command:

```
arch
```

Description:

Display the list of available fonts.

Command:

```
fc-list
```

Description:

Create two directories (myfiles, files).

Command:

```
mkdir myfiles files
```

Description:

install apache (CentOS).

Command:

```
yum install httpd
```

Description:

install apache (Ubuntu).

Command:

```
apt install httpd
```

Description:

upgrade apache (CentOS).

Command:

```
yum update httpd
```

Description:

upgrade apache (Ubuntu).

Command:

```
apt update httpd
```

Description:

uninstall apache (CentOS).

Command:

```
yum remove httpd
```

Description:

uninstall apache (Ubuntu).

Command:

```
apt remove httpd
```

Description:

Display usage summary for the command (date).

Command:

```
date --help
```

Description:

List active connections to/from system.

Command:

```
ss -tup
```

Description:

List internet services on a system.

Command:

```
ss -tupl
```

Description:

Display all active UNIX listening ports.

Command:

```
netstat  -lx
```

Description:

Display all the active interfaces details.

Command:

```
ifconfig
```

Description:

Display information of all network interfaces.

Command:

```
ifconfig  -a
```

Description:

Compare the contents of two files (1.txt, 2.txt).

Command:

```
diff 1.txt 2.txt
```

Description:

Tells you how many lines, words, and characters there are in a file (1.txt).

Command:

```
wc 1.txt
```

Description:

Compresses file (1.txt), so that it take up much less space.

Command:

```
gzip 1.txt
```

Description:

Uncompresses file (1.txt) compressed by gzip.

Command:

```
gunzip 1.txt
```

Description:

Examine the contents of the file (1.txt).

Command:

```
cat 1.txt
```

Description:

Display calendar.

Command:

```
ncal
```

Description:

Removes the file (1.txt).

Command:

```
rm 1.txt
```

Description:

Rename a file named 1.txt to 0.txt.

Command:

```
mv 1.txt 0.txt
```

Description:

Replace the contents of 0.txt with that of 1.txt.

Command:

```
cp 1.txt 0.txt
```

Description:

Create a empty file (test.txt).

Command:

```
touch test.txt
```

Description:

Print the last 10 lines of a file (1.txt).

Command:

```
tail 1.txt
```

Description:

Print N number of lines from the file (1.txt).

Command:

```
tail  -n N 1.txt
```

Description:

Prints the number of words in a file (1.txt).

Command:

```
wc -w 1.txt
```

Description:

Prints the number of characters from a file (1.txt).

Command:

```
wc -m 1.txt
```

Description:

Prints the length of the longest line in a file (1.txt).

Command:

```
wc -L 1.txt
```

Description:

Print information about usb ports, graphics cards, network adapters etc.

Command:

```
lspci
```

Description:

View contents of a file (1.txt).

Command:

```
less 1.txt
```

Description:

Display calendar (last month, current month, and next month).

Command:

```
cal -3
```

Description:

Compare the contents of three files (1.txt, 2.txt, 3.txt) line by line.

Command:

```
diff3 1.txt 2.txt 3.txt
```

Description:

Compare two files (1.txt, 2.txt) line-by-line.

Command:

```
comm 1.txt 2.txt
```

Description:

Perform byte-by-byte comparison of two files (1.txt, 2.txt).

Command:

```
cmp 1.txt 2.txt
```

Description:

Prints the CRC checksum and byte count for the file "myfiles.txt".

Command:

```
cksum myfiles.txt
```

Description:

Append contents of files (1.txt, 2.txt) into one file (0.txt).

Command:

```
cat 1.txt 2.txt > 0.txt
```

Description:

Append contents of files (1.txt, 2.txt, 3.txt) into one file (0.txt).

Command:

```
sed  r 1.txt 2.txt 3.txt > 0.txt
```

Description:

Append contents of files (1.txt, 2.txt, 3.txt) into one file (0.txt).

Command:

```
sed  h 1.txt 2.txt 3.txt > 0.txt
```

Description:

Append contents of files (1.txt, 2.txt, 3.txt) into one file (0.txt).

Command:

```
sed  -n  p 1.txt 2.txt 3.txt > 0.txt
```

Shortcuts:

```
| ctrl+c            | Halts the current command           |
| ctrl+z            | Stops the current command           |
|                   |                                     |
| ctrl+d            | Logout the current session          |
| ctrl+w            | Erases one word in the current line |
|                   |                                     |
| ctrl+u            | Erases the whole line               |
| ctrl+r            | Type to bring up a recent command   |
```

Description:

Writes contents of a file (0.txt) to output, and prepends each line with line number.

Command:

```
nl 0.txt
```

Description:

Create a empty file (test1.txt) inside a directory (test).

Command:

```
mkdir test
cd test
pwd
touch test1.txt
```

Description:

Gather information about hardware components such as CPU, disks, memory, USB controllers etc.

Command:

```
sudo lshw
```

Description:

Gather information about file system partitions.

Command:

```
sudo fdisk -l
```

Description:

Displays the line (good morning) in which the string (good) is found in the file (1.txt).

Command:

```
grep good 1.txt
```

Description:

Append contents of files (1.txt, 2.txt, 3.txt) into one file (0.txt) using for loop.

Command:

```
for i in {1..3}; do cat "$i.txt" >> 0.txt; done
```

Description:

Search for files (test.txt, test1.txt, test2.txt, test.php, test.html) in a directory as well as its sub-directories.

Command:

```
find test*
```

Description:

Displays status related to a file (1.txt).

Command:

```
stat 1.txt
```

```
###
|     Command      |  Description       |
|:-----------------|-------------:       |
| vi               | Open vi editor     |
| i                | Go to Insert mode  |
|                  |                    |
| a =20; b =64;    |                    |
| print (a + b);   |                    |
| Hit Escape to return to Normal mode.  |
| :w hello.py      | Save text          |
| :q               | Quit               |
| python hello.py  |Print the output:84 |
```

Description:

Download the file (file.txt) from url "http: //website.com/files/file.txt".

Command:

```
wget http://website.com/files/file.txt
```

Description:

Display host's numeric ID in hexadecimal format.

Command:

```
hostid
```

Description:

Display file type of the file (myfiles.txt).

Command:

```
file myfiles.txt
```

Description:

Create a file (myfiles.txt) containing a text (Hello World).

Command:

```
echo 'Hello World' > myfiles.txt
```

Description:

Create a file (myfiles.txt) containing a text (Hello World).

Command:

```
printf 'Hello World' > myfiles.txt
```

Description:

Display IP address of the hostname.

Command:

```
hostname  -i
```

Description:

Add a new line of text to an existing file (1.txt).

Command:

```
echo "Hello world!" >> 1.txt
echo "this is 2nd line text" >> 1.txt
echo "last line!" >> 1.txt
```

Description:

Displays a single line description about a command (cal).

Command:

```
whatis cal
```

```
###
|     Command      |  Description       |
|:-----------------|-------------:      |
| vi               | Open vi editor     |
| i                | Go to Insert mode  |
| Type some text.  |                    |
| Hit Escape to return to Normal mode.  |
| :w test.txt      | Save text          |
| :q               | Quit               |
| :q!              |Quit without saving |


###
|     Command      |  Description          |
|:-----------------|-------------:         |
| vi               | Open vi editor        |
| i                | Go to Insert mode     |
| $name = "Paul";  |                       |
| print "$name";   |                       |
| Hit Escape to return to Normal mode.     |
| :w hello.pl      | Save text             |
| :q               | Quit                  |
| perl hello.pl    |Print the output: Paul |


###
|     Command                |  Description           |
|:---------------------------|-------------:          |
| vi                         | Open vi editor         |
| i                          | Go to Insert mode      |
| echo "What is your name?"  |                        |
| read PERSON                |                        |
| echo "Hello, $PERSON"      |                        |
| Hit Escape to return to Normal mode.                |
| :w hello.sh                | Save text              |
| :q                         | Quit                   |
| sh hello.sh                | Output:                |
|                            | What is your name?     |
|                            | If you enter: Zara Ali |
|                            | Hello, Zara Ali        |
```

Description:

Check the network connectivity between host (your connection) and server (Google server).

Command:

```
ping google.com
```

Description:

Find the location of source/binary file of a command (cal).

Command:

```
whereis cal
```

There are 2 ways to use the command:

- Numeric mode
- Symbolic mode

Numeric mode	**Permission Type**	**Symbolic mode**
0	No Permission	---
1	Execute	--x
2	Write	-w-
3	Execute + Write	-wx
4	Read	r--
5	Read + Execute	r-x
6	Read + Write	rw-
7	Read + Write + Execute	rwx

Name of the shell	Program name	Symbol
Bourne Shell	sh	$
Korn Shell	ksh	$
C Shell	csh	%

Linux Command Cheat Sheet

Command	Description
`ls`	Lists all files and directories in the present working directory
`ls-R`	Lists files in sub-directories as well
`ls-a`	Lists hidden files as well
`ls-al`	Lists files and directories with detailed information like permissions, size, owner, etc.
`cd or cd ~`	Navigate to HOME directory
`cd ..`	Move one level up
`cd`	To change to a particular directory
`cd /`	Move to the root directory
`cat > filename`	Creates a new file
`cat filename`	Displays the file content
`cat file1 file2 > file3`	Joins two files (file1, file2) and stores the output in a new file (file3)
`mv file "new file path"`	Moves the files to the new location
`mv filename new_file_name`	Renames the file to a new filename
`sudo`	Allows regular users to run programs with the security privileges of the super user or root
`rm filename`	Deletes a file
`man`	Gives help information on a command
`history`	Gives a list of all past commands typed in the current terminal session
`clear`	Clears the terminal
`mkdir directory name`	Creates a new directory in the present working directory or a at the specified path
`rmdir`	Deletes a directory
`mv`	Renames a directory

pr -x	Divides the file into x columns
pr -h	Assigns a header to the file
pr -n	Denotes the file with Line Numbers
lp -nc , lpr c	Prints "c" copies of the File
lp-d lp-P	Specifies name of the printer
apt-get	Command used to install and update packages
mail -s 'subject' -c 'cc-address' -b 'bcc-address' 'to-address'	Command to send email
mail -s "Subject" to-address < Filename	Command to send email with attachment

ls-l	to show file type and access permission
r	read permission
w	write permission
x	execute permission
-=	no permission
Chown user	For changing the ownership of a file/directory
Chown user:group filename	change the user as well as group for a file or directory

echo $VARIABLE	To display value of a variable
env	Displays all environment variables
VARIABLE_NAME= variable_value	Create a new variable
Unset	Remove a variable
export Variable=value	To set value of an environment variable

sudo adduser username	To display value of a variable
sudo passwd -l 'username'	Displays all environment variables
sudo userdel -r 'username'	Create a new variable
sudo usermod -a -G GROUPNAME USERNAME	Remove a variable
sudo deluser USER GROUPNAME	To set value of an environment variable
finger	Gives information on all logged in user
finger username	Gives information of a particular user

SSH username@ip-address or hostname	login into a remote Linux machine using SSH
Ping hostname="" or =""	To ping and Analyzing network and host connections

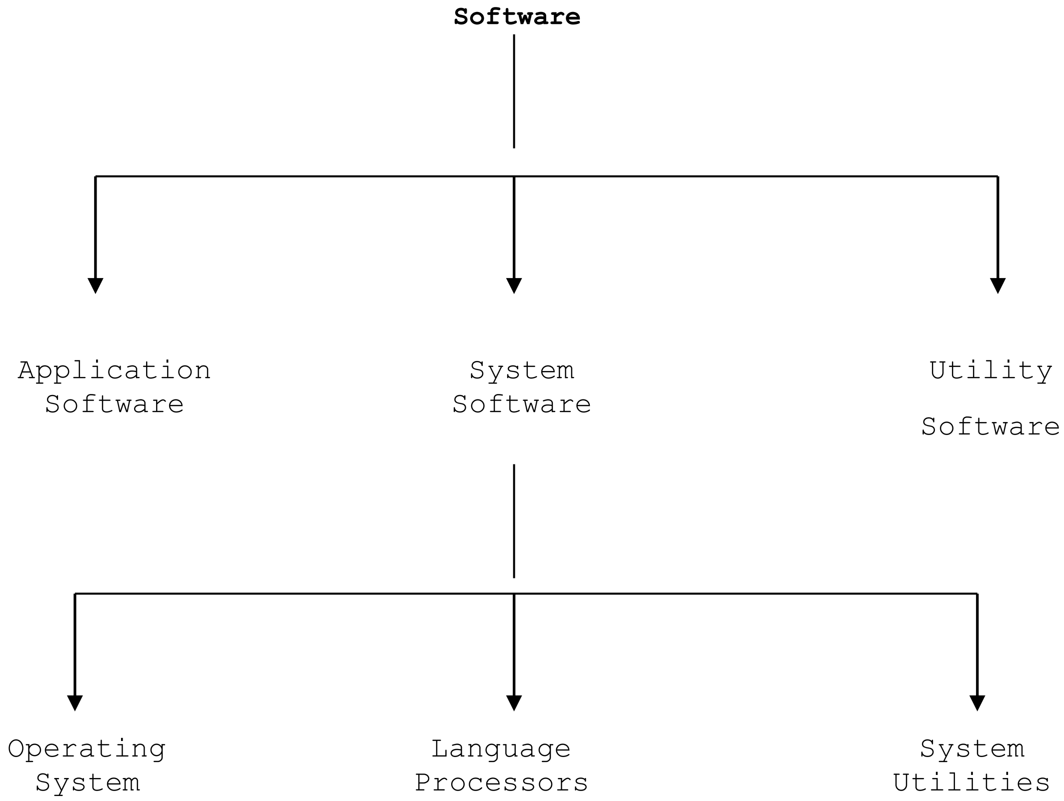
`dir`	Display files in the current directory of a remote computer
`cd "dirname"`	change directory to "dirname" on a remote computer
`put file`	upload 'file' from local to remote computer
`get file`	Download 'file' from remote to local computer
`quit`	Logout

`bg`	To send a process to the background
`fg`	To run a stopped process in the foreground
`top`	Details on all Active Processes
`ps`	Give the status of processes running for a user
`ps PID`	Gives the status of a particular process
`pidof`	Gives the Process ID (PID) of a process
`kill PID`	Kills a process
`nice`	Starts a process with a given priority
`renice`	Changes priority of an already running process
`df`	Gives free hard disk space on your system
`free`	Gives free RAM on your system

`i`	Insert at cursor (goes into insert mode)
`a`	Write after cursor (goes into insert mode)
`A`	Write at the end of line (goes into insert mode)
`ESC`	Terminate insert mode
`u`	Undo last change
`U`	Undo all changes to the entire line
`o`	Open a new line (goes into insert mode)
`dd`	Delete line
`3dd`	Delete 3 lines
`D`	Delete contents of line after the cursor
`C`	Delete contents of a line after the cursor and insert new text. Press ESC key to end insertion.
`dw`	Delete word
`4dw`	Delete 4 words

cw	Change word
x	Delete character at the cursor
r	Replace character
R	Overwrite characters from cursor onward
s	Substitute one character under cursor continue to insert
S	Substitute entire line and begin to insert at the beginning of the line
~	Change case of individual character

Editors for document files	Editors for non-document files
Notepad	Turbo C
WordPad	Turbo C++
MS-WORD	Borland C/C++
	vi editor
	vim editor
	pico
	Emacs



- **Operating system** [DOS, UNIX, LINUX, Windows, Novel NetWare]
- **Language processors** [Interpreter, Compiler, Assembler, Editor]
- **System Utilities** [Loader and Linker]

LAN	**WAN**
Small computer network that covers a building or a campus	Wider computer network that covers a city or the entire globe
High bandwidth	Low bandwidth
Lower delays as they cover smaller distances	Greater delays as they cover far distances
Security is very high	Security is high

Generally, programmers commit three types of errors. They are:

- Syntax errors
- Logical errors
- Run-time errors

Syntax errors:

The syntax of assignment statement in C is:

```
c = a+b;
```

If the above statement is typed without the semicolon at the end, then there will be a ***syntax error*** because of a missing semicolon.

Logical errors: A Mistake in a ***program's source code*** that result in incorrect or unexpected behavior. It is a type of runtime error that may simply produce the wrong output or may cause a program to crash while running.

Run-time errors → Device errors, improper sequencing of constructs, errors in system software, Keypunch errors, incorrect data input.

This program illustrates the Function with arguments but has no return value:

```
#include <stdio.h>

void checkPrimeNumber();

int main()
{
    checkPrimeNumber();    // argument is not passed
    return 0;
}

// return type is void meaning doesn't return any value
void checkPrimeNumber()
{
    int n, i, flag = 0;

    printf("Enter a positive integer: ");
    scanf("%d", &n);

    for(i=2; i <= n/2; ++i)
    {
        if(n%i == 0)
        {
            flag = 1;
        }
    }
    if (flag == 1)
```

```
        printf("%d is not a prime number.", n);
    else
        printf("%d is a prime number.", n);
}
```

This program illustrates the Function with arguments but has a return value:

```
#include <stdio.h>
int getInteger();

int main()
{
    int n, i, flag = 0;

   // no argument is passed
    n = getInteger();

    for(i=2; i<=n/2; ++i)
    {
        if(n%i==0){
            flag = 1;
            break;
        }
    }

    if (flag == 1)
        printf("%d is not a prime number.", n);
    else
        printf("%d is a prime number.", n);

    return 0;
}

// returns integer entered by the user
```

```
int getInteger()
{
    int n;

    printf("Enter a positive integer: ");
    scanf("%d", &n);

    return n;
}
```

- ***Recursion*** is the technique that defines a function in terms of itself. That is, a function which performs a particular task is repeatedly calling itself. The best example of recursively defined function is computing the factorial of a given number.

The Importance of Algorithms

A Computer Program can be viewed as an elaborate algorithm and algorithms are very important in Computer Science for solving a problem -- based on conducting a sequence of specified actions. The best chosen algorithm usually means a small procedure that solves a recurrent problem and makes sure computer will do the given task at best possible manner. In cases where efficiency matters -- a proper algorithm is really vital to be used. An algorithm is important in optimizing a computer program according to the available resources – often play a very significant part in the structure of artificial intelligence, where simple algorithms are used in simple applications, while more complex ones help frame strong artificial intelligence.

You might have an algorithm for getting from office to home, for making a chunk of code that calculates the terms of the Fibonacci sequence, or for finding what you’re looking for in a retail store. Algorithms are the building blocks of computer programs or sequence of unambiguous instructions (the term 'unambiguous' indicates that there is no room for subjective interpretation) that tells how the problem could be addressed and solved -- which is definitely overblown in their importance like road maps for accomplishing a given, well-defined automated reasoning task -- which always have a clear stopping point.

Long division and column addition are examples that everyone is familiar with − even a simple function for adding two numbers is implementation of a particular algorithm. Online grammar checking uses algorithms. Financial computations use algorithms. Robotic field uses algorithms for controlling their robot using algorithms. An encryption algorithm transforms data according to specified actions to protect it. A search engine like Google uses search engine algorithms (such as, takes search strings of keywords as input, searches its associated database for relevant web pages, and returns results). In fact, it is difficult to think of a task performed by your computer that does not use computer rules that are a lot like a recipes (called algorithms).

The use of computer algorithms (step-by-step techniques used for Problem-solving) plays an essential role in space search programs. Scientists have to use enormous calculations, and they are managed by high-end supercomputers, which are enriched with detailed sets of instructions that computers follow to arrive at an answer. Algorithms have applications in many different disciplines from science to math to physics and, of course, computing -- and provide us the most ideal option of accomplishing a task. Here is some ***importance of algorithms*** in computer programming.

- To improve the effectiveness of a computer program: An algorithm (procedure or formula for solving a problem, based on conducting a sequence of specified actions) can be used to improve the speed at which a program executes a problem and has the potential of reducing the time that a program takes to solve a problem.
- Proper usage of resources: The right selection of an algorithm will ensure that a program consumes the least amount of memory. Apart from memory, the algorithm can determine the amount of processing power that is needed by a program.

The algorithm for a child's morning routine could be the following:

- **Step 1:** Wake up and turn off alarm
- **Step 2:** Get dressed
- **Step 3:** Brush teeth
- **Step 4:** Eat breakfast
- **Step 5:** Go to school

The algorithm to add two numbers entered by user would look something like this:

- **Step 1:** Start
- **Step 2:** Declare variables num1, num2 and sum
- **Step 3:** Read values num1 and num2
- **Step 4:** Add num1 and num2 and assign the result to sum

  $$sum \leftarrow num1 + num2$$

- **Step 5:** Display sum
- **Step 6:** Stop

Two of these algorithms accomplish exactly the same goal, but each algorithm does it in completely different way to achieve the required output or to accomplish our task. In computer programming, there are often many different ways – algorithms (any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values as output) − to accomplish any given task. Each algorithm has credits and demerits in different situations. If you have a million integer values between -2147483648 and +2147483647 and you need to sort them, the bin sort is the accurate algorithm to use. If you have a million book titles, the quick sort algorithm might be the best choice. By knowing the toughness and weaknesses of the different algorithms, you pick the best one to accomplish a specific task or to solve a specific problem.

One of the most important aspects of an algorithm is how fast it can manipulate data in various ways, such as inserting a new data item, searching for a particular item or sorting an item. It is often easy to come up with a list of rules to follow in order to solve a problem, but if the algorithm is too slow, it's back to the drawing board. Efficiency of an algorithm depends on its design and implementation. Since every procedure or formula for solving a problem based on conducting a sequence of specified actions -- uses computer resources to run − execution time and internal memory usage are important considerations to analyze an algorithm.

Why Study Algorithms?

Algorithms are the heart of computer science (usually means a procedure or basically instance of logic written in software that solves a recurrent problem of finding an item with specific properties among collection of items or transforming data according to specified actions to protect it), and the subject has countless practical applications as well as intellectual depth that is widely used throughout all areas of information technology including solving a mathematical problem (as of finding the greatest common divisor) in a finite number of steps that often involves repetition of an operation. The word algorithm -- a mathematical concept whose roots date back to 600 AD with invention of the decimal system -- derives from the name of the ninth century Persian mathematician and geographer, **Mohammed ibn-Musa al-Khwarizmi**, who was part of the royal court in Baghdad and who lived from about 780 to 850. On the other hand, it turns out algorithms (widely recognized as the foundation of modern computer coding) have a long and distinguished history stretching back as far as the ***Babylonians***.

Although there is some available body of facts or information about early multiplication algorithms in Egypt (around 1700-2000 BC) the oldest algorithm is widely recognized to be valid or correct to have been found on a set of Babylonian clay tablets that date to around 1600−1800 BC. Their exact significance only came to be revealed or exposed around 1972 when an American computer scientist, mathematician, and professor emeritus at Stanford University

Donald E. Knuth published the first English translations of various Babylonian cuneiform mathematical tablets.

Here are some short extracts from his 1972 manuscript that explain these early algorithms:-

"The calculations described in Babylonian tablets are not merely the solutions to specific individual problems; they are actually general procedures for solving a whole class of problems." - Pages 672 to 673 of "Ancient Babylonian Algorithms".

The wedge-shaped marks on clay tablets also seem to have been an early form of instruction manual:-

"Note also the stereotyped ending, 'This is the procedure,' which is commonly found at the end of each section on a table. Thus the Babylonian procedures are genuine algorithms, and we can commend the Babylonians for developing a nice way to explain an algorithm by example as the algorithm itself was being defined " - Pages 672 to 673 of "Ancient Babylonian Algorithms".

The use of computers, however, has raised the use of algorithms in daily transactions (like accessing an automated teller machine (ATM), booking an air or train or buying something online) to unprecedented levels of real-world problems with solutions requiring advanced algorithms abounds. From Google search to morning routines, algorithms are ubiquitous in our everyday life -- and their use is only likely to grow to break down tasks into chunks that can be solved through specific implementations. Many of the problems, though they may not seem realistic, need the set of well-defined algorithmic knowledge that comes up every day in the real world. By developing a good understanding of a series of logical steps in an algorithmic language, you will be able to choose the right one for a problem and apply it properly. Different algorithms play different roles in programming – and algorithms are used by computer programs where a program –

- Get input data.
- Process it using the complex logics.
- Stop when it finds an answer or some conditions are met.
- Produce the desired output.

To give you a better picture, here is the most common type of algorithms:

- Searching Algorithms
- Sorting Algorithms
- Path finding Algorithms
- Tree and graph based algorithms
- Approximate Algorithms
- Compression Algorithms
- Random Algorithms
- Pattern Matching
- Sequence Finding and a lot more

You only need to define your problem then select the right algorithm to use. The word algorithm may not appear closely connected to kids, but the truth is that -- for kids -- understanding the process of building a step by step method of solving a problem helps them build a strong foundation in logical thinking and problem solving. Here are some problems you can ask your kid to discuss algorithmic solutions with you:

- How do we know if a number is odd or even?
- How do we calculate all of the factors of a number?
- How can we tell if a number is prime?
- Given a list of ten numbers in random order, how can we put them order?

Algorithms has shown it can yield results in all industries — from predicting insurance sales opportunities and generating the millions of search inquiries every day to automating medicine research, optimizing transportation routes, and much more. While algorithms help companies like Master Card and Visa to keep their users' information, such as card number, password, and bank statement safely -- algorithms aren't perfect. They fail and some fail spectacularly. Over the past few years, there have been some serious fails with algorithms, which are the formulas or sets of rules used in digital decision-making processes. Now people are questioning whether we're putting too much trust in the algorithms. When algorithms go bad: Online failures show humans are still needed. Disturbing events at Facebook, Instagram and Amazon reveal the importance of context.

Timeline of algorithms

Medieval Period

- Before – writing about "recipes" (on cooking, rituals, agriculture and other themes)
- c. 1700–2000 BC – Egyptians develop earliest known algorithms for multiplying two numbers
- c. 1600 BC – Babylonians develop earliest known algorithms for factorization and finding square roots
- c. 300 BC – Euclid's algorithm
- c. 200 BC – the Sieve of Eratosthenes
- 263 AD – Gaussian elimination described by Liu Hui
- 628 – Chakravala method described by Brahmagupta
- c. 820 – Al-Khawarizmi described algorithms for solving linear equations and quadratic equations in his *Algebra*; the word *algorithm* comes from his name
- 825 – Al-Khawarizmi described the algorism, algorithms for using the Hindu-Arabic numeral system, in his treatise *On the Calculation with Hindu Numerals*, which was translated into Latin as *Algoritmi de numero Indorum*, where "Algoritmi", the translator's rendition of the author's name gave rise to the word algorithm (Latin *algorithmus*) with a meaning "calculation method"
- c. 850 – cryptanalysis and frequency analysis algorithms developed by Al-Kindi (Alkindus) in *A Manuscript on Deciphering Cryptographic Messages*, which contains algorithms on breaking encryptions and ciphers
- c. 1025 – Ibn al-Haytham (Alhazen), was the first mathematician to derive the formula for the sum of the fourth powers, and in turn, he develops an algorithm for determining the general formula for the sum of any integral powers, which was fundamental to the development of integral calculus

- c. 1400 – Ahmad al-Qalqashandi gives a list of ciphers in his *Subh al-a'sha* which include both substitution and transposition, and for the first time, a cipher with multiple substitutions for each plaintext letter; he also gives an exposition on and worked example of cryptanalysis, including the use of tables of letter frequencies and sets of letters which cannot occur together in one word

Before 1940

- 1540 – Lodovico Ferrari discovered a method to find the roots of a quartic polynomial
- 1545 – Gerolamo Cardano published Cardano's method for finding the roots of a cubic polynomial
- 1614 – John Napier develops method for performing calculations using logarithms
- 1671 – Newton–Raphson method developed by Isaac Newton
- 1690 – Newton–Raphson method independently developed by Joseph Raphson
- 1706 – John Machin develops a quickly converging inverse-tangent series for π and computes π to 100 decimal places
- 1789 – Jurij Vega improves Machin's formula and computes π to 140 decimal places,
- 1805 – FFT-like algorithm known by Carl Friedrich Gauss
- 1842 – Ada Lovelace writes the first algorithm for a computing engine
- 1903 – A Fast Fourier Transform algorithm presented by Carle David Tolmé Runge
- 1926 – Borůvka's algorithm
- 1926 – Primary decomposition algorithm presented by Grete Hermann
- 1934 – Delaunay triangulation developed by Boris Delaunay
- 1936 – Turing machine, an abstract machine developed by Alan Turing, with others developed the modern notion of *algorithm*.

1940s

- 1942 – A Fast Fourier Transform algorithm developed by G.C. Danielson and Cornelius Lanczos
- 1945 – Merge sort developed by John von Neumann
- 1947 – Simplex algorithm developed by George Dantzig

1950s

- 1952 – Huffman coding developed by David A. Huffman
- 1953 – Simulated annealing introduced by Nicholas Metropolis

- 1954 – Radix sort computer algorithm developed by Harold H. Seward
- 1956 – Kruskal's algorithm developed by Joseph Kruskal
- 1957 – Prim's algorithm developed by Robert Prim
- 1957 – Bellman–Ford algorithm developed by Richard E. Bellman and L. R. Ford, Jr.
- 1959 – Dijkstra's algorithm developed by Edsger Dijkstra
- 1959 – Shell sort developed by Donald L. Shell
- 1959 – De Casteljau's algorithm developed by Paul de Casteljau
- 1959 – QR factorization algorithm developed independently by John G.F. Francis and Vera Kublanovskaya

1960s

- 1960 – Karatsuba multiplication
- 1962 – AVL trees
- 1962 – Quicksort developed by C. A. R. Hoare
- 1962 – Ford–Fulkerson algorithm developed by L. R. Ford, Jr. and D. R. Fulkerson
- 1962 – Bresenham's line algorithm developed by Jack E. Bresenham
- 1962 – Gale–Shapley 'stable-marriage' algorithm developed by David Gale and Lloyd Shapley
- 1964 – Heapsort developed by J. W. J. Williams
- 1964 – multigrid methods first proposed by R. P. Fedorenko
- 1965 – Cooley–Tukey algorithm rediscovered by James Cooley and John Tukey
- 1965 – Levenshtein distance developed by Vladimir Levenshtein
- 1965 – Cocke–Younger–Kasami (CYK) algorithm independently developed by Tadao Kasami
- 1965 – Buchberger's algorithm for computing Gröbner bases developed by Bruno Buchberger
- 1966 – Dantzig algorithm for shortest path in a graph with negative edges
- 1967 – Viterbi algorithm proposed by Andrew Viterbi
- 1967 – Cocke–Younger–Kasami (CYK) algorithm independently developed by Daniel H. Younger
- 1968 – A* graph search algorithm described by Peter Hart, Nils Nilsson, and Bertram Raphael
- 1968 – Risch algorithm for indefinite integration developed by Robert Henry Risch
- 1969 – Strassen algorithm for matrix multiplication developed by Volker Strassen

1970s

- 1970 – Dinic's algorithm for computing maximum flow in a flow network by Yefim (Chaim) A. Dinitz
- 1970 – Knuth–Bendix completion algorithm developed by Donald Knuth and Peter B. Bendix
- 1970 – BFGS method of the quasi-Newton class
- 1972 – Graham scan developed by Ronald Graham
- 1972 – Red–black trees and B-trees discovered
- 1973 – RSA encryption algorithm discovered by Clifford Cocks
- 1973 – Jarvis march algorithm developed by R. A. Jarvis
- 1973 – Hopcroft–Karp algorithm developed by John Hopcroft and Richard Karp
- 1974 – Pollard's $p - 1$ algorithm developed by John Pollard
- 1975 – Genetic algorithms popularized by John Holland
- 1975 – Pollard's rho algorithm developed by John Pollard
- 1975 – Aho–Corasick string matching algorithm developed by Alfred V. Aho and Margaret J. Corasick
- 1975 – Cylindrical algebraic decomposition developed by George E. Collins
- 1976 – Salamin–Brent algorithm independently discovered by Eugene Salamin and Richard Brent
- 1976 – Knuth–Morris–Pratt algorithm developed by Donald Knuth and Vaughan Pratt and independently by J. H. Morris
- 1977 – Boyer–Moore string search algorithm for searching the occurrence of a string into another string.
- 1977 – RSA encryption algorithm rediscovered by Ron Rivest, Adi Shamir, and Len Adleman
- 1977 – LZ77 algorithm developed by Abraham Lempel and Jacob Ziv
- 1977 – multigrid methods developed independently by Achi Brandt and Wolfgang Hackbusch
- 1978 – LZ78 algorithm developed from LZ77 by Abraham Lempel and Jacob Ziv
- 1978 – Bruun's algorithm proposed for powers of two by Georg Bruun
- 1979 – Khachiyan's ellipsoid method developed by Leonid Khachiyan
- 1979 – ID3 decision tree algorithm developed by Ross Quinlan

1980s

- 1980 – Brent's Algorithm for cycle detection Richard P. Brendt
- 1981 – Quadratic sieve developed by Carl Pomerance
- 1983 – Simulated annealing developed by S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi
- 1983 – Classification and regression tree (CART) algorithm developed by Leo Breiman, *et al.*
- 1984 – LZW algorithm developed from LZ78 by Terry Welch
- 1984 – Karmarkar's interior-point algorithm developed by Narendra Karmarkar
- 1984 - ACORN_PRNG discovered by Roy Wikramaratna and used privately

- 1985 – Simulated annealing independently developed by V. Cerny
- 1985 - Car–Parrinello molecular dynamics developed by Roberto Car and Michele Parrinello
- 1985 – Splay trees discovered by Sleator and Tarjan
- 1986 – Blum Blum Shub proposed by L. Blum, M. Blum, and M. Shub
- 1986 – Push relabel maximum flow algorithm by Andrew Goldberg and Robert Tarjan
- 1987 – Fast multipole method developed by Leslie Greengard and Vladimir Rokhlin
- 1988 – Special number field sieve developed by John Pollard
- 1989 - ACORN_PRNG published by Roy Wikramaratna

1990s

- 1990 – General number field sieve developed from SNFS by Carl Pomerance, Joe Buhler, Hendrik Lenstra, and Leonard Adleman
- 1991 – Wait-free synchronization developed by Maurice Herlihy
- 1992 – Deutsch–Jozsa algorithm proposed by D. Deutsch and Richard Jozsa
- 1992 – C4.5 algorithm, a descendant of ID3 decision tree algorithm, was developed by Ross Quinlan
- 1993 – Apriori algorithm developed by Rakesh Agrawal and Ramakrishnan Srikant
- 1993 – Karger's algorithm to compute the minimum cut of a connected graph by David Karger
- 1994 – Shor's algorithm developed by Peter Shor
- 1994 – Burrows–Wheeler transform developed by Michael Burrows and David Wheeler
- 1994 – Bootstrap aggregating (bagging) developed by Leo Breiman
- 1995 – AdaBoost algorithm, the first practical boosting algorithm, was introduced by Yoav Freund and Robert Schapire
- 1995 – soft-margin support vector machine algorithm was published by Vladimir Vapnik and Corinna Cortes. It adds a soft-margin idea to the 1992 algorithm by Boser, Nguyon, Vapnik, and is the algorithm that people usually refer to when saying SVM
- 1995 – Ukkonen's algorithm for construction of suffix trees
- 1996 – Bruun's algorithm generalized to arbitrary even composite sizes by H. Murakami
- 1996 – Grover's algorithm developed by Lov K. Grover
- 1996 – RIPEMD-160 developed by Hans Dobbertin, Antoon Bosselaers, and Bart Preneel
- 1997 – Mersenne Twister a pseudo random number generator developed by Makoto Matsumoto and Tajuki Nishimura
- 1998 – PageRank algorithm was published by Larry Page

- 1998 – rsync algorithm developed by Andrew Tridgell
- 1999 – gradient boosting algorithm developed by Jerome H. Friedman
- 1999 – Yarrow algorithm designed by Bruce Schneier, John Kelsey, and Niels Ferguson

2000s

- 2000 – Hyperlink-induced topic search a hyperlink analysis algorithm developed by Jon Kleinberg
- 2001 – Lempel–Ziv–Markov chain algorithm for compression developed by Igor Pavlov
- 2001 – Viola–Jones algorithm for real-time face detection was developed by Paul Viola and Michael Jones.
- 2002 – AKS primality test developed by Manindra Agrawal, Neeraj Kayal and Nitin Saxena
- 2002 – Girvan–Newman algorithm to detect communities in complex systems

The Most Significant Failures When Al Turned Rogue, Causing Disastrous Results

Artificial intelligence (sometimes called machine intelligence) is a part of computer science that emphasizes the creation of intelligent machines with generalized human cognitive abilities that work and reacts like intelligent beings. Artificial intelligence has made a major breakthrough in the processes, including learning (the acquisition of information for using the data), reasoning (using rules to reach definite conclusions) and self-correction − and advancements are accelerating to present a range of new functionality for businesses. But nothing in this world can be made perfect; hence everything accompanies some notable failures and fallacies in them. Here we list some of the significant AI failures from the last decade that hint that the companies need to work harder and keep coming up with better and improved versions of their innovations.

From self-driving cars to industrial robots, all complex real world problems are being solved with applications of intelligence (AI). Artificial intelligence (AI) is progressing rapidly and makes it possible for machines to think like humans and mimic their actions − adjust to new inputs and perform human-like tasks by processing large amounts of data and recognizing patterns in the data. While science fiction often renders AI as robots (a machine − especially one programmable by a computer -- capable of carrying out a complex series of actions without conscious thought or attention) with human-like characteristics, AI can encompass anything from missile guidance to tumor detection to face recognition.

The applications for artificial intelligence are countless and John McCarthy, who coined the term in 1956, defines it as: "the science and engineering of making intelligent machines." The study and design of intelligent agents − where an intelligent agent is a system that becomes aware or conscious of its environment and takes actions which maximizes its chances of success -- can be applied to many sectors and industries including computer science, psychology, philosophy, neuroscience, cognitive science, linguistics, operations research, economics, control theory, probability, optimization, and logic. The simulation of human intelligence in machines is being tested and used in the maintenance or improvement of health industry for dosing drugs and different treatment in patients, and for surgical procedures in the hospital operating room.

A property of machines: the intelligence that the system demonstrates -- today is properly known as Weak Artificial intelligence, in that it is designed to perform a narrow task (such as web searches, control systems, scheduling, data mining, logistics, speech recognition, facial recognition and many others). However, the long-term goal of many technical researchers is to create Strong Artificial intelligence. While Weak Artificial intelligence may outperform humans at whatever its specific task is, like playing games or solving mathematical problems, Strong Artificial intelligence would outsmart humans at nearly every cognitive task.

In little over a decade, Artificial intelligence (a wide-ranging tool that enables people to rethink how we integrate information, analyze data, and use the resulting insights to improve decision making) has made leaps and bounds. Every single day, a new thousand word post showcase the most recent advancement in Artificial intelligence. Being Artificial intelligence has made remarkable breakthroughs, and many scientists dream of creating the Master Algorithm proposed by Pedro Domingos − which can solve all problems envisioned by humans − failure is at the core of human advancement − notable failures are emerging. From self-driving car accidents to Face ID hacks, AI didn't have a perfect year.

The Most Significant Failures When Al Turned Rogue, Causing Disastrous Results:

- 1959: AI designed to be a General Problem Solver failed to solve real world problems.
- 1982: Software designed to make discoveries, discovered how to cheat instead.
- 1983: Nuclear attack early warning system falsely claimed that an attack is taking place.
- 2010: Complex AI stock trading software caused a trillion dollar flash crash.
- 2011: E-Assistant told to "call me an ambulance" began to refer to the user as Ambulance.
- 2013: Object recognition neural networks saw phantom objects in particular noise images.
- 2015: An automated email reply generator created inappropriate responses, such as writing "I love you" to a business colleague.
- 2015: A robot for grabbing auto parts grabbed and killed a man.
- 2015: Image tagging software classified black people as gorillas.
- 2015: Medical AI classified patients with asthma as having a lower risk of dying of pneumonia.
- 2015: Adult content filtering software failed to remove inappropriate content, exposing children to violent and sexual content.
- 2016: AI designed to predict recidivism acted racist.
- 2016: An AI agent exploited a reward signal to win a game without actually completing the game.
- 2016: Video game NPCs (non-player characters, or any character that is not controlled by a human player) designed unauthorized super weapons.
- 2016: AI judged a beauty contest and rated dark-skinned contestants lower.
- 2016: A mall security robot collided with and injured a child.
- 2016: The AI "Alpha Go" lost to a human in a world-championship-level game of "Go."

- 2016: A self-driving car had a deadly accident.
- 2017: Google Translate shows gender bias in Turkish-English translations.
- 2017: Facebook chat bots shut down after developing their own language.
- 2017: Autonomous van in accident on its first day.
- 2017: Google Allo suggested man in turban emoji as response to a gun emoji.
- 2017: Face ID beat by a mask.
- 2017: AI misses the mark with Kentucky Derby predictions.
- 2017: Google Home Minis spied on their owners.
- 2017: Google Home outage causes near 100% failure rate.
- 2017: Facebook allowed ads to be targeted to "Jew Haters".
- 2018: Chinese billionaire's face identified as jaywalker.
- 2018: Uber self-driving car kills a pedestrian.
- 2018: Amazon AI recruiting tool is gender biased.
- 2018: Google Photo confuses skier and mountain.
- 2018: LG robot Cloi gets stagefright at its unveiling.
- 2018: IBM Watson comes up short in healthcare.

While these are only a few instances of failures that have been observed so far, they are pieces of evidence to the fact that Artificial intelligence (the simulation of human intelligence processes by machines, especially computer systems) has the potential to develop a will of its own that may be in conflict with members of the human race. This is definitely a warning about the potential dangers of Artificial intelligence which should be addressed while exploring its potential interests.

"I believe there is no deep difference between what can be achieved by a biological brain and what can be achieved by a computer. It therefore follows that computers can, in theory, emulate human intelligence — and exceed it."

– Stephen Hawking.

Artificial intelligence in general, context remains a challenge. Despite Its Many Failures, why is artificial intelligence important?

- Artificial intelligence automates repetitive learning and discovery through data.
- Artificial intelligence analyzes more and deeper data.
- Artificial intelligence adds intelligence to existing products.
- Artificial intelligence adapts through progressive learning algorithms to let the data do the programming.
- Artificial intelligence gets the most out of data.
- Artificial intelligence achieves unbelievable accuracy through deep neural networks – which was previously impossible. For example, your interactions with Amazon Alexa, Google Search and Google Photos are all based on deep learning – and they keep getting more precise the more we use them.

The threat of AI-charged job loss is spreading (AI and automation will eliminate the most mundane tasks). No matter what industry you're in, AI-powered bots (which can answer common questions and point users to FAQs and knowledge base articles) and software are taking a crack at it. Artificial intelligence seems to be ringing the death sound of a bell for all manner of jobs, tasks, chores and activities. From hospitality, to customer service, to home assistants, no job feels safe. Naturally, this has made people worried about the future. But is Artificial intelligence ready to take over our jobs, or even likely to do so ever? Prevalent AI- charged failures would suggest not.

Timeline of artificial intelligence

Date	Development
Antiquity	Greek myths of Hephaestus and Pygmalion incorporated the idea of intelligent robots (such as Talos) and artificial beings (such as Galatea and Pandora).
	Sacred mechanical statues built in Egypt and Greece were believed to be capable of

	wisdom and emotion. Hermes Trismegistus would write "they have *sensus* and *spiritus* ... by discovering the true nature of the gods, man has been able to reproduce it." Mosaic law prohibits the use of automatons in religion.
10th century BC	Yan Shi presented King Mu of Zhou with mechanical men.
384 BC–322 BC	Aristotle described the syllogism, a method of formal, mechanical thought and theory of knowledge in The Organon.
1st century	Heron of Alexandria created mechanical men and other automatons.
260	Porphyry of Tyros wrote *Isagogê* which categorized knowledge and logic.
~800	Geber developed the Arabic alchemical theory of *Takwin*, the artificial creation of life in the laboratory, up to and including human life.
1206	Al-Jazari created a programmable orchestra of mechanical human beings.
1275	Ramon Llull, Spanish theologian, invents the *Ars Magna*, a tool for combining concepts mechanically, based on an Arabic astrological tool, the Zairja. The method would be developed further by Gottfried Leibniz in the 17th century.
~1500	Paracelsus claimed to have created an artificial man out of magnetism, sperm and alchemy.
~1580	Rabbi Judah Loew ben Bezalel of Prague is said to have invented the Golem, a clay man brought to life.
Early 17th century	René Descartes proposed that bodies of animals are nothing more than complex machines (but that mental phenomena are of a different "substance").

1620	Sir Francis Bacon developed empirical theory of knowledge and introduced inductive logic in his work The New Organon, a play on Aristotle's title The Organon.
1623	Wilhelm Schickard drew a calculating clock on a letter to Kepler. This will be the first of five unsuccessful attempts at designing a *direct entry* calculating clock in the 17th century (including the designs of Tito Burattini, Samuel Morland and René Grillet).
1641	Thomas Hobbes published *Leviathan* and presented a mechanical, combinatorial theory of cognition. He wrote "...for reason is nothing but reckoning".
1642	Blaise Pascal invented the mechanical calculator, the first digital calculating machine.
1672	Gottfried Leibniz improved the earlier machines, making the Stepped Reckoner to do multiplication and division. He also invented the binary numeral system and envisioned a universal calculus of reasoning (alphabet of human thought) by which arguments could be decided mechanically. Leibniz worked on assigning a specific number to each and every object in the world, as a prelude to an algebraic solution to all possible problems.
1726	Jonathan Swift published *Gulliver's Travels*, which includes this description of the Engine, a machine on the island of Laputa: "a Project for improving speculative Knowledge by practical and mechanical Operations " by using this "Contrivance", "the most ignorant Person at a reasonable Charge, and with a little bodily Labour, may write Books in Philosophy, Poetry, Politicks, Law, Mathematicks, and Theology, with the least Assistance from Genius or study." The machine is a parody of *Ars Magna*, one of the inspirations of Gottfried Leibniz' mechanism.
1750	Julien Offray de La Mettrie published *L'Homme Machine*, which argued that human

	thought is strictly mechanical.
1769	Wolfgang von Kempelen built and toured with his chess-playing automaton, The Turk. The Turk was later shown to be a hoax, involving a human chess player.
1818	Mary Shelley published the story of *Frankenstein; or the Modern Prometheus,* a fictional consideration of the ethics of creating sentient beings.
1822–1859	Charles Babbage & Ada Lovelace worked on programmable mechanical calculating machines.
1837	The mathematician Bernard Bolzano made the first modern attempt to formalize semantics.
1854	George Boole set out to "investigate the fundamental laws of those operations of the mind by which reasoning is performed, to give expression to them in the symbolic language of a calculus", inventing Boolean algebra.
1863	Samuel Butler suggested that Darwinian evolution also applies to machines, and speculates that they will one day become conscious and eventually supplant humanity.

Date	**Development**
1913	Bertrand Russell and Alfred North Whitehead published *Principia Mathematica,* which revolutionized formal logic.
1915	Leonardo Torres y Quevedo built a chess automaton, El Ajedrecista, and published speculation about thinking and automata.

1923	Karel Čapek's play *R.U.R. (Rossum's Universal Robots)* opened in London. This is the first use of the word "robot" in English.
1920s and 1930s	Ludwig Wittgenstein and Rudolf Carnap led philosophy into logical analysis of knowledge. Alonzo Church developde Lambda Calculus to investigate computability using recursive functional notation.
1931	Kurt Gödel showed that sufficiently powerful formal systems, if consistent, permit the formulation of true theorems that are unprovable by any theorem-proving machine deriving all possible theorems from the axioms. To do this he had to build a universal, integer-based programming language, which is the reason why he is sometimes called the "father of theoretical computer science".
1940	Edward Condon displays Nimatron, a digital computer that played Nim perfectly.
1941	Konrad Zuse built the first working program-controlled computers.
1943	Warren Sturgis McCulloch and Walter Pitts publish "A Logical Calculus of the Ideas Immanent in Nervous Activity" (1943), laying foundations for artificial neural networks.
	Arturo Rosenblueth, Norbert Wiener and Julian Bigelow coin the term "cybernetics". Wiener's popular book by that name published in 1948.
1945	Game theory which would prove invaluable in the progress of AI was introduced with the 1944 paper, Theory of Games and Economic Behavior by mathematician John von Neumann and economist Oskar Morgenstern.
	Vannevar Bush published *As We May Think* (The Atlantic Monthly, July 1945) a prescient vision of the future in which computers assist humans in many activities.

1948	John von Neumann (quoted by E.T. Jaynes) in response to a comment at a lecture that it was impossible for a machine to think: "You insist that there is something a machine cannot do. If you will tell me *precisely* what it is that a machine cannot do, then I can always make a machine which will do just that!". Von Neumann was presumably alluding to the Church-Turing thesis which states that any effective procedure can be simulated by a (generalized) computer.

Date	Development
1950	Alan Turing proposes the Turing Test as a measure of machine intelligence.
	Claude Shannon published a detailed analysis of chess playing as search.
	Isaac Asimov published his Three Laws of Robotics.
1951	The first working AI programs were written in 1951 to run on the Ferranti Mark 1 machine of the University of Manchester: a checkers-playing program written by Christopher Strachey and a chess-playing program written by Dietrich Prinz.
1952–1962	Arthur Samuel (IBM) wrote the first game-playing program, for checkers (draughts), to achieve sufficient skill to challenge a respectable amateur. His first checkers-playing program was written in 1952, and in 1955 he created a version that learned to play.
1956	The Dartmouth College summer AI conference is organized by John McCarthy, Marvin Minsky, Nathan Rochester of IBM and Claude Shannon. McCarthy coins the term *artificial intelligence* for the conference.
	The first demonstration of the Logic Theorist (LT) written by Allen Newell, J.C. Shaw and Herbert A. Simon (Carnegie Institute of Technology, now Carnegie

	Mellon University or CMU). This is often called the first AI program, though Samuel's checkers program also has a strong claim.
1958	John McCarthy (Massachusetts Institute of Technology or MIT) invented the Lisp programming language.
	Herbert Gelernter and Nathan Rochester (IBM) described a theorem prover in geometry that exploits a semantic model of the domain in the form of diagrams of "typical" cases.
	Teddington Conference on the Mechanization of Thought Processes was held in the UK and among the papers presented were John McCarthy's *Programs with Common Sense*, Oliver Selfridge's *Pandemonium*, and Marvin Minsky's *Some Methods of Heuristic Programming and Artificial Intelligence.*
1959	The General Problem Solver (GPS) was created by Newell, Shaw and Simon while at CMU.
	John McCarthy and Marvin Minsky founded the MIT AI Lab.
Late 1950s, early 1960s	Margaret Masterman and colleagues at University of Cambridge design semantic nets for machine translation.

Date	**Development**
1960s	Ray Solomonoff lays the foundations of a mathematical theory of AI, introducing universal Bayesian methods for inductive inference and prediction.

1960	*Man-Computer Symbiosis* by J.C.R. Licklider.
1961	James Slagle (PhD dissertation, MIT) wrote (in Lisp) the first symbolic integration program, SAINT, which solved calculus problems at the college freshman level.
	In *Minds, Machines and Gödel*, John Lucas denied the possibility of machine intelligence on logical or philosophical grounds. He referred to Kurt Gödel's result of 1931: sufficiently powerful formal systems are either inconsistent or allow for formulating true theorems unprovable by any theorem-proving AI deriving all provable theorems from the axioms. Since humans are able to "see" the truth of such theorems, machines were deemed inferior.
	Unimation's industrial robot Unimate worked on a General Motors automobile assembly line.
1963	Thomas Evans' program, ANALOGY, written as part of his PhD work at MIT, demonstrated that computers can solve the same analogy problems as are given on IQ tests.
	Edward Feigenbaum and Julian Feldman published *Computers and Thought*, the first collection of articles about artificial intelligence.
	Leonard Uhr and Charles Vossler published "A Pattern Recognition Program That Generates, Evaluates, and Adjusts Its Own Operators", which described one of the first machine learning programs that could adaptively acquire and modify features and thereby overcome the limitations of simple perceptrons of Rosenblatt.
1964	Danny Bobrow's dissertation at MIT (technical report #1 from MIT's AI group, Project MAC), shows that computers can understand natural language well enough to solve algebra word problems correctly.
	Bertram Raphael's MIT dissertation on the SIR program demonstrates the power of a logical representation of knowledge for question-answering systems.

1965	Lotfi Zadeh at U.C. Berkeley publishes his first paper introducing fuzzy logic "Fuzzy Sets" (Information and Control 8: 338–353).
	J. Alan Robinson invented a mechanical proof procedure, the Resolution Method, which allowed programs to work efficiently with formal logic as a representation language.
	Joseph Weizenbaum (MIT) built ELIZA, an interactive program that carries on a dialogue in English language on any topic. It was a popular toy at AI centers on the ARPANET when a version that "simulated" the dialogue of a psychotherapist was programmed.
	Edward Feigenbaum initiated Dendral, a ten-year effort to develop software to deduce the molecular structure of organic compounds using scientific instrument data. It was the first expert system.
1966	Ross Quillian (PhD dissertation, Carnegie Inst. of Technology, now CMU) demonstrated semantic nets.
	Machine Intelligence workshop at Edinburgh – the first of an influential annual series organized by Donald Michie and others.
	Negative report on machine translation kills much work in Natural language processing (NLP) for many years.
	Dendral program (Edward Feigenbaum, Joshua Lederberg, Bruce Buchanan, Georgia Sutherland at Stanford University) demonstrated to interpret mass spectra on organic chemical compounds. First successful knowledge-based program for scientific reasoning.
1968	Joel Moses (PhD work at MIT) demonstrated the power of symbolic reasoning for integration problems in the Macsyma program. First successful knowledge-based program in mathematics.

	Richard Greenblatt (programmer) at MIT built a knowledge-based chess-playing program, MacHack, that was good enough to achieve a class-C rating in tournament play.
	Wallace and Boulton's program, Snob (Comp.J. 11(2) 1968), for unsupervised classification (clustering) uses the Bayesian Minimum Message Length criterion, a mathematical realisation of Occam's razor.
1969	Stanford Research Institute (SRI): Shakey the Robot, demonstrated combining animal locomotion, perception and problem solving.
	Roger Schank (Stanford) defined conceptual dependency model for natural language understanding. Later developed (in PhD dissertations at Yale University) for use in story understanding by Robert Wilensky and Wendy Lehnert, and for use in understanding memory by Janet Kolodner.
	Yorick Wilks (Stanford) developed the semantic coherence view of language called Preference Semantics, embodied in the first semantics-driven machine translation program, and the basis of many PhD dissertations since such as Bran Boguraev and David Carter at Cambridge.
	First International Joint Conference on Artificial Intelligence (IJCAI) held at Stanford.
	Marvin Minsky and Seymour Papert publish *Perceptrons*, demonstrating previously unrecognized limits of this feed-forward two-layered structure, and This book is considered by some to mark the beginning of the AI winter of the 1970s, a failure of confidence and funding for AI. Nevertheless, significant progress in the field continued (see below).
	McCarthy and Hayes started the discussion about the frame problem with their essay, "Some Philosophical Problems from the Standpoint of Artificial Intelligence".

Date	Development
Early 1970s	Jane Robinson and Don Walker established an influential Natural Language Processing group at SRI.
1970	Seppo Linnainmaa publishes the reverse mode of automatic differentiation. This method became later known as backpropagation, and is heavily used to train artificial neural networks.
	Jaime Carbonell (Sr.) developed SCHOLAR, an interactive program for computer assisted instruction based on semantic nets as the representation of knowledge.
	Bill Woods described Augmented Transition Networks (ATN's) as a representation for natural language understanding.
	Patrick Winston's PhD program, ARCH, at MIT learned concepts from examples in the world of children's blocks.
1971	Terry Winograd's PhD thesis (MIT) demonstrated the ability of computers to understand English sentences in a restricted world of children's blocks, in a coupling of his language understanding program, SHRDLU, with a robot arm that carried out instructions typed in English.
	Work on the Boyer-Moore theorem prover started in Edinburgh.
1972	Prolog programming language developed by Alain Colmerauer.
	Earl Sacerdoti developed one of the first hierarchical planning programs, ABSTRIPS.
1973	The Assembly Robotics Group at University of Edinburgh builds Freddy Robot, capable of

	using visual perception to locate and assemble models. (See Edinburgh *Freddy* Assembly Robot: a versatile computer-controlled assembly system.)
	The Lighthill report gives a largely negative verdict on AI research in Great Britain and forms the basis for the decision by the British government to discontinue support for AI research in all but two universities.
1974	Ted Shortliffe's PhD dissertation on the MYCIN program (Stanford) demonstrated a very practical rule-based approach to medical diagnoses, even in the presence of uncertainty. While it borrowed from DENDRAL, its own contributions strongly influenced the future of expert system development, especially commercial systems.
1975	Earl Sacerdoti developed techniques of partial-order planning in his NOAH system, replacing the previous paradigm of search among state space descriptions. NOAH was applied at SRI International to interactively diagnose and repair electromechanical systems.
	Austin Tate developed the Nonlin hierarchical planning system able to search a space of partial plans characterised as alternative approaches to the underlying goal structure of the plan.
	Marvin Minsky published his widely read and influential article on Frames as a representation of knowledge, in which many ideas about schemas and semantic links are brought together.
	The Meta-Dendral learning program produced new results in chemistry (some rules of mass spectrometry) the first scientific discoveries by a computer to be published in a refereed journal.
Mid-1970s	Barbara Grosz (SRI) established limits to traditional AI approaches to discourse modeling. Subsequent work by Grosz, Bonnie Webber and Candace Sidner developed the notion of "centering", used in establishing focus of discourse and anaphoric references in Natural

	language processing. David Marr and MIT colleagues describe the "primal sketch" and its role in visual perception.
1976	Douglas Lenat's AM program (Stanford PhD dissertation) demonstrated the discovery model (loosely guided search for interesting conjectures). Randall Davis demonstrated the power of meta-level reasoning in his PhD dissertation at Stanford.
1978	Tom Mitchell, at Stanford, invented the concept of Version spaces for describing the search space of a concept formation program. Herbert A. Simon wins the Nobel Prize in Economics for his theory of bounded rationality, one of the cornerstones of AI known as "satisficing". The MOLGEN program, written at Stanford by Mark Stefik and Peter Friedland, demonstrated that an object-oriented programming representation of knowledge can be used to plan gene-cloning experiments.
1979	Bill VanMelle's PhD dissertation at Stanford demonstrated the generality of MYCIN's representation of knowledge and style of reasoning in his EMYCIN program, the model for many commercial expert system "shells". Jack Myers and Harry Pople at University of Pittsburgh developed INTERNIST, a knowledge-based medical diagnosis program based on Dr. Myers' clinical knowledge. Cordell Green, David Barstow, Elaine Kant and others at Stanford demonstrated the CHI system for automatic programming.

	The Stanford Cart, built by Hans Moravec, becomes the first computer-controlled, autonomous vehicle when it successfully traverses a chair-filled room and circumnavigates the Stanford AI Lab.
	BKG, a backgammon program written by Hans Berliner at CMU, defeats the reigning world champion (in part via luck).
	Drew McDermott and Jon Doyle at MIT, and John McCarthy at Stanford begin publishing work on non-monotonic logics and formal aspects of truth maintenance.
Late 1970s	Stanford's SUMEX-AIM resource, headed by Ed Feigenbaum and Joshua Lederberg, demonstrates the power of the ARPAnet for scientific collaboration.

Date	**Development**
1980s	Lisp machines developed and marketed. First expert system shells and commercial applications.
1980	First National Conference of the American Association for Artificial Intelligence (AAAI) held at Stanford.
1981	Danny Hillis designs the connection machine, which utilizes Parallel computing to bring new power to AI, and to computation in general. (Later founds Thinking Machines Corporation)
1982	The Fifth Generation Computer Systems project (FGCS), an initiative by Japan's Ministry of International Trade and Industry, begun in 1982, to create a "fifth generation computer" which was supposed to perform much calculation utilizing massive parallelism.

1983	John Laird and Paul Rosenbloom, working with Allen Newell, complete CMU dissertations on Soar (program).
	James F. Allen invents the Interval Calculus, the first widely used formalization of temporal events.
Mid-1980s	Neural Networks become widely used with the Backpropagation algorithm, also known as the reverse mode of automatic differentiation published by Seppo Linnainmaa in 1970 and applied to neural networks by Paul Werbos.
1985	The autonomous drawing program, AARON, created by Harold Cohen, is demonstrated at the AAAI National Conference (based on more than a decade of work, and with subsequent work showing major developments).
1986	The team of Ernst Dickmanns at Bundeswehr University of Munich builds the first robot cars, driving up to 55 mph on empty streets.
	Barbara Grosz and Candace Sidner create the first computation model of discourse, establishing the field of research.
1987	Marvin Minsky published *The Society of Mind*, a theoretical description of the mind as a collection of cooperating agents. He had been lecturing on the idea for years before the book came out (c.f. Doyle 1983).
	Around the same time, Rodney Brooks introduced the subsumption architecture and behavior-based robotics as a more minimalist modular model of natural intelligence; Nouvelle AI.
	Commercial launch of generation 2.0 of Alacrity by Alacritous Inc./Allstar Advice Inc. Toronto, the first commercial strategic and managerial advisory system. The system was based upon a forward-chaining, self-developed expert system with 3,000 rules about the evolution of markets and competitive strategies and co-authored by Alistair Davidson and

	Mary Chung, founders of the firm with the underlying engine developed by Paul Tarvydas. The Alacrity system also included a small financial expert system that interpreted financial statements and models.
1989	The development of metal–oxide–semiconductor (MOS) very-large-scale integration (VLSI), in the form of complementary MOS (CMOS) technology, enabled the development of practical artificial neural network (ANN) technology in the 1980s. A landmark publication in the field was the 1989 book *Analog VLSI Implementation of Neural Systems* by Carver A. Mead and Mohammed Ismail.
	Dean Pomerleau at CMU creates ALVINN (An Autonomous Land Vehicle in a Neural Network).

Date	**Development**
1990s	Major advances in all areas of AI, with significant demonstrations in machine learning, intelligent tutoring, case-based reasoning, multi-agent planning, scheduling, uncertain reasoning, data mining, natural language understanding and translation, vision, virtual reality, games, and other topics.
Early 1990s	TD-Gammon, a backgammon program written by Gerry Tesauro, demonstrates that reinforcement (learning) is powerful enough to create a championship-level game-playing program by competing favorably with world-class players.
1991	DART scheduling application deployed in the first Gulf War paid back DARPA's investment of 30 years in AI research.
1992	Carol Stoker and NASA Ames robotics team explore marine life in Antarctica with an undersea robot Telepresence ROV operated from the ice near McMurdo Bay, Antarctica and remotely via satellite link from Moffett Field, California.

1993	Ian Horswill extended behavior-based robotics by creating Polly, the first robot to navigate using vision and operate at animal-like speeds (1 meter/second).
	Rodney Brooks, Lynn Andrea Stein and Cynthia Breazeal started the widely publicized MIT Cog project with numerous collaborators, in an attempt to build a humanoid robot child in just five years.
	ISX corporation wins "DARPA contractor of the year" for the Dynamic Analysis and Replanning Tool (DART) which reportedly repaid the US government's entire investment in AI research since the 1950s.
1994	Lotfi Zadeh at U.C. Berkeley creates "soft computing" and builds a world network of research with a fusion of neural science and neural net systems, fuzzy set theory and fuzzy systems, evolutionary algorithms, genetic programming, and chaos theory and chaotic systems ("Fuzzy Logic, Neural Networks, and Soft Computing," Communications of the ACM, March 1994, Vol. 37 No. 3, pages 77-84).
	With passengers on board, the twin robot cars VaMP and VITA-2 of Ernst Dickmanns and Daimler-Benz drive more than one thousand kilometers on a Paris three-lane highway in standard heavy traffic at speeds up to 130 km/h. They demonstrate autonomous driving in free lanes, convoy driving, and lane changes left and right with autonomous passing of other cars.
	English draughts (checkers) world champion Tinsley resigned a match against computer program Chinook. Chinook defeated 2nd highest rated player, Lafferty. Chinook won the USA National Tournament by the widest margin ever.
	Cindy Mason at NASA organizes the First AAAI Workshop on AI and the Environment.
1995	Cindy Mason at NASA organizes the First International IJCAI Workshop on AI and the

	Environment.
	"No Hands Across America": A semi-autonomous car drove coast-to-coast across the United States with computer-controlled steering for 2,797 miles (4,501 km) of the 2,849 miles (4,585 km). Throttle and brakes were controlled by a human driver.
	One of Ernst Dickmanns' robot cars (with robot-controlled throttle and brakes) drove more than 1000 miles from Munich to Copenhagen and back, in traffic, at up to 120 mph, occasionally executing maneuvers to pass other cars (only in a few critical situations a safety driver took over). Active vision was used to deal with rapidly changing street scenes.
1997	The Deep Blue chess machine (IBM) defeats the (then) world chess champion, Garry Kasparov.
	First official RoboCup football (soccer) match featuring table-top matches with 40 teams of interacting robots and over 5000 spectators.
	Computer Othello program Logistello defeated the world champion Takeshi Murakami with a score of 6–0.
1998	Tiger Electronics' Furby is released, and becomes the first successful attempt at producing a type of A.I to reach a domestic environment.
	Tim Berners-Lee published his Semantic Web Road map paper.
	Ulises Cortés and Miquel Sànchez-Marrè organize the first Environment and AI Workshop in Europe ECAI, "Binding Environmental Sciences and Artificial Intelligence."
	Leslie P. Kaelbling, Michael Littman, and Anthony Cassandra introduce POMDPs and a scalable method for solving them to the AI community, jumpstarting widespread use in

	robotics and automated planning and scheduling
1999	Sony introduces an improved domestic robot similar to a Furby, the AIBO becomes one of the first artificially intelligent "pets" that is also autonomous.
Late 1990s	Web crawlers and other AI-based information extraction programs become essential in widespread use of the World Wide Web.
	Demonstration of an Intelligent room and Emotional Agents at MIT's AI Lab.
	Initiation of work on the Oxygen architecture, which connects mobile and stationary computers in an adaptive network.

Date	**Development**
2000	Interactive robopets ("smart toys") become commercially available, realizing the vision of the 18th century novelty toy makers.
	Cynthia Breazeal at MIT publishes her dissertation on Sociable machines, describing Kismet (robot), with a face that expresses emotions.
	The Nomad robot explores remote regions of Antarctica looking for meteorite samples.
2002	iRobot's Roomba autonomously vacuums the floor while navigating and avoiding obstacles.
2004	OWL Web Ontology Language W3C Recommendation (10 February 2004).
	DARPA introduces the DARPA Grand Challenge requiring competitors to produce autonomous vehicles for prize money.
	NASA's robotic exploration rovers Spirit and Opportunity autonomously navigate the surface of Mars.

2005	Honda's ASIMO robot, an artificially intelligent humanoid robot, is able to walk as fast as a human, delivering trays to customers in restaurant settings.
	Recommendation technology based on tracking web activity or media usage brings AI to marketing. See TiVo Suggestions.
	Blue Brain is born, a project to simulate the brain at molecular detail.
2006	The Dartmouth Artificial Intelligence Conference: The Next 50 Years (AI@50) AI@50 (14–16 July 2006)
2007	Philosophical Transactions of the Royal Society, B – Biology, one of the world's oldest scientific journals, puts out a special issue on using AI to understand biological intelligence, titled *Models of Natural Action Selection*
	Checkers is solved by a team of researchers at the University of Alberta.
	DARPA launches the Urban Challenge for autonomous cars to obey traffic rules and operate in an urban environment.
2008	Cynthia Mason at Stanford presents her idea on Artificial Compassionate Intelligence, in her paper on "Giving Robots Compassion".
2009	Google builds autonomous car.

Date	**Development**
2010	Microsoft launched Kinect for Xbox 360, the first gaming device to track human body movement, using just a 3D camera and infra-red detection, enabling users to play their Xbox 360 wirelessly. The award-winning machine learning for human motion capture technology for this device was developed by the Computer Vision group at Microsoft Research, Cambridge.
2011	Mary Lou Maher and Doug Fisher organize the First AAAI Workshop on AI and

	Sustainability.
	IBM's Watson computer defeated television game show Jeopardy! champions Rutter and Jennings.
2011–2014	Apple's Siri (2011), Google's Google Now (2012) and Microsoft's Cortana (2014) are smartphone apps that use natural language to answer questions, make recommendations and perform actions.
2013	Robot HRP-2 built by SCHAFT Inc of Japan, a subsidiary of Google, defeats 15 teams to win DARPA's Robotics Challenge Trials. HRP-2 scored 27 out of 32 points in 8 tasks needed in disaster response. Tasks are drive a vehicle, walk over debris, climb a ladder, remove debris, walk through doors, cut through a wall, close valves and connect a hose.
	NEIL, the Never Ending Image Learner, is released at Carnegie Mellon University to constantly compare and analyze relationships between different images.
2015	An open letter to ban development and use of autonomous weapons signed by Hawking, Musk, Wozniak and 3,000 researchers in AI and robotics.
	Google DeepMind's AlphaGo (version: Fan) defeated 3 time European Go champion 2 dan professional Fan Hui by 5 games to 0.
2016	Google DeepMind's AlphaGo (version: Lee) defeated Lee Sedol 4–1. Lee Sedol is a 9 dan professional Korean Go champion who won 27 major tournaments from 2002 to 2016.
2017	Asilomar Conference on Beneficial AI was held, to discuss AI ethics and how to bring about beneficial AI while avoiding the existential risk from artificial general intelligence.
	Deepstack is the first published algorithm to beat human players in imperfect information games, as shown with statistical significance on heads-up no-limit poker. Soon after, the

	poker AI Libratus by different research group individually defeated each of its 4 human opponents—among the best players in the world—at an exceptionally high aggregated winrate, over a statistically significant sample. In contrast to Chess and Go, Poker is an imperfect information game.
	Google DeepMind's AlphaGo (version: Master) won 60–0 rounds on two public Go websites including 3 wins against world Go champion Ke Jie.
	A propositional logic boolean satisfiability problem (SAT) solver proves a long-standing mathematical conjecture on Pythagorean triples over the set of integers. The initial proof, 200TB long, was checked by two independent certified automatic proof checkers.
	An OpenAI-machined learned bot played at The International 2017 *Dota 2* tournament in August 2017. It won during a 1v1 demonstration game against professional *Dota 2* player Dendi.
	Google DeepMind revealed that AlphaGo Zero—an improved version of AlphaGo—displayed significant performance gains while using far fewer tensor processing units (as compared to AlphaGo Lee; it used same amount of TPU's as AlphaGo Master). Unlike previous versions, which learned the game by observing millions of human moves, AlphaGo Zero learned by playing only against itself. The system then defeated AlphaGo Lee 100 games to zero, and defeated AlphaGo Master 89 to 11. Although unsupervised learning is a step forward, much has yet to be learned about general intelligence. AlphaZero masters chess in 4 hours, defeating the best chess engine, StockFish 8. AlphaZero won 28 out of 100 games, and the remaining 72 games ended in a draw.
2018	Alibaba language processing AI outscores top humans at a Stanford University reading and comprehension test, scoring 82.44 against 82.304 on a set of 100,000 questions.
	The European Lab for Learning and Intelligent Systems (*aka* Ellis) proposed as a pan-European competitor to American AI efforts, with the aim of staving off a brain drain of

	talent, along the lines of CERN after World War II.
	Announcement of Google Duplex, a service to allow an AI assistant to book appointments over the phone. The LA Times judges the AI's voice to be a "nearly flawless" imitation of human-sounding speech.

DevOps (a set of software development practices that combines software development (Dev) and information technology operations (Ops) to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives) is becoming the standard way of working for Enterprises. Among the few powerful trends we had experienced in the recent times, one is undoubtedly the adoption of DevOps practices – and adoption of DevOps within the organization is rising on a broader scale, and Enterprises are trending toward it. DevOps builds upon best practices to help drive enterprise performance in modernizing environments. It offers organizations a new way to move the business forward and turn technology into a strategic advantage. An increasing number of businesses recognize the power that DevOps can bring a natural extension for Agile and continuous delivery approaches.

"At its essence, DevOps is a culture, a practice, a philosophy."

DevOps expertise is in high demand. Job postings with "DevOps" in a title or keyword are sprouting up everywhere. DevOps is an enterprise software development phrase emerging from combination of IT teams, process and products to enable the continuous delivery of value to end users. It is a firm bond between development and operations that emphasizes a shift in mindset, better collaboration, and tighter integration and aims to create a culture and environment where building, testing, and releasing software can happen rapidly, often, and more reliably, so organizations can solve critical issues quickly, and better serve their customers and compete more effectively in the market.

What is DevOps?

"A software development method formed out of a fundamental need that stresses communication, collaboration and integration between software developers and IT professionals." DevOps could be explained simply as operations working together with engineers to get things done faster in an automated and repeatable way.

History of DevOps

At the 2008 Agile Toronto conference, Andrew Shafer and Patrick Debois introduced the term in their talk on "Agile Infrastructure". Since 2009, the DevOps term has been steadily promoted based on a simple philosophy — business works best when efforts being coordinated and collaborative — and brought into more mainstream usage through a series of "DevOpsDays", which started in Belgium and has now spread into Web-enabled sphere to resolve the conflict between the software developers and the operations teams when it comes to getting great work done quickly. In recent years, more tangential DevOps initiatives have also evolved, such as OpsDev, WinOps, and BizDevOps to encourage the communication between software developers and IT Operations to increase the speed at which applications being delivered.

Benefits of DevOps

The technical benefits include:

- Continuous software delivery
- Less complexity to manage

- Faster resolution of problems The cultural benefits include:
- More productive teams
- Higher employee engagement
- Greater professional development opportunities

The business benefits include:

- Faster delivery of features
- More stable operating environments
- Improved communication and collaboration
- More time to innovate (and not fix / keep up)

Features of DevOps

- Source control: Software developers need to safely store their code and keep track of source- code history and versions. For this reason alone, source control is of critical importance.
- Issue tracking system: An issue tracking system allows everyone involved to track current issues, estimates, and deadlines.
- Build system: The build system supports continuous integration by building the software, running unit and integration tests, deploying to the integration environment, and performing any other automated checks defined for new versions of the software.
- Monitoring system: Monitoring systems continuously track all autonomous systems within the DevOps environment, notifying necessary maintenance staff if a system failure occurs.
- Communications system: The constant exchange of information is important so email, wiki, and a real-time chat system being enabled for effective communication and collaboration among all members of the project team.

- Integration environment: The integration environment hosts all the virtual machines that make up our DevOps environment
- Code review system: To make sure software quality, every line of code being reviewed by an experienced developer. The practice of reviewing code also accelerates career growth and learning.
- Documentation system: Regrettably, documentation often remains an afterthought in production software projects. To ensure that documentation being written throughout the project, an automated system being developed to allow developers to write documentation easily, along with source code.

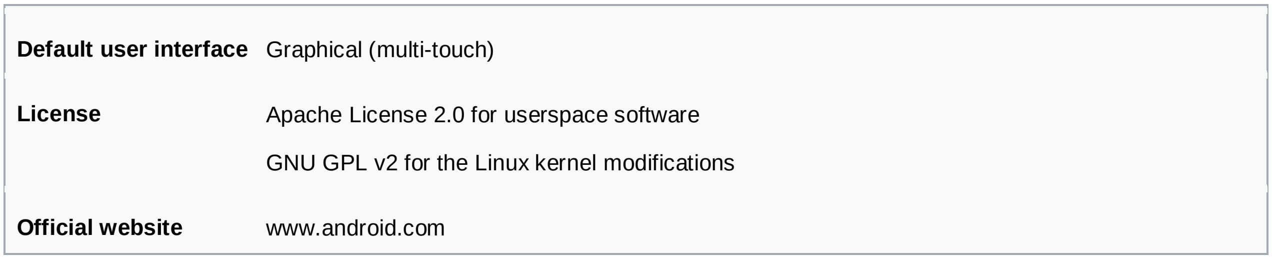
DevOps Goals

- Improved deployment frequency
- To make faster time to market
- Less failure rate to new releases
- Short lead time between fixes
- Improve mean time to recovery

Is DevOps a good career?

DevOps practitioners are among the highest paid IT professionals today, and the market demand for them is growing rapidly because organizations using DevOps practices are overwhelmingly high-functioning to deliver IT services that offer value to the business. According to a study on the application economy and the role of DevOps, 88% of enterprise IT organizations and LOB (line of business) executives already have planned to adopt DevOps sometime within the next five years to accelerate delivery of apps and offer customers with higher-quality software. In the last two years, listings for DevOps jobs at Indeed.com increased 75 percent. On LinkedIn.com, mentions of DevOps as a skill increased 50 percent. In a recent survey by Puppetlabs, half of their 4,000-plus respondents (in more than 90 countries) said their companies consider DevOps skills when hiring.

Android	
Developer	Various (mostly Google and the Open Handset Alliance)
Written in	Java (UI), C (core), C++ and others
OS family	Unix-like (Modified Linux kernel)
Working state	Current
Source model	Open source (most devices include proprietary components, such as Google Play)
Initial release	September 23, 2008; 11 years ago
Latest release	Android 10 / September 3, 2019; 9 months ago
Latest preview	Android 11 Developer Preview 4 (RPP4.200409.015) / May 6, 2020; 38 days ago
Repository	android.googlesource.com
Marketing target	Smartphones, tablet computers, smart TVs (Android TV), Android Auto and smartwatches (Wear OS)
Available in	100+ languages
Update method	Over-the-air
Package manager	APK-based
Platforms	32- and 64-bit ARM, x86 and x86-64
Kernel type	Linux kernel
Userland	Bionic libc, mksh shell, Toybox as core utilities (beginning with Android 6.0)



Default user interface	Graphical (multi-touch)
License	Apache License 2.0 for userspace software GNU GPL v2 for the Linux kernel modifications
Official website	www.android.com

"I think right now it's a battle for the mindshare of developers and for the mindshare of customers, and right now iPhone and Android are winning that battle."

− Steve Jobs

Android application development is one of the hottest topics in the present time. To be up-to-date with the latest trends in mobile application development, one can perceive by chance or unexpectedly a plethora of tech blogs all over the internet. Contemplating Android application development is a great choice as per current market scenario and importance of Android application development for businesses of today is expanding itself, to wearable, automobiles and other areas. Applications like WhatsApp, Facebook, Twitter, Amazon etc. have brought the world around us in our handset. In a statistical study that spans the America, Europe, Asia, and the Middle East, GlobalWebIndex reports that Android tablets outnumber Apple iPad by more than 34 million and has now garnered the interest of a million smart phone users and it powers hundreds of millions of mobile devices in more than 190 countries of the world.

More than a million applications are available for download at the digital distribution platform operated by Google (double the number of apps that were available in the last few years). And more than 9 million developers write code using Java, XML (the languages that empowers an array of software intended for mobile devices that features an operating system, core applications and middleware). With the increase in the number of Android based smartphones (the devices that we started to use just for the communication purpose (i.e. for talking and messaging), abruptly became the most powerful and dependable source of our day-to-day living) and owing to popularity of android and access of internet over mobiles, people using android smart-phones demand for new Android applications, this in turn creates an outstanding career in technology innovation (to push the boundaries of hardware and software forward to bring new capabilities to users and developers) and a demand for better applications and update for existing one.

The Mobile Application Development is the future of Software Development and Android is on the path of proving the same - according to Google's Eric Schmidt. Companies like Nokia, BlackBerry, Samsung, HTC, Motorola, Google and many others are going wild with their innovations to alter the software applications according to their requirements to get in touch with millions of users all over the world including their potential customer and the global client base. This adds a big sign of scope for the Android market would be beaming with lots of opportunities in the nearby future.

Introduction

Android is the world's most popular open source mobile operating system (OS) based on the Linux Kernel − which run on 53 percent of all smartphones in the United States and on 80 percent of all smartphones worldwide − developed by Android Incorporation (a Palo Alto-based startup company, founded in 2003) and later after acquired by and further advanced by coalition of hardware, software and telecommunications companies i.e., open hand set alliance (a group of 84 technology and mobile companies including Dell, Motorola, Samsung Electronics, Sony, Intel, LG Electronics, Qualcomm, Broadcom, HTC, Sprint, Texas Instruments and Japanese wireless carriers KDDI and NTT DoCoMo etc.) − led by Google − designed primarily for touchscreen mobile devices such as smartphones and tablet computers. But now this technology is growing at such a rapid pace that it is going to hit the markets of Television, Cars and Wrist Watches very soon too.

Android Architecture

1. Linux Kernel

What is a Kernel? The basic layer is the Linux kernel. The whole Android OS built on top of the Linux Kernel with some further architectural changes made by Google. It is the core part of the Android Operating System that acts as an abstraction layer between the hardware and the rest of the software stack – which consists of drivers (i.e., a well-defined set of instructions – what we call programs or software written in C language that installed into mobile phones and stored in the form of files in the phone) – that tells your mobile phone how to communicate with its hardware components such as camera, display etc. – without which keypad, Bluetooth, Audio, Wi-Fi, Camera won't work properly and it is responsible for:

- **Inter Process Communication** − A Mechanism which allows applications running in different processes to share data and communicate with each other i.e., a mechanism which allows an application running in a process to send requests and receive responses from an application running in another process.
- **Power Management** (conserves power in the cost of performance and holds the device not to get to sleep state).
- **Memory Management** (make the best or most effective use of memory).

Android uses the Linux Kernel for all its core functionality such as Memory management, process management, networking, security settings etc.

2. Libraries

The next layer is the Android's native libraries. It is this layer that enables the device to handle different types of data. These libraries are a Collection of pre-written non-volatile data (written in C / C++ language) and pre-compiled programming codes – which support the well-functioning of android operating system.

Some of the important native libraries include the following:

- Surface Manager / Screen Manager that supports the display screen.
- OpenGL (Open Graphics Library) that supports 3 dimensional graphics.
- SGL (Scalable Graphics Library) that supports 2 dimensional graphics.
- Media Framework that supports recording and playback of audio and video and image formats (MP3, JPG, JPEG, PNG, GIF etc.)
- Free Type that is responsible for font support (i.e., font size, color etc.)
- SSL (Secured Sockets layer) / TLS (Transport Layer Security) that is responsible for internet security and support network applications.
- WebKit that supports the display of web pages (i.e., supports inbuilt browser)
- SQLite that is responsible for storage of user data.
- Bionic is the standard C library that supports embedded Linux-based devices in mobile phones.

3. Android Run Time (ART)

Android Runtime consists of Core Java libraries and Dalvik Virtual machine.

- Java Core Libraries that consists of Java packages that enable Android application developers to write Android applications using standard Java programming language.
- DVM (Dalvik Virtual Machine) that is responsible to run android application.

4. Application Frame Work

Software Frame work (written in Java language) that supports the features of android applications and manage the basic functions of phone like resource management, voice call management etc.

Important blocks of Application framework are:

- Content Provider that enable applications to get access data from other applications (such as Contacts), or to share their own data.

- Notifications Manager that enables all applications to display custom alerts in the status bar.
- Activity Manager that manages the life-cycle of applications and provides a common navigation back stack.
- Window Manager that organizes the display screen for the application.
- Location Manager that provides the periodic updates of the geographical location of the mobile device using GPS (Global Positioning System which is a satellite-based navigation system) or cell tower.
- View Manager that manages the Application User Interface.
- Package Manager that provides information about the list of installed apps in Android Mobile Device.
- Telephony Manager that provides information about the Telephony Services (such as Phone Network, SIM Serial Number, IMEI Number etc.)
- XMPP (Extensible Messaging and Presence Protocol) that supports Online Chat Application (like Yahoo Messenger etc.)
- Resource Manager that manages the various types of resources we use in our Application and provides access to non-code resources such as localized strings, graphics, and layout files.

5. Applications

Applications are the top layer in the Android architecture. Examples of such applications are:

- SMS client app
- Dialer
- Web browser
- Contact manager
- Facebook
- WhatsApp

Android Application Development Tools and IDE's:

- Android SDK (Software Development Kit) - It contains debugger, libraries, emulator, sample code, documentation and tutorials.
- Android Studio by Google (official IDE for developing Android Apps)
- Eclipse IDE using ADT plugin
- IntelliJ IDEA IDE
- NetBeans IDE

Usage of mobile phones in India has rapidly increased from the past year and counting is still on. Out of the six billion smart phone devices in the world, close to one billion is being used in India. This comes to about 70% of our current population of India. Lots and lots of startups and other Mobile Application Development industries in India are considering Android Application Development as one of the best remunerative business opportunities. Scope of Android App Development in India is huge since every website or company in India needs its own android app (especially if it is providing a web-based service) to make their business plan into action and for capture their services in phone.

The bright future of the App Development in India can better understand with this one example. The telecommunications companies such as idea, Vodafone, MobikWik, FreeRecharge, Aircel and other cellular depends on the third-party app like, Paytm or free charge for the recharge. Thus they are making their own apps to earn direct profit from it and this is the golden opportunity for the Android Developers. In essence, India considered as a country with several globally recognized IT hubs and Android is a choice at the best for exploration in India.

Benefits of Choosing Android Application Development:

i. Android is Open Source
ii. Adaptable User Interface
iii. Massive Mobile App Market

- Google PlayStore - contains more than 2.7 million android apps
- Amazon Appstore - contains 800,000+ apps
- Aptoide - contains more than 750,000 apps
- 1Mobile Market - contains more than 800,000 apps
- Opera Mobile Store - contains more than 300,000 apps
- Mobango-contains over 100k mobile apps
- GetJar - contains over 850,000 apps

Pseudo code → neither an algorithm nor a program.

Advantages:

Easy to read, understand and modify.

Example:

Pseudo code to perform the basic arithmetic operations

```
read n1, n2
sum = n1 + n2
diff = n1 − n2
mult = n1 * n2
quot = n1 / n2
print sum, diff, mult, quot
end
```

Java Programming Language	
Paradigm	Multi-paradigm: generic, object-oriented (class-based), imperative, reflective
Designed by	James Gosling
Developer	Sun Microsystems
First appeared	May 23, 1995; 25 years ago
Stable release	Java SE 14 / March 17, 2020; 2 months ago
Typing discipline	Static, strong, safe, nominative, manifest
Filename extensions	.java, .class, .jar
Website	oracle.com/java/
Influenced by	
CLU, Simula67, LISP, SmallTalk, Ada 83, C++, C#, Eiffel, Mesa, Modula-3, Oberon, Objective-C, UCSD Pascal, Object Pascal	
Influenced	
Ada 2005, BeanShell, C#, Chapel, Clojure, ECMAScript, Fantom, Gambas, Groovy, Hack, Haxe, J#, Kotlin, PHP, Python, Scala, Seed7, Vala	

Java is one of the most used programming languages used in the development of virus-free systems [because:

- No explicit pointer
- Java Programs run inside a virtual machine sandbox

] and a open-source and free high level programming language and a computing platform for application development conceived by **James Gosling**, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan at Sun Microsystems, Inc. in 1991 to create programs to control consumer electronics (**which is now a subsidiary of Oracle Corporation**) and released in 1995, runs on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX, used in internet programming, mobile devices, games, *e-business solutions* etc., because of its **reliability**, high performance, simplicity and easy to use and **quick to learn** and rigid versus extensibility. Since Java has a runtime environment (JRE) and API, it is called a platform. As a language that has the Object-Oriented feature, Java supports:

- Polymorphism
- Inheritance
- Encapsulation
- Abstraction
- Classes
- Objects
- Instance
- Method
- Message Passing

Advantages:

- Object Oriented
- Platform Independent
- Simple

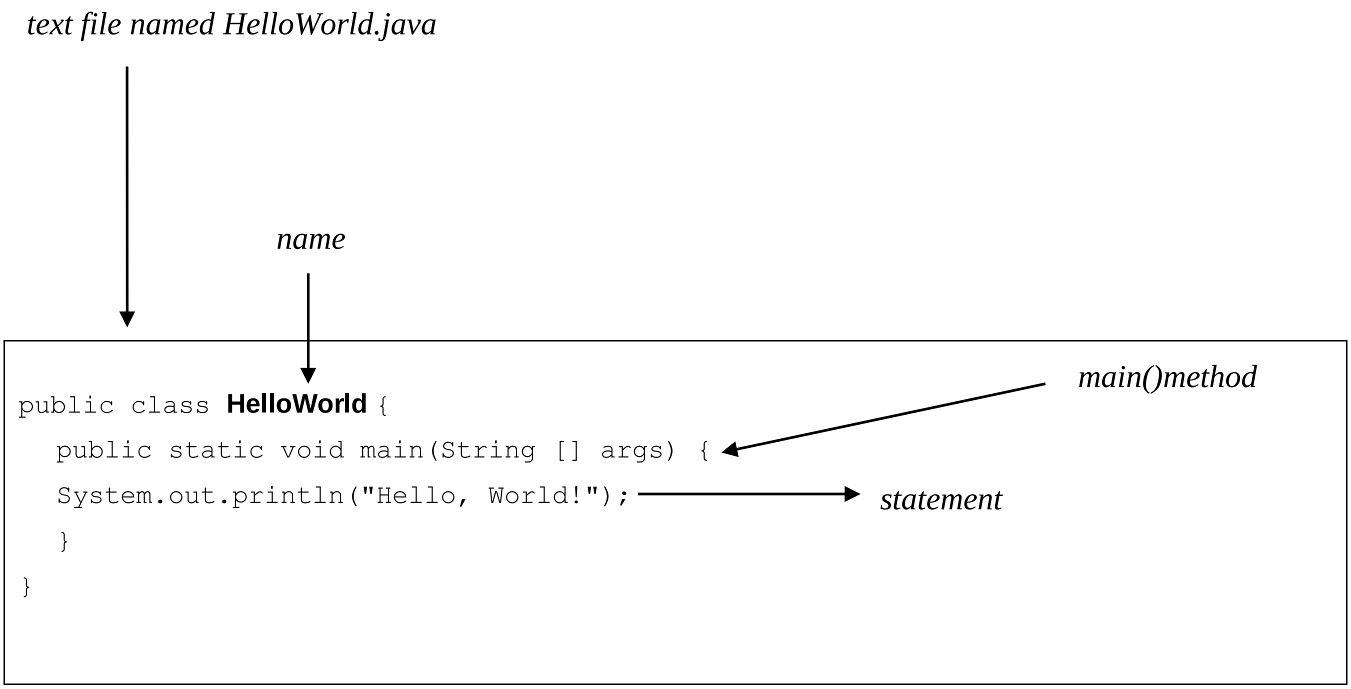
- Dynamic
- Secure
- Multi-threaded
- Architecture-neutral
- Portable
- Robust

Types of Java Applications:

- Standalone Application
- Web Application
- Enterprise Application
- Mobile Application

Java Language Keywords

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while



```
public class HelloWorld {
   public static void main(String [] args) {
   System.out.println("Hello, World!");
   }
}
```

- Declare a class with name **HelloWorld**.
- Declare the main method `public static void main(String args[])`
- Now Type the `System.out.println("Hello, World!");` which displays the text Hello World.

Process of Java program execution:

A Java program:

```
public class HelloWorld {
public static void main(String [] args) {
System.out.println("Hello, World!");
}
}
```

is written using Text Editor , such as [**Notepad++**, Notepad] and saved with [.*java*] Extension. File Saved with [.*java*] extension is called **Source Program** or Source Code.

```
//  HelloWorld.java


public class HelloWorld {
public static void main(String [] args) {
System.out.println("Hello, World!");
}
}


/* Because the class name is HelloWorld the source file should be named as
HelloWorld.java */
```

and sent to the ***java compiler*** (i.e., *javac compiler*) where the source program is compiled i.e., the program is entirely read and translated into Java byte codes (but not into **machine language**). If the ***javac compiler*** finds any error during compilation, it provides information about the error to the programmer. The programmer has to review code and check for the solution. And if there are no errors the translated program (i.e., java byte codes − a highly optimized set of instructions) is stored in computers main memory as ***HelloWorld.class*** and since the java byte codes cannot be trusted to be correct. Therefore before execution they are verified and converted to machine level language i.e., machine code sequence of 0s and 1s by Java run-time system, which is called the ***Java Virtual Machine*** (JVM) and is executed by a Java interpreter and

```
Hello, World!
```

is displayed on the console screen.

```
// Comment on one line

/* Comment on one or
```

```
More lines */

/** Documentation comment */
```

JVM (`Java Virtual Machine`) resides under RAM (**Random Access Memory** – the stuff that boosts up your computer to run faster and allows your computer to perform many tasks at the same time) and **JVM** comprises:

- **Class Loader:** it loads .class file that contains Java byte codes.
- **Byte Code Verifier:** it verifies byte codes.
- **Execution Engine:** it translates java byte codes to machine codes and executes them.

In the statement:

```
public class HelloWorld
```

The word "`HelloWorld`" implies: name of the class is **HelloWorld** and this class is public. public means that the class HelloWorld can be accessed by any other class in any package.

In the program:

```
public class HelloWorld {

/* This is my first java program.
    * This will print 'Hello, World!' as the output
    */

public static void main(String [] args) {
System.out.println("Hello, World!"); // prints Hello, World!
}
}

```

public class HelloWorld {

}

imply the body of the class HelloWorld (Here: the curly brace '{' imply the beginning of the class and the curly brace '}' imply the end of the class) within which the **main method**

```
public static void main(String [] args) {

}
```

is written. All method names should start with a Lower Case letter. For all class names the first letter should be in Upper Case.

`Java program processing starts from the main() method which is a mandatory part of every Java program.`

`public static void main(String [] args)` → **main method** (a collection of statements or methods like `System.out.println( )` that are grouped together to perform an operation) and this **main method** is *public*

and

{

} imply the body of the main method

(Here: the curly brace

`{`

imply the beginning of the main method and the curly brace

`}`

imply the end of the *main method*) within which the statement:

```
System.out.println("Hello, World!");
```

is written and executed.

- `main method` in java ***functions*** like main function **`main()`** in C and C++.

If the statement:

```
public class HelloWorld
```

is replaced by the statement:

`public class sample` i.e.,

```
public class sample {
public static void main(String [] args) {
System.out.println("Hello, World!");
}
}
```

Then the error will be displayed on the console screen because the program written in **notepad** is saved as ***HelloWorld.java*** not as **sample.java**. Name of the program file should exactly match the class name. When saving the file, you should save it using the class name and append '.java' to the end of the name.

Like C and C++, Java is also a case sensitive language i.e., capital letters (or **upper case letters**) must be avoided to prevent the display of error on the screen. **For example:** If the statement:

```
PUBLIC static void main(String [] args)
```

is written instead of the statement:

```
public static void main(String [] args)
```

, compilation Error will be displayed on the screen.

Each code statement must end with a semicolon. If we forget to end each **program statement** within the body of main method with a **semicolon** (";") − Error will be displayed on the screen.

The program begins its execution with the method:

```
public static void main(String [] args)
```

the main method − the **entry point** of the program execution i.e., the point from where the execution of Java program begins.

In the statement:

```
System.out.println();
```

- **System** → name of a standard class that contains variables and methods for supporting simple keyboard and character output to the display.
- **out** → represents the standard output stream

- **println()** → output method of the Java language which makes provision to print the output in the next line:

```
Hello,world!
```

  on the screen.

The text **Hello,world!** should be enclosed by the double quotation marks (" ") and should be written within the println method and this **println method** should be ended with the semicolon i.e.,

```
System.out.println("Hello,world!");
```

Otherwise the **compilation error** will be displayed on the console screen.

```
public class HelloWorld {
public static void main(String [] args) {
System.out.println("Hello, World!");
System.out.println("Hello, World!");
}
}
```

Output on the screen:

```
Hello, World!
Hello, World!
```

```
public class HelloWorld {
public static void main(String [] args) {
System.out.print("Hello, World!");
System.out.print("Hello, World!");
}
}
```

Output on the screen:

```
Hello, World!Hello, World!
```

In the statement:

```
public static void main(String [] args)
```

- public → implies: this method can accessed from anywhere outside the class HelloWorld

If the word "**public**" in the statement:

```
public static void main(String [] args)
```

is replaced by the word

```
private

or

protected
```

Then compilation error will be flagged on the screen because if the method is declared private or protected then this method does not make itself available to **JVM** for execution.

- main → implies the name of the method
- **static** means the main method is the part of the class HelloWorld

Why static?

Because the program execution begins from the main method and if the main method is not declared static then the execution of the program does not take place.

- `void` → implies the main method does not return any value i.e., **main method** return nothing when it completes execution.
- `String args[]`→ While running the program if we want to pass something to the main method, then this parameter is used as the way of taking input from the user − so we can pass some strings while running the program if we want.

Moreover, **JVM** cannot recognize the method:

```
public static void main(String [] args)
```

as method if the parameter `String [] args` is not included.

If the word args in the statement: `public static void main(String [] args)` is replaced by another word say ***jamesgosling*** or *java*

i.e.,

```
public class HelloWorld
{
public static void main(String [] jamesgosling)
{
System.out.println("Hello, World!");
}
}

public class HelloWorld {
public static void main (String [] java) {
```

```
System.out.println("Hello, World!");
}
}
```

No error will be displayed on the screen i.e.,

```
Hello,world!
```

will be displayed on the console screen.

If the statement:

```
                  public static void main(String [] args)
```

is replaced by the statement `public static void main(String [])` − Then the error is displayed on the screen.

Most Java programmers prefer args and argv i.e., the statements:

```
▪  public static void main(String [] args)
▪  public static void main(String [] argv)
```

are preferred.

If the space is left between the words Hello and World i.e., if the statement:

```
public class Hello   World
```

is written instead of the statement:

```
public class HelloWorld
```

Then the ***compilation error*** will be displayed on the console screen.

Java Modifiers

↓

- **Access Modifiers –** default, public, protected, private
- **Non-access Modifiers –** final, abstract, strictfp

Java Variables

↓

- Local Variables
- Class Variables (**Static Variables**)
- Instance Variables (**Non-static Variables**)

Java String Methods

Method	Description	Return Type
charAt()	Returns the character at the specified index (position)	char
codePointAt()	Returns the Unicode of the character at the specified index	int
codePointBefore()	Returns the Unicode of the character before the specified index	int

codePointCount()	Returns the Unicode in the specified text range of this String	int
compareTo()	Compares two strings lexicographically	int
compareToIgnoreCase()	Compares two strings lexicographically, ignoring case differences	int
concat()	Appends a string to the end of another string	String
contains()	Checks whether a string contains a sequence of characters	boolean
contentEquals()	Checks whether a string contains the exact same sequence of characters of the specified CharSequence or StringBuffer	boolean
copyValueOf()	Returns a String that represents the characters of the character array	String
endsWith()	Checks whether a string ends with the specified character(s)	boolean
equals()	Compares two strings. Returns true if the strings are equal, and false if not	boolean
equalsIgnoreCase()	Compares two strings, ignoring case considerations	boolean
format()	Returns a formatted string using the specified locale, format string, and arguments	String
getBytes()	Encodes this String into a sequence of bytes using the named charset, storing the result into a new byte array	byte[]
getChars()	Copies characters from a string to an array of chars	void
hashCode()	Returns the hash code of a string	int
indexOf()	Returns the position of the first found occurrence of specified characters in a	int

	string	
intern()	Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index	String
isEmpty()	Checks whether a string is empty or not	boolean
lastIndexOf()	Returns the position of the last found occurrence of specified characters in a string	int
length()	Returns the length of a specified string	int
matches()	Searches a string for a match against a regular expression, and returns the matches	boolean
offsetByCodePoints()	Returns the index within this String that is offset from the given index by codePointOffset code points	int
regionMatches()	Tests if two string regions are equal	boolean
replace()	Searches a string for a specified value, and returns a new string where the specified values are replaced	String
replaceFirst()	Replaces the first occurrence of a substring that matches the given regular expression with the given replacement	String
replaceAll()	Replaces each substring of this string that matches the given regular expression with the given replacement	String
split()	Splits a string into an array of substrings	String[]
startsWith()	Checks whether a string starts with specified characters	boolean
subSequence()	Returns a new character sequence that is	CharSequence

	a subsequence of this sequence	
substring()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character	String
toCharArray()	Converts this string to a new character array	char[]
toLowerCase()	Converts a string to lower case letters	String
toString()	Returns the value of a String object	String
toUpperCase()	Converts a string to upper case letters	String
trim()	Removes whitespace from both ends of a string	String
valueOf()	Returns the primitive value of a String object	String

Java Math Methods

Method	Description	Return Type
abs(x)	Returns the absolute value of x	double|float|int|long
acos(x)	Returns the arccosine of x, in radians	double
asin(x)	Returns the arcsine of x, in radians	double
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians	double
atan2(y,x)	Returns the angle theta from the conversion of rectangular coordinates (x, y) to polar coordinates (r, theta).	double
cbrt(x)	Returns the cube root of x	double
ceil(x)	Returns the value of x rounded up to its nearest integer	double
copySign(x, y)	Returns the first floating point x with the	double

	sign of the second floating point y	
cos(x)	Returns the cosine of x (x is in radians)	double
cosh(x)	Returns the hyperbolic cosine of a double value	double
exp(x)	Returns the value of E^x	double
expm1(x)	Returns e^x -1	double
floor(x)	Returns the value of x rounded down to its nearest integer	double
getExponent(x)	Returns the unbiased exponent used in x	int
hypot(x, y)	Returns sqrt(x^2 +y^2) without intermediate overflow or underflow	double
IEEEremainder(x, y)	Computes the remainder operation on x and y as prescribed by the IEEE 754 standard	double
log(x)	Returns the natural logarithm (base E) of x	double
log10(x)	Returns the base 10 logarithm of x	double
log1p(x)	Returns the natural logarithm (base E) of the sum of x and 1	double
max(x, y)	Returns the number with the highest value	double|float|int|long
min(x, y)	Returns the number with the lowest value	double|float|int|long
nextAfter(x, y)	Returns the floating point number adjacent to x in the direction of y	double|float
nextUp(x)	Returns the floating point value adjacent to x in the direction of positive infinity	double|float
pow(x, y)	Returns the value of x to the power of y	double
random()	Returns a random number between 0 and 1	double
round(x)	Returns the value of x rounded to its nearest integer	int
rint()	Returns the double value that is closest to x and equal to a mathematical integer	double
signum(x)	Returns the sign of x	double
sin(x)	Returns the sine of x (x is in radians)	double
sinh(x)	Returns the hyperbolic sine of a double value	double
sqrt(x)	Returns the square root of x	double
tan(x)	Returns the tangent of an angle	double
tanh(x)	Returns the hyperbolic tangent of a double	double

	value	
toDegrees(x)	Converts an angle measured in radians to an approx. equivalent angle measured in degrees	double
toRadians(x)	Converts an angle measured in degrees to an approx. angle measured in radians	double
ulp(x)	Returns the size of the unit of least precision (ulp) of x	double|float

- All Math methods are **static**.

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

- **Program 1.1**

Java program to print the word "hello Bill Gates" on screen

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println("hello Bill Gates");
}
}
```

The output on the screen:

```
hello Bill Gates
```

- **Program 1.2**

Java program to print the word " ****hello silicon city**** " on screen

```
public class HelloWorld {
public static void main(String [] args) {
System.out.println(" ****hello silicon city**** ");
}
}
```

The output on the screen:

```
****hello silicon city****
```

- **Program 1.3**

Java program to print

```
  *

*****

*****

*****

*****
```

on screen

```
public class HelloWorld {
public static void main(String [] args) {
System.out.println("\n    *   ");
System.out.println("\n  ***** ");
System.out.println("\n  ***** ");
System.out.println("\n  ***** ");
System.out.println("\n  ***** ");
}
}
```

The output on the screen:

```
  *

*****

*****

*****

*****
```

If new line \n is not included in the above program then the output on the screen is:

```
*********************
```

- **Write a program to print the following outputs:**

(a)

```
    *
   ****
 **java**
   ****
    *
```

(b)

```
***************
     *  *
* Hello World! *
     *  *
***************
```

(c)

```
Braces come in pairs!

Comments come in pairs!

All statements end with a semicolon!

Spaces are optional!

Must have a main method!

java is done mostly in lowercase. Like C & C++ it's also a case-sensitive language
```

Answers:

a)

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println("\n       *     ");
System.out.println("\n      ****   ");
System.out.println("\n   **java** ");
System.out.println("\n      ****    ");
System.out.println("\n       *      ");
}
}
```

b)

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println("\n          ****************         ");
System.out.println("\n                * *                ");
System.out.println("\n           * Hello World! *        ");
System.out.println("\n                * *                ");
System.out.println("\n         ****************          ");
}
}
```

c)

```
public class HelloWorld {
public static void main (String [] args) {
System.out.println("\n Braces come in pairs!");
System.out.println("\n Comments come in pairs!");
System.out.println("\n  All statements end with a semicolon!");
System.out.println("\n  Spaces are optional!");
System.out.println("\n Must have a main method!");
System.out.println("\n java is done mostly in lowercase. Like C & C++ it's also a case-
sensitive language");
}
}
```

- **Program 1.4**

  Java program to find the area of the circle

```
public class HelloWorld {
public static void main (String [] args) {
int r, area;
r = 2;
area = 3.14 * r * r;
System.out.println("The area of the circle = " + area);
}
}
```

The output on the screen:

```
The area of the circle = 12
```

In `C language`, the statement:

```
printf("The area of the circle = %d ",  area);
```

make the provision to print the output on the screen.

In `C++ language`, the statement

```
cout<<"The area of the circle = "<< area;
```

make the provision to print the output on the screen.

whereas in the Java language, the statement:

```
System.out.println("The area of the circle =  " + area);
```

make the provision to print the output on the screen.

In the statement:

```
System.out.println("The area of the circle =  " + area);
```

There are two strings:

- The area of the circle =
- area

plus operator (+) functions as the concatenation operator (concatenation means connecting two statements to produce a single statement) – which (here) concatenates the string:

```
"The area of the circle = "
```

and the string:

```
"area (which is 3.14 * r * r (= 12 since r = 2))"
```

producing a String statement:

```
The area of the circle = 12
```

which will be displayed on the screen as the result.

Even though if we write `ARGS` instead of `args` i.e., even though if we express `args` in capital letter, `No error` will be displayed on the screen.

```
public static void main(String [] ARGS) ⟶ no error will be displayed on the console screen
```

Operator	**Name**	**Description**	**Example**
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

- **Program 1.5**

  Java program to find the circumference of the circle

```
public class HelloWorld {
public static void main (String [] args) {
float r, circumference;
r = 2;
circumference = 2 * 3.14 * r;
System.out.println("The circumference of the circle = " + circumference);
}
}
```

The output on the screen is:

```
The circumference of the circle = 12.57
```

- **What will be the output of the following programs:**

  a)

```
public class HelloWorld {
public static void main (String [] args) {
double l, b, area;
l=2;
b=2.5;
area = l*b;
System.out.println("The area of the rectangle = " + area);
}
}
```

Answer:

```
The area of the rectangle = 5.0
```

b)

```
public class HelloWorld {
public static void main (String [] args) {
int a, b, c;
a= 3;
b=3;
c=3;
if ((a + b< c) || (b + c < a) || (a==b && b==c))
System.out.println(" the triangle is equilateral");
else
System.out.println(" the triangle is not possible");
}
}
```

Answer:

```
the triangle is equilateral
```

- **Program 1.6**

Java program to convert the temperature in Celsius to Fahrenheit

```
public class HelloWorld{
public static void main(String [] args){
float C, F;
C=38.5;
F = 9*C/5 +32;
System.out.println("temperature in Fahrenheit= " +F);
}
}
```

The output on the screen:

```
temperature in Fahrenheit= 101.3
```

- **Program 1.7**

  Java program to find the sum of two numbers

```
public class HelloWorld
{
public static void main(String [] args)
{
int a, b, sum;
a=1;
b=2;
sum = a + b;
System.out.println("the sum of a and b = " + sum);
}
}
```

The output on the screen:

```
the sum of a and b = 3
```

If you want to supply the values ***for*** a and b through the **key board**, then we have to rewrite the program as follows:

```
import java.util.Scanner;
public class HelloWorld
{
public static void main(String [] args) {
int a, b, sum;
Scanner scan = new Scanner(System.in);
System.out.print("Enter any two Numbers: ");
a = scan.nextInt();
b = scan.nextInt();
sum = a + b;
System.out.println("the sum of a and b = " + sum);
}
}
```

The output on the screen:

```
Enter any two Numbers:

If you enter two numbers 2 and 3

the sum of a and b = 5

will be outputted on the screen
```

Scanner is a class found in ***java.util package***. So to use Scanner class, we first need to include:

java.util package

in our program.

```
import java.util.Scanner; // This will import just the Scanner class
import java.util.*; // This will import the entire java.util package
```

The statement:

```
Scanner scan = new Scanner(System.in);
```

implies: declaring an object of the **Scanner class** "scan" to read the values entered for a and b through the key board. And the statements:

```
a = scan.nextInt();
b = scan.nextInt();
```

imply: scan is an object of Scanner class and nextInt() is a method of the object "scan" that allows the object "scan" to read only integer values from the keyboard.

- `nextInt()` that allows the object "scan" to read only integer values from the keyboard, methods that allows the object "scan" to read other data types from the **keyboard** are listed below:

Methods	**Datatype**
nextInt()	Integer
nextFloat()	Float
nextDouble()	Double
nextLong()	Long
nextShort()	Short
next()	Single word
nextLine()	Line of Strings
nextBoolean()	Boolean

- **Program 1.8**

Java program to find the square root of a number i)

```
public class HelloWorld
{
public static void main(String [] args) {
float x;
x = 233;
System.out.println(" square root of a number = " + Math.sqrt(x));
}
}
```

The output on the screen:

```
square root of a number = 15.264
```

If you want to supply the value for x through the **key board**, then the above program should take the form:

```
import java.util.Scanner;
public class HelloWorld {
public static void main(String [] args) {
int x;
Scanner scan = new Scanner(System.in);
System.out.print("Enter any Number: ");
x = scan.nextFloat();
System.out.println(" square root of a number = " + Math.sqrt(x));
}
}
```

The output on the screen:

```
Enter any Number:

If you enter the number 233
```

```
square root of a number = 15.264337522

will be outputted on the screen.
```

ii)

```
public class HelloWorld
{
public static void main(String [] args) {
double x;
x = 233;
System.out.println(" square root of a number = " + Math.sqrt(x));
}
}
```

The output on the screen:

```
square root of a number = 15.264337522473747
```

If you want to supply the value for x through the key board, then the above program should take the form:

```
import java.util.Scanner;
public class HelloWorld {
public static void main(String [] args) {
double x;
Scanner scan = new Scanner(System.in);
System.out.print("Enter any Number: ");
x = scan.nextDouble();
System.out.println(" square root of a number = " + Math.sqrt(x));
}
}
```

The output on the screen:

```
Enter any Number:
If you enter the number 233
square root of a number = 15.264337522473747
will be outputted on the screen.
```

- **Program 1.9**

  What will be the output of the following program:

```
public class HelloWorld{
public static void main(String[] args) {
char c;
c = 'A';
System.out.println("ch= " + c);
}
}
```

The output on the screen:

ch=A

If you want to supply the value for c through the **key board**, then the above program should take the form:

```
public class HelloWorld {
public static void main(String[] args) throws Exception {
char c;
System.out.print("Enter a character:");
c = (char)System.in.read();
System.out.println("ch= " + c);
}
}
```

The output on the screen:

```
Enter a character:

If you enter the character K

ch= K

will be outputted on the screen.
```

- **Note:** Exception is a problem that arises during the execution of a program. When an exception occurs, program abnormally terminates and disrupts – throws Exception should be written after the statement `public static void main(String[] args)` so that the exceptions are thrown to the **operating system** to handle and the program will be successfully executed and the output will be displayed on the screen.

- **Program 2.0**

```
import java.util.Scanner;
public class HelloWorld {
public static void main(String [] args) {
String m;
Scanner in = new Scanner(System.in);
System.out.print("Enter the word: ");
m = in.nextLine();
System.out.println(" the word you entered = " + m);
}
}
```

The output on the screen:

```
Enter the word:

If you enter the word dog

the word you entered = dog

will be outputted on the screen.
```

```
If the statement:

                                 m = scan.nextLine();

is written instead of

                                  m = in.nextLine();

Then we have to replace the statement:

                         Scanner in = new Scanner(System.in);

by the statement:

                        Scanner scan = new Scanner(System.in);

Otherwise compilation error will be displayed on the console screen.
```

- **What is the mistake in the following program:**

```
public class HelloWorld
{
static public void main(String args []) {
float x;
x = 233;
System.out.println(" cube root of a number = " + Math.cbrt(x));
}
}
```

Answer:

There is no mistake in the above program.

The statement:

```
public static void main(String[] args)
```

can also be written as:

```
static public void main(String args [])
```

The output on the screen is:

```
cube root of a number = 6.1534494936636825
```

- **Program 2.1**

Java program to find the product of two numbers.

```
public class HelloWorld{
public static void main(String [] args) {
int a, b, product;
a=1;
b=2;
product = a * b;
System.out.println("the product of a and b = " + product);
}
}
```

The output on the screen:

```
the sum of a and b = 2
```

If you want to supply the values *for* a and b through the **key board**, then we have to rewrite the above program as follows:

```
import java.util.Scanner;
public class HelloWorld {
public static void main(String [] args) {
int a, b, product;
Scanner scan = new Scanner(System.in);
System.out.print("Enter any two Numbers: ");
a = scan.nextInt();
b = scan.nextInt();
product = a * b;
System.out.println("the product of a and b = " + product);
}
}
```

The output on the screen:

```
Enter any two Numbers:
If you enter two numbers 6 and 3
the product of a and b = 18
will be outputted on the screen
```

If you want to assign the **floating point values** *for* a and b, then the above program should take the form:

```
import java.util.Scanner;
public class HelloWorld {
public static void main(String [] args) {
float a, b, product;
Scanner scan = new Scanner(System.in);
System.out.print("Enter any two Numbers: ");
a = scan.nextFloat();
b = scan.nextFloat();
product = a * b;
System.out.println("the product of a and b = " + product);
}
}
```

The output on the screen:

```
Enter any two Numbers:

If you enter two floating point values 2.9 and 3.6

the product of a and b = 10.44

will be outputted on the screen.
```

If the statement:

```
System.out.println("the product of a and b = " + product);
```

is replaced by the statement:

```
System.out.println(a + "* " + b + " = " + product);
```

Then the **output on the screen** is:

```
2.9 * 3.6 = 10.44
```

- **Note:** The word public in the statement:

```
public class HelloWorld
```

implies: that the program or the data within the program (such as methods, variables etc.) can be accessed directly by an external java program.

If replace the word **public** by **private** i.e.,

```
private class HelloWorld
```

is written instead of

```
public class HelloWorld
```

then the program or the **data within the program** (such as methods, variables etc.) cannot be accessed directly by an external program.

If you want to insert a **10 digit number** for a and b i.e.,

a=1000000000

b=3000000000, then the statement:

int a, b, product;

should be replaced by the statement:

long int a, b, product;

i.e.,

```
public class HelloWorld{
```

```
public static void main(String [] args){
long int a, b, product;
a=1000000000;
b=2000000000;
product = a * b;
System.out.println("the product of a and b = " + product);
}
}
```

The output on the screen:

the product of a and b = 3000000000000000000

- **What will be the output of the following program:**

```
public class HelloWorld{
static public void main(String args []) {
float x;
x = 2;
System.out.println(" square of a number = " + Math.pow((x), 2));
}
}
```

Answer:

square of a number = 4

- **Program 2.2**

  Java program to find the square of a number

```
public class HelloWorld{
public static void main(String [] args){
int a, b;
a=2;
b = a * a;
System.out.println("the square of a = " + b);
}
}
```

The output on the screen:

the square of a = 4

If you want to supply the value for a through the **key board**, then we have to rewrite the above program as follows:

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args) {
int a, b;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any Number: ");
a = scan.nextInt();
b = a * a;
System.out.println("the square of a = " + b);
}
}
```

The output on the screen:

Enter any number:

If you enter a number 3

the square of a = 9 will be outputted on the screen.

- **Note:**

- If scan.nextint() is written instead of **scan.nextInt()**
- public static void main(string [] args); is written instead of

```
public static void main(String [] args)
```

- system.out.println("the square of a = " + b); is written instead of

System.out.println("the square of a = " + b);

Then the compilation error will be displayed on the screen.

- **Program 2.3**

Java program to find the greatest of two numbers using *if* - *else* statement

The syntax of *if* – *else* statement is:

```
if (this condition is true)

{

print this statement using the println method

}

else

{

print this statement using the println method

}
```

```
public class HelloWorld{
public static void main(String [] args){
int a, b;
a=2;
b =3;
if(a>b)
{
System.out.println("a is greater than b");
}
else
{
System.out.println("b is greater than a");
}
}
}
```

The output on the screen:

b is greater than a

In the above program:

if the condition (a> b) is true, then the statement

```
{
System.out.println("a is greater than b");
}
```

is executed to print the output:

a is greater than b

else the statement

```
{
System.out.println("b is greater than a");
}
```

is executed to print the output:

b is greater than a

If you want to supply the values *for* a and b through the **key board**, then the above program should be rewritten as:

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args){
int a, b;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any two Numbers: ");
a = scan.nextInt();
b = scan.nextInt();
if(a>b)
{
System.out.println("a is greater than b");
}
else
{
System.out.println("b is greater than a");
}
}
}
```

The output on the screen:

```
Enter any two Numbers:

If you enter two numbers 2 and 3

b is greater than a

will be outputted on the screen.
```

- **Note:**

Even if the statements:

```
System.out.println("a is greater than b");
System.out.println ("b is greater than a");
```

are not written within the braces { }

i.e.,

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args){
int a, b;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any two Numbers: ");
a = scan.nextInt();
b = scan.nextInt();
if(a>b)
System.out.println("a is greater than b");
if(b>a)
System.out.println("b is greater than a");
}
}
```

There will no display of *compilation error* on the screen or there will be **no change in the output** displayed on the screen (i.e., b is greater than a will be outputted on the screen).

- **Program 2.4**

  Java program to find the greatest of three numbers using ***else if*** statement

  The syntax of *else if* statement is:

```
if (this condition is true)
{
```

```
print this statement using the method System.out.println( );

}

else if (this condition is true)

{

print this statement using the method System.out.println( );

}

else

{

print this statement using the method System.out.println( );

}
```

```
public class HelloWorld{
public static void main(String [] args){
int a, b, c;
a=2;
b =3;
c=4;
if(a>b&&a>c)
{
System.out.println("a is greater than b and c");
}
else if(b>a&&b>c)
{
System.out.println("b is greater than a and c");
}
else
{
System.out.println("c is greater than b and a");
}
}
}
```

The output on the screen:

```
c is greater than b and a
```

If the statements:

```
if(a>b&&a>c)
{
System.out.println("a is greater than b and c");
}
else if(b>a&&b>c)
{
System.out.println("b is greater than a and c");
}
else
{
System.out.println("c is greater than b and a");
}
```

are replaced by the statements:

```
if(a>b&&a>c)
{
System.out.println(a + "is greater than" + b  + "and" + c);
}
else if(b>a&&b>c)
{
System.out.println(b + "is greater than" + a  + "and" + c);
}
else
{
System.out.println(c + "is greater than" + b  + "and" + a);
}
```

Then the output on the screen is:

4 is greater than 3 and 2

- **Program 2.5**

Java program to find the average of 10 numbers

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args) {
int N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, X;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any ten Numbers: ");
N1 = scan.nextInt();
N2 = scan.nextInt();
N3 = scan.nextInt();
N4 = scan.nextInt();
N5 = scan.nextInt();
N6 = scan.nextInt();
N7 = scan.nextInt();
N8 = scan.nextInt();
N9 = scan.nextInt();
N10 = scan.nextInt();
X = (N1 + N2 + N3 + N4 + N5 + N6 + N7 + N8 + N9 + N10) /10;
System.out.println("the average of 10 numbers = " + X);
}
}
```

The output on the screen:

```
Enter any ten Numbers:

If you enter ten numbers 1, 2, 3, 4, 5, 6, 7, 8, 9 and 10

the average of 10 numbers = 5

will be outputted on the screen.
```

- **Note:** The average of 10 numbers is 5.5, the output on the screen is 5 because int is used instead of float.

- **Program 2.6**

  Java program to find the simple interest

```
public class HelloWorld{
public static void main(String [] args) {
int P,T, R, SI;
P = 1000;
T = 2;
R = 3;
SI = P*T*R/100;
System.out.println("the simple interest = " + SI);
}
}
```

The output on the screen:

```
the simple interest = 60
```

If you want to supply the values for P, T and R through the **key board**, then the above program should take the form:

```
import java.util.Scanner;
public class HelloWorld {
public static void main(String [] args) {
int P,T, R, SI;
Scanner scan = new Scanner(System.in);
System.out.println("Enter principal amount:");
P = scan.nextInt();
System.out.println("Enter time:");
T = scan.nextInt();
System.out.println("Enter rate of interest:");
R = scan.nextInt();
SI = P*T*R/100;
System.out.println("the simple interest = " + SI);
}
}
```

The output on the screen:

```
Enter principal amount:

If you enter the principal amount 1000

Enter time:

If you enter the time 2

Enter rate of interest:

If you enter the rate of interest 3

the simple interest = 60

will be outputted on the screen.
```

- **Program 2.7**

  Java program to find the senior citizen

```
public class HelloWorld{
public static void main(String [] args){
int age;
age=20;
if(age> = 60)
{
System.out.println("senior citizen");
}
else
{
System.out.println("not a senior citizen");
}
}
}
```

The output on the screen:

```
not a senior citizen
```

- (age> = 60) implies age greater than or equal to 60

If you want to supply the value for age through the **key board**, then the above program should be rewritten as:

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args){
int age;
Scanner scan = new Scanner(System.in);
System.out.println("Enter the age: ");
age = scan.nextInt();
if(age> = 60)
{
System.out.println("senior citizen");
}
else
{
System.out.println("not a senior citizen");
}
}
}
```

The output on the screen:

```
Enter the age:

If you enter the age 60
```

senior citizen

will be outputted on the screen.

Suppose if you enter the age 28

not a senior citizen

will be outputted on the screen.

- **Program 2.8**

  Java program to get marks for 3 subjects and declare the result:

  If the marks >= 35 in all the subjects the student passes else fails.

```
public class HelloWorld{
public static void main(String [] args){
int M1, M2,M3;
M1 = 38;
M2= 45;
M3 = 67;
if(M1>= 35 && M2>= 35 && M3>= 35)
{
System.out.println("candidate is passed");
}
else
{
System.out.println("candidate is failed");
}
}
}
```

The output on the screen:

```
candidate is passed
```

If you want to supply the values for marks M1, M2 and M3 through the **key board**, then the above program should be rewritten as:

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args) {
int age;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any three Numbers: ");
M1= scan.nextInt();
M2 = scan.nextInt();
M3 = scan.nextInt();
if(M1>= 35 && M2>= 35 && M3>= 35)
{
System.out.println("candidate is passed");
}
else
{
System.out.println("candidate is failed");
}
}
}
```

The output on the screen:

```
Enter any three Numbers:
If you enter three numbers 26, 28, 39
candidate is failed
will be outputted on the screen.
```

- **Program 2.9**

  Java program to find profit or loss

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args) {
int CP, SP, loss, profit;
Scanner scan = new Scanner(System.in);
System.out.println("Enter cost price: ");
CP = scan.nextInt();
System.out.println("Enter selling price: ");
SP = scan.nextInt();
if(SP>CP)
{
System.out.println("profit= " + (SP-CP));
}
else
{
System.out.println("loss ="  +(CP-SP));
}
}
}
```

The output on the screen:

```
Enter cost price:

If you enter the cost price 25

Enter selling price:

If you enter the selling price 26

profit = 1

will be outputted on the screen.
```

- **Program 3.0**

  Java program to find the incremented and decremented values of two numbers

```
public class HelloWorld{
public static void main(String [] args){
int a, b, c, d, e, f;
a = 10;
b=12;
c=a+1;
d=b+1;
e=a-1;
f=b-1;

System.out.print("the incremented value of a = "+ c);
System.out.print("the incremented value of b = "+ d);
System.out.print("the decremented value of a = "+ e);
System.out.print("the decremented value of b = "+ f);
}
}
```

The output on the screen:

```
the incremented value of a = 11 the incremented value of b = 13
the decremented value of a = 9 the decremented value of b = 11
```

If the statements:

```
System.out.print("the incremented value of a = "+ c);
System.out.print("the incremented value of b = " + d);
System.out.print("the decremented value of a = " + e);
System.out.print("the decremented value of b = " + f);
```

are replaced by the statements:

```
System.out.print("\n the incremented value of a = "  + c);
System.out.print("\n the incremented value of b = " + d);
System.out.print("\n the decremented value of a = " + e);
System.out.print("\n the decremented value of b = " + f);
```

Then the **output on the screen** is:

```
the incremented value of a = 11
the incremented value of b = 13
the decremented value of a = 9
the decremented value of b = 11
```

i.e., \n make provision for the another result to print in the new line. If you want to supply the values *for* a and b through the **key board**, then the above program should take the form:

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args){
int a, b, c, d, e, f;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any Number: ");
a = scan.nextInt();
System.out.println("Enter any Number: ");
b = scan.nextInt();
c=a+1;
d=b+1;
e=a-1;
f=b-1;
System.out.print("\n the incremented value of a = "  + c);
System.out.print("\n the incremented value of b = " + d);
System.out.print("\n the decremented value of a = " + e);
System.out.print("\n the decremented value of b = " + f);
}
}
```

The output on the screen:

```
Enter any Number:
If you enter the value 2
Enter any Number:
If you enter the value 3

the incremented value of a = 3
the incremented value of b = 4
the decremented value of a = 1
the decremented value of b = 2

will be outputted on the screen.
```

- **What will be the output of the following programs:**

A)

```
import java.util.Scanner;
public class temperature{
public static void main(String [] args) {
float T1, T2, A;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any Number: ");
T1 = scan.nextFloat();
System.out.println("Enter any Number: ");
T2 = scan.nextFloat();
A = (T1 + T2) / 2;
System.out.println("the average temperature of the day = " + A);
}
}
```

Answer:

Enter any Number:

If you enter the number 2

Enter any Number:

If you enter the number 3

the average temperature of the day = 2.5

will be outputted on the screen.

B)

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args) {
int P;
Scanner scan = new Scanner(System.in);
System.out.println("Enter the percentage: ");
P = scan.nextInt();
if(P >= 60)
{
System.out.println("first class");
}
else if(P>=50&&P <60)
{
System.out.println("second class");
}
else
{
System.out.println("pass class");
}
if(P<40)
{
System.out.println("fail");
}
}
}
```

Answer:

```
Enter the percentage:

If you enter the number 60

first class

will be outputted on the screen.
```

- **Program 3.1**

  Java program to calculate the discounted price and the total price after discount

  Given:

  - If purchase value is greater than 1000, 10% discount
  - If purchase value is greater than 5000, 20% discount
  - If purchase value is greater than 10000, 30% discount

- **discounted price**

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args) {
int PV, dis;
Scanner scan = new Scanner(System.in);
System.out.println("Enter purchased value: ");
PV = scan.nextInt();
if(PV<1000)
{
```

```
System.out.println("dis = " + PV* 0.1);
}
else if(PV>5000)
{
System.out.println("dis = " + PV* 0.2);
}
else
{
System.out.println("dis= " + PV* 0.3);
}
}
}
```

The output on the screen:

```
Enter purchased value:

If you enter the purchased value 6500

dis = 1300

will be outputted on the screen.
```

- **total price**

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args) {
int PV, total;
Scanner scan = new Scanner(System.in);
System.out.println("Enter purchased value: ");
PV = scan.nextInt();
if(PV<1000)
{
System.out.println("total= " +  PV - PV* 0.1);
}
else if(PV>5000)
{
```

```
System.out.println("total = " + PV- PV* 0.2);
}
else
{
System.out.println("total= " + PV- PV* 0.3);
}
}
}
```

The output on the screen:

```
Enter purchased value:

If you enter the purchased value 650

total = 585

will be outputted on the screen.
```

- **Combing both the programs (above), we can write:**

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args){
int PV, dis, total;
Scanner scan = new Scanner(System.in);
System.out.println("Enter purchased value: ");
PV = scan.nextInt();
if(PV<1000)
{
System.out.println("dis = " + PV* 0.1);
System.out.println("total= " + total - dis);
}
else if(PV>5000)
{
System.out.println("dis = " + PV* 0.2);
System.out.println("total= " + total - dis);
}
else
{
System.out.println("dis = " + PV* 0.3);
System.out.println("total= " + total - dis);
}
}
}
```

The output on the screen:

```
Enter purchased value:
If you enter the purchased value 850
dis = 85
total = 765
will be outputted on the screen.
```

- **Program 3.2**

Java program to print the first ten natural numbers using for loop statement

```
public class HelloWorld{
public static void main(String [] args){
int i;
for (i=1; i<=10; i++)
System.out.println("value of i = " + i);
}
}
```

The output on the screen is:

```
value of i = 1 value of i = 2  value of i= 3  value of i= 4  value of i= 5  value of
i= 6 value of i = 7 value of i= 8  value of i = 9  value of i = 10
```

If the statement:

```
System.out.println("value of i = " + i);
```

is replaced by the statement:

```
System.out.println("\n value of i = " + i);
```

Then the **output on the screen** is:

```
value of i = 1
value of i = 2
value of i = 3
value of i = 4
value of i = 5
value of i = 6
value of i = 7
value of i = 8
value of i = 9
value of i = 10
```

If the *for loop* statement:

```
for (i=2; i<=10; i++)
```

is written instead of the statement:

`for(i=1; i<=10; i++)`, then the **output on the screen** is:

```
value of i = 2  value of i = 3  value of i= 4  value of i= 5  value of i= 6
value of i = 7 value of i= 8  value of i = 9  value of i= 10
```

If the *for loop* statement:

```
for (i=1; i<=10; i++)
```

is written instead of the statement: `for (i=1; i<10; i++)`, then the **output on the screen** is:

```
value of i = 1 value of i = 2  value of i= 3  value of i= 4  value of i= 5  value of
i= 6 value of i = 7 value of i= 8  value of i = 9
```

- **Note:** the condition i<=10 tells to print till value of i =10 but the condition i<10 tells to print till value of i=9

If the statement:

`for(i=1; i=10; i++)`

is written instead of the statement: `for (i=1; i<=10; i++)`, then the output on the screen is:

```
value of i = 10  value of i = 10  value of i = 10  value of  i = 10  value of i= 10
value of i= 10 value of i = 10 value of i= 10  value of i = 10   value of i = 10
value of i = 10   value of i = 10  value of i = 10    value of i = 10   value of i =
10

continues ....
```

If the statement:

System.out.println("\n value of i = " + i); is replaced by the statement

```
System.out.println("\n " + i);
```

Then the **output on the screen** is:

```
1

2

3

4

5

6

7

8

9

10
```

- **What is the mistake in the following program:**

```
public class HelloWorld{
public static void main(String []args) throws Exception{
System.out.println("Hello World");
}
}
```

Answer:

There is no mistake in the above program. Addition of the statement throws Exception does not make any change in the output displayed on the screen or give rise to **any compilation error** on the screen.

- **Program 3.3**

What will be the output of the following program:

```
public class HelloWorld{
public static void main(String [] args) {
int i;
for (i =1; i<=5; i ++)
System.out.println("\n Linux is not portable");
}
}
```

Answer:

```
Linux is not portable

Linux is not portable

Linux is not portable

Linux is not portable

Linux is not portable
```

- **Java program to print the first ten natural numbers using *while loop* statement**

The syntax of *while loop* statement is:

```
while (this is the condition)

{

execute this statement;

}
```

```
public class HelloWorld{
public static void main(String [] args)
{
int i = 1;
while (i<=10)
{
System.out.println("\n  " + i++);
}
}
}
```

The output on the screen is:

```
1

2

3

4

5

6

7

8

9

10
```

If the statement:

int i = 1;

is replaced by

```
int i = 0;
```

Then the output on the screen is:

```
0

1

2

3

4

5

6

7

8

9

10
```

Similarly if the statement int i = 0; is replaced by

```
int i = 7;
```

Then the **output on the screen** is:

```
7

8

9

10
```

- **Java program to print first 10 numbers using *do while loop* statement**

  The syntax of do while loop statement is:

  ```
  do

  {

  execute this statement;
  ```

```
        System.out.println(" \n i= " + i++);

}

while(this is the condition);
```

```
public class HelloWorld{
public static void main(String [] args)
{
int i =1;
do
{
System.out.println(" \n i= " + i++);
} while (i<=10);
}
}
```

The output on the screen is:

```
i=1

i=2

i=3

i=4

i=5

i=6

i=7

i=8

i=9

i=10
```

The statement:

```
System.out.println(" \n i= " + i++);
```

is executed and then condition **(i<=10)** is checked. If condition (i<=10) is true then

The statement:

```
System.out.println(" \n i= " + i++);
```

is executed again. This process repeats until the given condition (i<=10) becomes false.

- **Program 3.4**

  Java program to print the characters from A to Z using *for loop*, ***do while loop*** and *while loop* statement.

  - **Java program to print the characters from A to Z using for loop statement:**

```
public class HelloWorld{
public static void main(String [] args) {
char a;
for( a='A'; a<='Z'; a++)
System.out.println("\n   " + a);
}
}
```

The output on the screen:

```
A

B

C
```

```
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
W
X
Y
Z
```

- **Java program to print the characters from A to Z using while loop statement:**

```
public class HelloWorld{
public static void main(String [] args) {
char a = 'A';
while (a<='Z')
{
System.out.println("\n  "  + a++);
}
}
}
```

- **Java program to print the characters from A to Z using do while loop statement:**

```
public class HelloWorld{
public static void main(String [] args) {
char a = 'A';
do
{
System.out.println("\n "  +  a++);
} while (a<='Z');
}
}
```

- **Program 3.5**

Java program to print the given number is even or odd.

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args) {
int a;
Scanner scan = new Scanner(System.in);
System.out.println("Enter a number: ");
a = scan.nextInt();
if(a%2 = = 0)
{
System.out.println("the number is even");
}
else
{
System.out.println("the number is odd");
}
}
}
```

The output on the screen:

```
Enter a number:

If you enter the number 4

the number is even

will be outputted on the screen.
```

- **Program 3.6**

  Java program to print the remainder of two numbers

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args) {
int a, b, c;
Scanner scan = new Scanner(System.in);
System.out.println("Enter a number: ");
a = scan.nextInt();
System.out.println("Enter a number: ");
b = scan.nextInt();
c = a%b;
System.out.println("the remainder of a and b =  " + c);
}
}
```

The output on the screen:

```
Enter a number:
If you enter the number 3
Enter a number:
If you enter the number 2

the remainder of a and b = 1

will be outputted on the screen.
```

Since (a =3 and b =2). **Therefore:** 3 divided by 2 (i.e., a divided by b) yields the remainder equal to 1.

- **Program 3.7**

  Java program to check equivalence of two numbers.

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args) {
int x, y;
Scanner scan = new Scanner(System.in);
System.out.println("Enter a number: ");
x = scan.nextInt();
System.out.println("Enter a number: ");
y = scan.nextInt();
if(x-y==0)
{
System.out.println("the two numbers are equivalent");
}
else
{
System.out.println("the numbers are not equivalent");
}
}
```

The output on the screen:

```
Enter a number:
If you enter the number 2
Enter a number:
If you enter the number 2

the two numbers are equivalent

will be outputted on the screen.
```

- **Program 3.8**

  Java program to print the leap year or not

```
public class HelloWorld{
public static void main(String [] args) {
int year;
year =1996;
if(year%4==0)
{
System.out.println("leap year");
}
else
{
System.out.println("not a leap year");
}
}
}
```

The output on the screen:

```
leap year
```

- **What will be the output on the screen:**

```
public class HelloWorld{
int a =5;
public static void main(String[] args){
int a =2 ;
System.out.println(" value of a = " + a);
}
}
```

Answer:

```
value of a = 2
```

If the statement:

```
System.out.println(" value of a = " + a);
```

is replaced by the statement

```
System.out.println(" value of a = " + ::a);
```

(where **::** denote scope resolution operator)

i.e.,

```
public class HelloWorld{
int a =5;
public static void main(String[] args){
int a =2 ;
System.out.println(" value of a = " + ::a);
}
}
```

Then the compilation error will be displayed on the screen because [like **C++]** JAVA does not hold **or** support the resolution operator.

- **Program 3.9**

  Java program to print whether the given number is positive or negative

```
public class HelloWorld{
public static void main(String [] args){
int a;
a = -35;
if(a>0)
{
System.out.println("number is positive");
}
else
{
System.out.println(" number entered is negative");
}
}
}
```

The output on the screen:

```
number entered is negative
```

Since a = −35. Therefore:

a is less than 0 i.e., a<0

The statement

```
{
System.out.println("number is negative");
}
```

is executed to print the output:

number entered is negative

- **Program 4.0**

  Java program to print the sum of the first 10 digits using for loop statement:

```
public class HelloWorld{
public static void main(String [] args) {
int i,  sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
System.out.println("sum of  the first 10 digits = " + sum);
}
}
```

The output on the screen:

sum of the first 10 digits = 55

The statement:

```
System.out.println("sum of  the first 10 digits = " + sum);
```

is executed to display the output:

sum of the first 10 digits = 55

on the screen.

If the statement:

int i, sum = 0;

is replaced by

```
int i, sum = 1;
```

Then the **output on the screen** is:

sum of the first10 digits = 56

- **What will be the output if the *for loop* statement** for(i =1; i<=10; i++) **is replaced by the statement** for(i =2; i<10; i++)?

Answer: `sum of 10 digits = 44`

If the statement

`int i, sum, sum = 0;` is written instead of `int i, sum = 0;`

Then the *compilation error message* will be displayed on the screen (stating that sum is twice declared).

If the *for loop* is ended with a semicolon i.e.,

```
for( i=1; i<=10; i++);
```

Then the **compilation error** will be displayed on the console screen.

- **Program 4.1**

  Java program to print the average of the first 10 numbers using for loop statement

```
public class HelloWorld{
public static void main(String [] args){
int i, avg, sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
System.out.println("sum of the first 10 numbers = " + sum);
System.out.println("average of the first 10 numbers =  " + avg);
}
}
```

The output on the screen:

```
sum of the first 10 numbers = 55
average of the first 10 numbers = 5
```

If the **data type** float is used i.e.,

```
public class HelloWorld{
public static void main(String [] args) {
float i, avg, sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
System.out.println("sum of the first 10 numbers = " + sum);
System.out.println("average of the first 10 numbers = " + avg);
}
}
```

The output on the screen:

```
sum of the first 10 numbers = 55

average of the first 10 numbers = 5.5
```

- **Program 4.2**

  Java program to print the product of the first 10 digits using *for loop* statement

```
public class HelloWorld{
public static void main(String [] args) {
int i, product = 1;
for( i=1; i<=10; i++)
product = product * i;
System.out.println("the product of the first 10 digits = " + product);
}
}
```

The output on the screen:

```
the product of the first 10 digits = 3628800
```

The statement:

```
System.out.println("the product of the first10 digits = " + product);
```

is executed to display the output:

```
the product of the first 10 digits = 3628800
```

If the statement int i, product = 1; is replaced by int i, product = 0;

Then the output on the screen is:

```
the product of the first 10 digits = 0
```

If the statement `for(i=1; i<=10; i++)` is replaced by `for(i=5; i<=8; i++)`

Then the output on the screen is:

```
the product of the first 10 digits = 1680
```

- **Program 4.3**

  Java Program to print the table of a number using the *for loop* statement

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args){
int n, i;
Scanner scan = new Scanner(System.in);
System.out.println("Enter a number: ");
n = scan.nextInt();
for( i=1; i<=5; i++)
System.out.println ( \n n  +  "  *   " +  i + "  =  " +  n * i);
}
}
```

Output on the screen:

```
Enter any number:
If you enter the number 2 (i.e., n=2)

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10

will be outputted on the screen.
```

If the symbol * is replaced by +

i.e.,

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String [] args){
int n, i;
Scanner scan = new Scanner(System.in);
System.out.println("Enter a number: ");
n = scan.nextInt();
for( i=1; i<=5; i++)
System.out.println ( \n n  +  "  +   " +  i + "  =  " +  n + i);
}
}
```

Then the output on the screen is:

```
Enter any number:
If you enter the number 2 (i.e., n=2)

2 + 1 = 3
2 + 2 = 4
2 + 3 = 5
2 + 4 = 6
2 + 5 = 7

will be outputted on the screen.
```

- **Program 4.4**

  Java program to print the first 10 numbers starting from one together with their squares

```
public class HelloWorld{
public static void main(String[] args){
int i;
for( i=1; i<=10; i++)
System.out.println(" number = " +  i +  " its square =  " + i*i);
}
}
```

The output on the screen:

```
number = 1 its square=1number = 2 its square=4number = 3 its square=9number = 4 its
square=16number = 5 its square=25number = 6 its square=36number = 7 its square=49number
= 8 its square=64number = 9 its square=81number= 10 its square=100
```

If the statement:

```
System.out.println(" number = " +  i +  " its square =  " + i*i);
```

is replaced by the statement

```
System.out.println(" \n number = " +  i +  " its square =  " + i*i);
```

i.e.,

```
public class HelloWorld{
public static void main(String[] args){
int i;
for( i=1; i<=10; i++)
System.out.println(" \n number = " +  i +  " its square =  " + i*i);
}
}
```

Then the output on the screen is:

```
number = 1 its square=1

number = 2 its square=4

number = 3 its square=9

number = 4 its square=16

number = 5 its square=25

number = 6 its square=36

number = 7 its square=49

number = 8 its square=64

number = 9 its square=81

number= 10 its square=100
```

If the statement:

```
System.out.println(" \n number = " +  i +  " its square =  " + i*i);
```

is replaced by the statement:

```
System.out.println(" \n number = " +  i +  " \t its square =  " + i*i);
```

i.e.,

```
public class HelloWorld{
public static void main(String[] args){
int i;
for( i=1; i<=10; i++)
System.out.println(" \n number = " +  i +  " \t its square =  " + i*i);
}
}
```

Then the output on the screen is:

```
number=1     its square=1
number=2     its square=4
number=3     its square=9
number=4     its square=16
number=5     its square=25
number=6     its square=36
number=7     its square=49
number=8     its square=64
number=9     its square=81
number=10    its square=100
```

tab /t is included because to leave space between

number =1   **and**   its square=1

If the statement:

```
System.out.println(" \n number = " + i +  " \t its square =  " + i*i);
```

is replaced by the statement:

```
System.out.println(" \n number = " +  i +  " \n its square =  " + i*i);
```

i.e.,

```
public class HelloWorld{
public static void main(String[] args){
int i;
for( i=1; i<=10; i++)
System.out.println(" \n number = " +  i +  " \n its square =  " + i*i);
}
}
```

Then the output on the screen is:

```
number = 1
its square=1
number = 2
its square=4
number = 3
its square=9
number = 4
its square=16
number = 5
its square=25
number = 6
its square=36
number = 7
its square=49
number = 8
its square=64
number = 9
```

```
its square=81
number= 10
its square=100
```

- **Write a program to print the first 10 numbers starting from one together with their squares and cubes:**

Answer:

```
public class HelloWorld{
public static void main(String[] args) throws Exception{
int i;
for( i=1; i<=10; i++)
System.out.println(" \n number = " +  i +  " its square =  " + i*i + " its cube = " +
i*i*i);
}
}
```

- **Program 4.5**

  Java program to print the sum of two numbers using method

```
public class HelloWorld{
public static void main(String[] args){
int a, b, c;
a = 11;
b = 6;
c = add (a, b);
System.out.println(" sum of two numbers = " + c);
}
public static int add (int a, int b) {
```

```
return (a+b) ;
}
}
```

The output on the screen:

```
sum of two numbers = 17
```

There are 2 methods in the above program:

- public static void main(String[] args)
- public static int add (int a, int b)

`public static void main(String[] args) imply`: **main method** and

{

} imply the body of the **main method** with in which the program statements:

```
int a, b, c;
a = 11;
b = 6;
c = add (a, b);
System.out.println(" sum of two numbers = " + c);
```

are written.

- Like in C ++ (the **function declaration** is not made) and unlike in C [(the **function declaration** is made) − there is no need for method declaration in Java (i.e., without the

method declaration the program will be successfully executed and the result will be outputted on the screen]

`public static int add (int a, int b)` **imply: the** method to add two integers x and y and

{

return (a+b) ;

}

imply the body of the method **public static int add (int a, int b)**

main method:

```
public static void main(String[] args)
```

and the **method:**

public static int add (int a, int b)

should be written inside the body of the **public class HelloWorld**.

The statement

```
int a, b, c;
```

imply that we creating the integer variables a, b and c.

The statements:

```
a = 11;
b = 6;
c = add (a, b);
```

imply that we are assigning the values to the created variables.

The statement:

```
c = add (x, y);
```

imply **method call** (i.e., we are calling the method public static int add (int a, int b) to add the values (i.e., 11 and 6) and return the result (i.e., 17) to the statement

```
System.out.println(" sum of two numbers = " + c);
```

to make provision to display the output of the sum of two entered numbers as 17 on the screen.

- **Java program to print the product of two numbers using method**

```
public class HelloWorld{
public static void main(String[] args) {
int a, b, c;
a = 2;
b = 3;
c = mult (a, b);
System.out.println(" product of two numbers  = " + c);
}
public static int mult (int a, int b){
return (a*b) ;
}
}
```

The output on the screen:

```
product of two numbers = 6
```

will be outputted on the screen.

- **Java program to print the greatest of two numbers using method**

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String[] args) {
int a, b;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any two numbers: ");
a = scan.nextInt();
b = scan.nextInt();
System.out.println(" largest of two numbers  = " + max (a, b) );
}
public static int max (int a, int b) {
if(a>b)
return a;
else
return b;
}
}
```

The output on the screen:

```
Enter any two numbers:

If you enter two numbers 5 and 2

largest of two numbers= 5
```

```
will be outputted on the screen.
```

- **Java program to print the greatest of three numbers using method**

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String[] args) {
int a, b, c;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any three numbers: ");
a = scan.nextInt();
b = scan.nextInt();
c= scan.nextInt();
System.out.println(" largest of two numbers  = " + max (a, b, c) );
}
public static int max (int a, int b, int c) {
if(a>b && a>c)
return a;
else if (b>c && b>a)
return b;
else
return c;
}
}
```

The output on the screen:

```
Enter any three numbers:
If you enter three numbers 3, 5 and 10

largest of three numbers = 10

will be outputted on the screen.
```

- **Java program to print the square of the number using method**

```
import java.util.Scanner;
public class HelloWorld{
public static void main(String[] args) {
int x;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any number: ");
x = scan.nextInt();
System.out.println("square of the number = " + square (x));
}
public static int square (int x){
return x*x;
}
}
```

The output on the screen is:

Enter any number:

If you enter the number 5

square of the number = 25

will be outputted on the screen.

- **Program 4.6**

  Switch (case) allows to make decision from the number of choices i.e., from the number of cases

For example:

```
public class HelloWorld{
```

```
public static void main(String[] args)throws Exception{
char ch;
System.out.print("Enter a character:");
ch = (char)System.in.read();
switch(ch)
{
case 'R':
System.out.print("Red");
break;
case 'W':
System.out.print("White");
break;
case 'Y':
System.out.print("Yellow");
break;
case 'G':
System.out.print("Green");
break;
default:
System.out.print("Error");
break;
}
}
}
```

The output on the screen is:

```
Enter a character:
If you enter a character R

Red

will be outputted on the screen.
```

- **Program 4.7**

Java program to print the output

Element [0] = 16

Element [1] = 18

Element [2] = 20

Element [3] = 25

Element [4] = 36

using arrays:

```
public class HelloWorld{
public static void main(String[] args){
int i;
int [] num = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
System.out.println("Element [" +  i + " ] = " + num[i]);
}
}
```

The output on the screen:

Element [0] = 16

Element [1] = 18

Element [2] = 20

Element [3] = 25

Element [4] = 36

Ends because of the condition **i**<**5**.

```
Array declaration in C:
int num [5] = {16, 18, 20, 25, 36};
```

```
or
int num [] = {16, 18, 20, 25, 36};

Array declaration in C++:
int num [5] = {16, 18, 20, 25, 36};
or
int num [] = {16, 18, 20, 25, 36};

But array declaration in java:
int [] num = {16, 18, 20, 25, 36};
```

- **Java program to print the sum of the elements in array.**

```
public class HelloWorld{
public static void main(String[] args){
int i, sum = 0;
int [] num = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
sum = sum + num[i];
System.out.println("Sum of the Elements in the array  = " +  sum);
}
}
```

The output on the screen:

```
Sum of the Elements in the array = 115
i.e., 16 + 18 + 20 + 25 + 36 = 115
```

- **Java program to print the average of the elements in the array**

```
public class HelloWorld{
public static void main(String[] args){
int i, avg, sum = 0;
int [] num = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
sum = sum + num[i];
avg = sum/5;
System.out.println("Sum of the Elements in the array = " +  sum);
System.out.println("average of the Elements in the array = " +  avg);
}
}
```

The output on the screen:

```
Sum of the Elements in the array = 115

average of the elements in the array = 23
```

- **Write a program to print**

  Einstein [0] = E

  Einstein [1] = I

  Einstein [2] = N

  Einstein [3] = S

  Einstein [4] = T

  Einstein [5] = E

  Einstein [6] = I

  Einstein [7] = N

  using arrays

Answer:

```
public class HelloWorld{
public static void main(String[] args) throws Exception{
int i;
char [] num = {'E' , 'I', 'N', 'S', 'T', 'E', 'I', 'N'};
for(i=0; i<8; i++)
System.out.println("Einstein [" + i + " ] = " + num[i]);
}
}
```

- **What will be the output of the following programs?**

i)

```
public class HelloWorld{
public static void main(String[] args) throws Exception{
int i;
int [] name = {'E' , 'I', 'N', 'S', 'T ', 'E', 'I', 'N'};
for(i=0; i<8; i++)
System.out.println("Einstein [" +  i + " ] = " + name[i]);
}
}
```

Answer:

```
Einstein [0] = 69
Einstein [1] = 73
Einstein [2] = 78
Einstein [3] = 83
Einstein [4] = 84
Einstein [5] = 69
Einstein [6] = 73
Einstein [7] = 78
```

ii)

```
public class HelloWorld{
public static void main(String[] args) throws Exception{
int i;
char [] body  = {'b', 'o', 'd', 'y'};
for(i=0; i<4; i++)
System.out.println("body [" +  body [i] + " ] = " + body [i]);
}
}
```

Answer:

```
body [b] = b
body [o] = o
body [d] = d
body [y] = y
```

```
import java.util.Scanner;
public class HelloWorld {
public static void main(String [] args) {
int x, y;
Scanner scan = new Scanner(System.in);
System.out.print("Enter any Number: ");
x = scan.nextFloat();
System.out.print("Enter any Number: ");
y = scan.nextInt();
System.out.println(" square root of x = " + Math.sqrt(x));
System.out.println(" square root of y = " + Math.sqrt(y));
}
}
```

The output on the screen:

```
Enter any Number:
If you enter the number 9

square root of x = 3

will be outputted on the screen.
Enter any Number:
If you enter the number 4

square root of y = 2

will be outputted on the screen.
```

If

```
/*

*/
```

is introduced i.e.,

```
import java.util.Scanner;
public class HelloWorld {
public static void main(String [] args) {
int x, y;
Scanner scan = new Scanner(System.in);
System.out.print("Enter any Number: ");
x = scan.nextInt();
/*
System.out.print("Enter any Number: ");
y = scan.nextInt();
*/
System.out.println(" square root of x = " + Math.sqrt(x));

/*
System.out.println(" square root of y = " + Math.sqrt(y));
*/
}
}
```

Then the output on the screen is:

Enter any Number:

If you enter the number 9

square root of x = 3

will be outputted on the screen.

- **What is the mistake in the following program:**

```
public class HelloWorld {
public static void main(String [] args) {
long float x;
Scanner scan = new Scanner(System.in);
System.out.print("Enter any Number: ");
x = scan.nextFloat();
System.out.println(" square root of x = " + Math.cbrt(x));
}
```

Answer:

long float x; should not be used – only float x should be used because **Java** do not support the data type such as long int, long float etc.

- **Program 4.8**

  continue and break statements:

A)

```
public class HelloWorld{
public static void main(String []args){
int i;
for (i=1; i<=5; i++){
if (i==3){
continue;
}

System.out.println("" + i);
}
}
}
```

Output on the screen:

```
1

2

4

5
```

B)

```
public class HelloWorld {
public static void main(String []args){
int i;
for (i=1; i<=5; i++){
if (i==3){
break;
}
System.out.println("" + i);
}
}
}
```

Output on the screen:

1

2

- **What will be the output of the following program:**

```
public class HelloWorld {
public static void main(String args[]){
System.out.println(Math.max(1269, 1356));
}
}
```

Output on the screen:

1356

```
Abstraction -→ hiding implementation details from the user by providing interface

Encapsulation -→ hiding data
```

In the statement:

"1 + 2"

"1" and "2" imply the operands and the plus symbol imply the operator.

- **Polymorphism**

Suppose if you are in **class room** that time you behave like a student, when you are in **shopping mall** at that time you behave like a customer, when you at your **home** at that time you behave like a son or daughter. Your ability to present in different-different behaviors is known as polymorphism.

In the example:

```
public class HelloWorld
{
public static void main(String [] args)
{
int a, b, sum;
a=1;
b=2;
sum = a + b;
System.out.println("the sum of a and b = " + sum);
}
}
```

Plus symbol ("+") act as an arithmetic operator in the statement:

```
sum = a+b;
```

and it acts as the **concatenation operator** in the statement:

```
System.out.println("the sum of a and b = " + sum);
```

The ability of plus symbol to behave both as **arithmetic operator** and concatenation operator is known as polymorphism.

- **Inheritance**

```
public class game {
}

public class player extends game{
}
```

Here public class player extends game implies: class player is public and it is the sub class of the **class game**. Since class player is the subclass of class game – class player automatically takes on all the behavior and attributes of its parent class "game" i.e., methods or fields within the class game will be automatically be included in the class player.

The statements:

```
public class player extends game

public class game extends ball
```

implies: that **class player** is not only a subclass of class game but also it is a subclass of class ball.

- **Encapsulation**

```
public class Account {
           private decimal accountBalance = 500.00;

           public decimal CheckBalance() {
                    return accountBalance;
           }
  }

/* accountBalance can be checked via public "CheckBalance" method provided by the
"Account" class but its value cannot be manipulated because data variable
accountBalance is declared private */
```

Encapsulation is the technique of bringing the data variables and methods in single frame and declaring data variable private (so it cannot be accessed by anyone outside the class, thereby **hiding** or **encapsulating** the data variable (String name) within the public class Student) and providing indirect access to the data variable via public methods.

SQL (Structured Query Language)	
Paradigm	Declarative
Family	Query language
Designed by	Donald D. Chamberlin Raymond F. Boyce
Developer	ISO/IEC

First appeared	1974; 46 years ago
Stable release	SQL:2016 / December 2016; 3 years ago
Typing discipline	Static, strong
OS	Cross-platform
Website	www.iso.org/standard/63555.html

Major implementations

Many

Dialects

- SQL-86
- SQL-89
- SQL-92
- SQL:1999
- SQL:2003
- SQL:2006
- SQL:2008
- SQL:2011
- SQL:2016

Influenced by

Datalog

Influenced

CQL, LINQ, SPARQL, SOQL, PowerShell, JPQL, jOOQ, N1QL

SQL (file format)

Filename extension	`.sql`
Internet media type	`application/sql`
Developed by	ISO/IEC
Initial release	1986
Type of format	Database
Standard	ISO/IEC 9075
Open format?	Yes
Website	www.iso.org/standard/63555.html

SQL: Structured query language – a standard computer language developed by American computer scientists **Donald D. Chamberlin** and **Raymond F. Boyce** at IBM in 1974 to create database, store, update, manipulate, delete and retrieve data stored in database. It became a standard of the American National Standards Institute (ANSI) in 1986 and of the International Organization for Standardization (ISO) in 1987. SQL is **NOT case sensitive**: select is the same as SELECT.

SQL Keywords

Keyword	**Description**
ADD	Adds a column in an existing table
ADD CONSTRAINT	Adds a constraint after a table is already created

ALTER	Adds, deletes, or modifies columns in a table, or changes the data type of a column in a table
ALTER COLUMN	Changes the data type of a column in a table
ALTER TABLE	Adds, deletes, or modifies columns in a table
ALL	Returns true if all of the subquery values meet the condition
AND	Only includes rows where both conditions is true
ANY	Returns true if any of the subquery values meet the condition
AS	Renames a column or table with an alias
ASC	Sorts the result set in ascending order
BACKUP DATABASE	Creates a back up of an existing database
BETWEEN	Selects values within a given range
CASE	Creates different outputs based on conditions
CHECK	A constraint that limits the value that can be placed in a column
COLUMN	Changes the data type of a column or deletes a column in a table
CONSTRAINT	Adds or deletes a constraint
CREATE	Creates a database, index, view, table, or

	procedure
CREATE DATABASE	Creates a new SQL database
CREATE INDEX	Creates an index on a table (allows duplicate values)
CREATE OR REPLACE VIEW	Updates a view
CREATE TABLE	Creates a new table in the database
CREATE PROCEDURE	Creates a stored procedure
CREATE UNIQUE INDEX	Creates a unique index on a table (no duplicate values)
CREATE VIEW	Creates a view based on the result set of a SELECT statement
DATABASE	Creates or deletes an SQL database
DEFAULT	A constraint that provides a default value for a column
DELETE	Deletes rows from a table
DESC	Sorts the result set in descending order
DISTINCT	Selects only distinct (different) values
DROP	Deletes a column, constraint, database, index, table, or view
DROP COLUMN	Deletes a column in a table

DROP CONSTRAINT	Deletes a UNIQUE, PRIMARY KEY, FOREIGN KEY, or CHECK constraint
DROP DATABASE	Deletes an existing SQL database
DROP DEFAULT	Deletes a DEFAULT constraint
DROP INDEX	Deletes an index in a table
DROP TABLE	Deletes an existing table in the database
DROP VIEW	Deletes a view
EXEC	Executes a stored procedure
EXISTS	Tests for the existence of any record in a subquery
FOREIGN KEY	A constraint that is a key used to link two tables together
FROM	Specifies which table to select or delete data from
FULL OUTER JOIN	Returns all rows when there is a match in either left table or right table
GROUP BY	Groups the result set (used with aggregate functions: COUNT, MAX, MIN, SUM, AVG)
HAVING	Used instead of WHERE with aggregate functions
IN	Allows you to specify multiple values in a WHERE clause

INDEX	Creates or deletes an index in a table
INNER JOIN	Returns rows that have matching values in both tables
INSERT INTO	Inserts new rows in a table
INSERT INTO SELECT	Copies data from one table into another table
IS NULL	Tests for empty values
IS NOT NULL	Tests for non-empty values
JOIN	Joins tables
LEFT JOIN	Returns all rows from the left table, and the matching rows from the right table
LIKE	Searches for a specified pattern in a column
LIMIT	Specifies the number of records to return in the result set
NOT	Only includes rows where a condition is not true
NOT NULL	A constraint that enforces a column to not accept NULL values
OR	Includes rows where either condition is true
ORDER BY	Sorts the result set in ascending or descending order
OUTER JOIN	Returns all rows when there is a match in either

	left table or right table
PRIMARY KEY	A constraint that uniquely identifies each record in a database table
PROCEDURE	A stored procedure
RIGHT JOIN	Returns all rows from the right table, and the matching rows from the left table
ROWNUM	Specifies the number of records to return in the result set
SELECT	Selects data from a database
SELECT DISTINCT	Selects only distinct (different) values
SELECT INTO	Copies data from one table into a new table
SELECT TOP	Specifies the number of records to return in the result set
SET	Specifies which columns and values that should be updated in a table
TABLE	Creates a table, or adds, deletes, or modifies columns in a table, or deletes a table or data inside a table
TOP	Specifies the number of records to return in the result set
TRUNCATE TABLE	Deletes the data inside a table, but not the table itself
UNION	Combines the result set of two or more SELECT

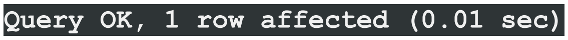
	statements (only distinct values)
UNION ALL	Combines the result set of two or more SELECT statements (allows duplicate values)
UNIQUE	A constraint that ensures that all values in a column are unique
UPDATE	Updates existing rows in a table
VALUES	Specifies the values of an INSERT INTO statement
VIEW	Creates, updates, or deletes a view
WHERE	Filters a result set to include only records that fulfill a specified condition

How to create database in MySQL

First you have to open `MySQL terminal` and then you have to enter the command:

```
create database data;
                                                     or
CREATE DATABASE data;
```

And press enter. Then



```
Query OK, 1 row affected (0.01 sec)
```

will be displayed on the console screen indicating that database named data is created. And if you enter the command:

```
show databases;
```

And press enter. Then

```
+--------------------
+

| Database           |

+--------------------
+

| CODINGGROUND       |

| data               |

| information_schema |

| mysql              |

| performance_schema |

| test
```

will be displayed on the console screen. And if you want to create a table in the database "data", then you have to enter the command:

```
use data;
```

And press enter. Then

```
Database changed
```

will be displayed on the console screen stating that your active database is now "data". And if you want to create a table named "states" with three fields: id, state, and population:

id	state	population

in your active database named "data", then you have to enter the command:

```
CREATE TABLE states (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, state CHAR(25),
population INT(9));
```

And press enter. Then

```
Query OK, 0 rows affected (0.07 sec)
```

will be displayed on the console screen stating that the above table is created.

Note:

- The INT command will make the id field contain only numbers (i.e., integers).
- The NOT NULL command makes sure that the id field cannot be left blank or empty.
- The PRIMARY KEY designates the id field as the key field in the table.

- The AUTO_INCREMENT command will automatically assign increasing values into the id field, essentially automatically numbering each entry.
- The CHAR(**characters**) and INT(**integers**) commands designate the types of data allowed in those fields. The number next to the commands CHAR and INT indicate how many characters or integers can fit in the field.

Now it’s time to start entering your information. Use the following **command**:

```
INSERT INTO states (id, state, population) VALUES (NULL, 'Karnataka', 256666); INSERT
INTO states (id, state, population) VALUES (NULL, 'Assam', 2568585); INSERT INTO states
(id, state, population) VALUES (NULL, 'Kashmir', 2569);
```

to input your entry. Then

```
Query OK, 1 row affected (0.03 sec)



Query OK, 1 row affected (0.01 sec)



Query OK, 1 row affected (0.00 sec)
```

will be displayed on the **console screen** stating that you have inputted your entry. And if you enter the following command:

```
select*from states;
```

Then, your created table named "states" will be displayed on the screen as follows:

```
+----+-----------+------------
+

| id | state     | population |


+----+-----------+------------
+

|  1 | Karnataka |     256666 |


|  2 | Assam     |    2568585 |


|  3 | Kashmir   |       2569 |


+----+-----------+------------+
```

And if you wish to create the following table

```
+----+-----------+------------+----------
+

| id | state     | population | language |


+----+-----------+------------+----------
+

|  1 | Karnataka |     256666 | Kannada  |
|  2 | Assam     |        2569 | Assami    |
```

You have to use the following command:

```
CREATE TABLE states (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, state CHAR (25),
population INT (9), language CHAR (25));
```

And press enter and

```
Query OK, 0 rows affected (0.03 sec)
```

will be displayed on the console screen and then you should enter the following command:

```
INSERT INTO states (id, state, population, language) VALUES (NULL, 'Karnataka', 256666,
'Kannada'); INSERT INTO states (id, state, population, language) VALUES
(NULL,'Assam',2569,'Assami');
```

And press enter and

```
Query OK, 1 row affected (0.01 sec)


Query OK, 1 row affected (0.00 sec)
```

will be displayed on the console screen and if you enter the command:

```
select*from states;
```

Then the above table will be displayed on the screen.

If you enter the command:

```
select state, population from states;
```

Then

```
state           | population |



+----------------+------------
+

| Karnataka      |       2562 |



| assam          |      25695 |
```

will be displayed on the console screen. And if you enter the command:

```
select state from states;
```

Then

```
| state          |



+----------------
+

| Karnataka      |



| assam
```

will be displayed on the console screen.

If you enter the command:

```
select*from states where language ='kannada';
```

Then

```
+----+-----------+------------+----------+-----------
+
| id | state     | population | language |
+----+-----------+------------+----------+-----------
+
|  1 | Karnataka |       2562 | kannada
```

will be displayed on the console screen. Similarly, if you enter the command:

```
select*from states where id =2;
```

Then

```
+----+-------+------------+----------+---------
+
| id | state | population | language |
+----+-------+------------+----------+---------
+
|   2 | assam |        25695 | assami
```

will be displayed on the console screen.

- **SQL and & or command**

If you enter the command:

```
select*from states where population =2566 and language ='kannada';

or

select*from states where population = 22666 or language = 'kannada';
```

Then

```
+----+-----------+------------+----------
+

| id | state     | population | language |


+----+-----------+------------+----------
+

|  1 | Karnataka |       2566 | Kannada  |


+----+-----------+------------+----------+
```

will be displayed on the console screen.

If you enter the command:

```
select *from states where population = 2566 or language ='assami';
```

```
+----+-----------+------------+----------
+

| id | state     | population | language |


+----+-----------+------------+----------
+

|  1 | Karnataka |       2566 | Kannada  |


|  2 | assam     |      22666 | assami   |
```

- **HOW to insert information into the table**

If you enter the command:

```
INSERT INTO states (id, state, population, language) VALUES (NULL, 'tamil nadu',
288,'tamil');
```

Then

```
+----+-----------+-----------+----------
+

| id | state      | population | language |


+----+-----------+-----------+----------
+

|  1 | Karnataka  |       2566 | Kannada  |


|  2 | assam      |      22666 | assami   |


|  3 | tamil nadu |        288 | tamil    |


+----+-----------+-----------+----------+
```

will be displayed on the console screen.

- **UPDATE INFORMATION**

If you enter the command:

```
update states set language =' telagu', population = 1 where state ='Karnataka';
```

Then

```
Query OK, 1 row affected (0.01 sec)


Rows matched: 1  Changed: 1  Warnings: 0
```

will be displayed on the console screen. And if you enter the command:

```
select*from states;
```

Then

```
+----+------------+------------+----------
+

| id | state      | population | language |


+----+------------+------------+----------
+

|  1 | Karnataka  |          1 |  telagu  |


|  2 | assam      |      22666 | assami   |


|  3 | tamil nadu |        288 | tamil    |


+----+------------+------------+----------
+

3 rows in set (0.00 sec)
```

will be displayed on the console screen.

- **DELETE information**

If you enter the command:

```
delete from states where language ='assami' and state ='assam';
```

Then

```
Query OK, 1 row affected (0.00 sec)
```

will be displayed on the console screen. And if you enter the command:

```
select*from states;
```

Then

```
+----+------------+-----------+----------
+

| id | state      | population | language |


+----+------------+-----------+----------
+

|  1 | Karnataka  |          1 |  telagu  |


|  3 | tamil nadu |        288 | tamil    |


+----+------------+-----------+----------
+

2 rows in set (0.00 sec)
```

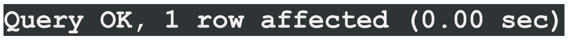
will be displayed on the console screen.

- **How to delete database in MYSQL**

If want to delete database "**dbtest**" from MySQL. Then you have to enter the command:

```
drop database dbtest;
```

Then

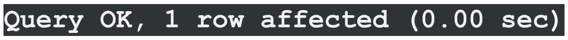


```
Query OK, 1 row affected (0.00 sec)
```

will be displayed on the console screen stating that database "**dbtest**" is deleted from MySQL. If want to delete table "states" from database "**dbtest**". Then you have to enter the command:

```
drop table states;
```

Then



```
Query OK, 1 row affected (0.00 sec)
```

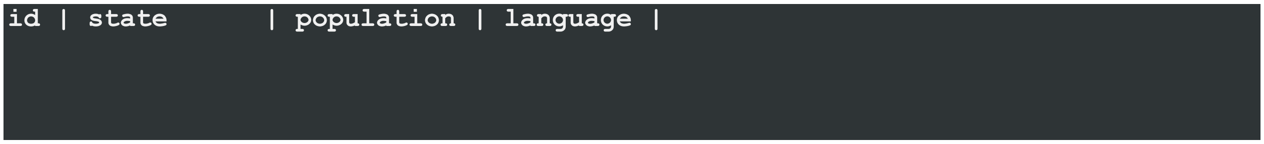
will be displayed on the console screen stating that table "states" is deleted from database "**dbtest**".

- **Limit Data Selection From MySQL Database**

If enter the command:

```
select*from states limit 1;
```

Then



```
id | state       | population | language |
+----+------------+-----------+----------
+

|  1 | Karnataka  |      2566 | Kannada  |
```

will be displayed on the console screen.

If enter the command:

`select*from states limit 2;`

Then

```
id | state       | population | language |


+----+------------+-----------+----------
+

|  1 | Karnataka  |      2566 | Kannada  |


|  2 | assam      |     22666 | assami   |
```

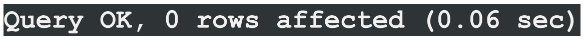
will be displayed on the console screen.

- **TRUNCATE**

If you enter the command:

`truncate states;`

Then

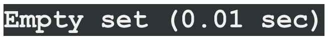


```
Query OK, 0 rows affected (0.06 sec)
```

will be displayed on the **console screen** stating that all the rows are removed from the table "states". And you can confirm it by entering the command:

```
select*from states;
```

Then:



```
Empty set (0.01 sec)
```

will be displayed on the console screen.

Some of The Most Important SQL Commands

- `SELECT - extracts data from a database`
- `UPDATE - updates data in a database`
- `DELETE - deletes data from a database`
- `INSERT INTO - inserts new data into a database`
- `CREATE DATABASE - creates a new database`
- `ALTER DATABASE - modifies a database`
- `CREATE TABLE - creates a new table`
- `ALTER TABLE - modifies a table`
- `DROP TABLE - deletes a table`
- `CREATE INDEX - creates an index (search key)`
- `DROP INDEX - deletes an index`

Python

Paradigm	Multi-paradigm: functional, imperative, object-oriented, structured, reflective
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	1990; 30 years ago
Stable release	3.8.3 / 13 May 2020; 32 days ago
Preview release	3.9.0b3 / 9 June 2020; 5 days ago
Typing discipline	Duck, dynamic, gradual (since 3.5)
OS	Linux, macOS, Windows Vista (and newer) and more
License	Python Software Foundation License
Filename extensions	.py, .pyi, .pyc, .pyd, .pyo (prior to 3.5), .pyw, .pyz (since 3.5)
Website	www.python.org
Major implementations	
CPython, PyPy, Stackless Python, MicroPython, CircuitPython, IronPython, Jython, RustPython	
Dialects	
Cython, RPython, Starlark	
Influenced by	
ABC, Ada, ALGOL 68, APL, C, C++, CLU, Dylan, Haskell, Icon, Java, Lisp, Modula-	

3, Perl, Standard ML
Influenced
Apache Groovy, Boo, Cobra, CoffeeScript, D, F#, Genie, Go, JavaScript, Julia, Nim, Ring, Ruby, Swift

Python is a popular, very powerful high-level language (like C, C++, Perl, and Java – and its name is derived from "**Monty Python's Flying Circus**" – a British television series), object- oriented programming scripting language ideally designed for rapid prototyping of complex applications by Dutch programmer "**Guido van Rossum**" in the early 1990s (often referred to as a "glue" language, meaning that it is capable to work in mixed-language environment) – which has simple syntax similar to the English language and is easy to understand, easy to use, write, modify and debug and flexible and easy to implement and run on open source operating systems like Linux, Windows, Macintosh, Solaris, FreeBSD, OS/2, Amiga, AROS, AS/400 and is employed to perform automated testing of applications (i.e., to execute tests of applications, report outcomes and compare results with earlier test runs) and to increase the effectiveness and speed of software testing and its other commercial uses include financial applications, educational software, games, production of special effects for such movies as The Phantom Menace and The Mummy Returns, and business software. And python might not be the **best choice** for building the following types of applications and systems: Graphics-intensive applications, such as action games − where performance is important (because it is true that CODE WRITTEN IN Python, C# and visual basic etc. is far slower than the same code write in C++. Hence, C++ is necessarily preferred). You should be very careful while working on python code. Indentation in python takes the center-stage. You cannot write a loop or a conditional statement without using indentation. There is a popular saying that code in python should be as guido indented it not how he intended it.

Python Keywords

Keyword	Description
and	A logical operator
as	To create an alias
assert	For debugging
break	To break out of a loop
class	To define a class
continue	To continue to the next iteration of a loop
def	To define a function
del	To delete an object
elif	Used in conditional statements, same as else if
else	Used in conditional statements
except	Used with exceptions, what to do when an exception occurs
False	Boolean value, result of comparison operations
finally	Used with exceptions, a block of code that will be executed no matter if there is an exception or not
for	To create a for loop
from	To import specific parts of a module

global	To declare a global variable
if	To make a conditional statement
import	To import a module
in	To check if a value is present in a list, tuple, etc.
is	To test if two variables are equal
lambda	To create an anonymous function
None	Represents a null value
nonlocal	To declare a non-local variable
not	A logical operator
or	A logical operator
pass	A null statement, a statement that will do nothing
raise	To raise an exception
return	To exit a function and return a value
True	Boolean value, result of comparison operations
try	To make a try...except statement

while	To create a while loop
with	Used to simplify exception handling
yield	To end a function, returns a generator

Python Built in Functions

Function	Description
abs()	Returns the absolute value of a number
all()	Returns True if all items in an iterable object are true
any()	Returns True if any item in an iterable object is true
ascii()	Returns a readable version of an object. Replaces none-ascii characters with escape character
bin()	Returns the binary version of a number
bool()	Returns the boolean value of the specified object
bytearray()	Returns an array of bytes
bytes()	Returns a bytes object
callable()	Returns True if the specified object is callable, otherwise False
chr()	Returns a character from the specified Unicode code.

classmethod()	Converts a method into a class method
compile()	Returns the specified source as an object, ready to be executed
complex()	Returns a complex number
delattr()	Deletes the specified attribute (property or method) from the specified object
dict()	Returns a dictionary (Array)
dir()	Returns a list of the specified object's properties and methods
divmod()	Returns the quotient and the remainder when argument1 is divided by argument2
enumerate()	Takes a collection (e.g. a tuple) and returns it as an enumerate object
eval()	Evaluates and executes an expression
exec()	Executes the specified code (or object)
filter()	Use a filter function to exclude items in an iterable object
float()	Returns a floating point number
format()	Formats a specified value
frozenset()	Returns a frozenset object
getattr()	Returns the value of the specified attribute (property or method)
globals()	Returns the current global symbol table as a dictionary

hasattr()	Returns True if the specified object has the specified attribute (property/method)
hash()	Returns the hash value of a specified object
help()	Executes the built-in help system
hex()	Converts a number into a hexadecimal value
id()	Returns the id of an object
input()	Allowing user input
int()	Returns an integer number
isinstance()	Returns True if a specified object is an instance of a specified object
issubclass()	Returns True if a specified class is a subclass of a specified object
iter()	Returns an iterator object
len()	Returns the length of an object
list()	Returns a list
locals()	Returns an updated dictionary of the current local symbol table
map()	Returns the specified iterator with the specified function applied to each item
max()	Returns the largest item in an iterable
memoryview()	Returns a memory view object

min()	Returns the smallest item in an iterable
next()	Returns the next item in an iterable
object()	Returns a new object
oct()	Converts a number into an octal
open()	Opens a file and returns a file object
ord()	Convert an integer representing the Unicode of the specified character
pow()	Returns the value of x to the power of y
print()	Prints to the standard output device
property()	Gets, sets, deletes a property
range()	Returns a sequence of numbers, starting from 0 and increments by 1 (by default)
repr()	Returns a readable version of an object
reversed()	Returns a reversed iterator
round()	Rounds a numbers
set()	Returns a new set object
setattr()	Sets an attribute (property/method) of an object
slice()	Returns a slice object

sorted()	Returns a sorted list
@staticmethod()	Converts a method into a static method
str()	Returns a string object
sum()	Sums the items of an iterator
super()	Returns an object that represents the parent class
tuple()	Returns a tuple
type()	Returns the type of an object
vars()	Returns the __dict__ property of an object
zip()	Returns an iterator, from two or more iterators

- **A simple python program to print the word "Hello World!" on screen:**

```
# Hello World program in Python

print"Hello World!\n"
```

Output on the screen:

```
Hello World!
```

`print "Hello World!\n"` → **implies the statement** that makes provision to print:

```
Hello World!
```

on the screen.

- # **Hello World program in Python** → the statement that implies: comment

- **What will be the output of the following programs?**

(a)

```
print "Hello World!\n"
print "Hello World!\n"
```

Answer:

Hello World!

Hello World!

(b)

```
print "Hello World!"
print "Hello World!"
```

Answer:

Hello World! Hello World!

- **What is the mistake in the following program:**

```
Hello World program in Python
print "Hello World!\n"
```

Answer:

# is not added to the statement:

Hello World program in Python

```
#Hello World program in Python
print "Hello World!\n"
```

- **Program 4.9**

  Python program to add two numbers:

```
number1=2
number2=3
print "The sum of {0} and {1} is {2}".format(number1, number2, float (number1) + float
(number2))
```

Output on the screen:

The sum of 2 and 3 is 5.0

```
print "The sum of {0} and {1} is {2}".format(number1, number2, float (number1) + float
(number2))
```

implies the statement that makes provision to print the output:

The sum of 2 and 3 is 5.0

on the screen.

If you replace the statement:

```
print "The sum of {0} and {1} is {2}".format(number1, number2, float (number1) + float
(number2))
```

by the statement:

```
print "The sum of {1} and {2} is {3}".format(number1, number2, float (number1) + float
(number2))
```

Then

```
Runtime error or IndexError: tuple index out of range
```

will be displayed on the screen.

Dot notation (.) in the statement:

```
print "The sum of {0} and {1} is {2}".format(number1, number2, float (number1) + float
(number2))
```

connects the two statements:

- "The sum of {0} and {1} is {2}"
- format(number1, number2, float (number1) + float (number2))

And if you forget to write dot notation in the statement:

```
print "The difference of {0} and {1} is {2}" .format(number1, number2, float (number1)
+ float (number2))
```

i.e.,

```
print "The difference of {0} and {1} is {2}" format(number1, number2, float (number1) +
float (number2))
```

Then

Syntax error

will be displayed on the console screen.

In the statement:

```
print "The difference of {0} and {1} is {2}" .format(number1, number2, float (number1)
+ float (number2))
```

- {0} → act as a placeholder for number1 in the format method
- {1} → act as a placeholder for number2 in the format method
- {2} → act as a placeholder for float (number1) + float (number2) in the format method

If you want to enter the values for number1 and number2 through **keyboard**, then you need to replace the statements:

```
number1 = 2
number2 = 3
```

by the statements:

```
number1=input(" Please Enter the First Number: ")
number2=input(" Please Enter the Second Number: ")
```

i.e.,

```
number1=input(" Please Enter the First Number: ")
number2=input(" Please Enter the Second Number: ")
print "The sum of {0} and {1} is {2}".format(number1, number2, float (number1) + float
(number2))
```

Output on the screen:

```
Please Enter the First Number:
If you enter 1
Please Enter the Second Number:
If you enter 2
                               The sum of 1 and 2 is 3.0
will be displayed on the screen.
```

Statements:

```
number1 = input(" Please Enter the First Number: ")
number2 = input(" Please Enter the Second Number: ")
```

ask the user to enter two integer numbers and stores the user entered values in variables number 1 and number 2.

- **Program 5.0**

  Python program to subtract two numbers:

```
number1=input(" Please Enter the First Number: ")
number2=input(" Please Enter the Second Number: ")
print "The difference of {0} and {1} is {2}".format(number1, number2, float (number1) -
float (number2))
```

Output on the screen:

Please Enter the First Number:

If you enter 6

Please Enter the Second Number:

If you enter 4

The difference of 6 and 4 is 2.0

will be displayed on the screen.

- **Program 5.1**

  Python program to divide two numbers:

```
number1=input(" Please Enter the First Number: ")
number2=input(" Please Enter the Second Number: ")
print "The division of {0} by {1} yields {2}".format(number1, number2, float (number1)
/ float (number2))
```

Output on the screen:

```
Please Enter the First Number:
If you enter 6
Please Enter the Second Number:
If you enter 2
                             The division of 6 by 2 is 3.0
will be displayed on the screen.
```

- **Program 5.2**

  Python program to multiply two numbers:

```
number1=input(" Please Enter the First Number: ")
number2=input(" Please Enter the Second Number: ")
print "The product of {0} and {1} is {2}".format(number1, number2, float (number1) *
float (number2))
```

Output on the screen:

```
Please Enter the First Number:
If you enter 6
Please Enter the Second Number:
If you enter 2
                             The product of 6 and 2 is 12.0
will be displayed on the screen.
```

- **Program 5.3**

  Python program to find the area of a circle:

```
number1=input(" Please Enter the radius: ")
print "The area of the circle is {0}".format(3.14* number1* number1)
```

Output on the screen:

Please Enter the radius:

If you enter 2

The area of the circle is 12.56

will be displayed on the screen.

- **Program 5.4**

  Python program to find the square root of a number:

```
number1=input(" Please Enter the number: ")
print "The square root of the entered number is {0}".format(number1 ** 0.5)
```

Output on the screen:

```
Please Enter the number:
If you enter 4

The square root of the entered number is 2.0

will be displayed on the screen.

1/2 = 0.5
** --→ implies: exponent operator
number1 ** 0.5 → implies:
number1 ** 1/2 (which implies number1 to the power of 1/2)
```

- **Program 5.5**

  Python program to find the square of a number

```
number1 = input(" Please Enter the number: ")
print "The square of the entered number is {0}".format(number1 * number1)
```

Output on the screen:

Please Enter the number:

If you enter 4

The square of the entered number is 16.0

will be displayed on the screen.

- **Program 5.6**

  Python program to find the incremented and decremented values of the entered number:

```
number1 = input(" Please Enter the number: ")
print "The increment of the entered number is {0}".format(number1 +1)
print "The decrement of the entered number is {0}".format(number1 - 1)
```

Output on the screen:

```
Please Enter the number:
If you enter 6
                        The increment of the entered number is 7
                        The decrement of the entered number is 5
will be displayed on the screen
```

- **What will be the output of the following programs:**

  (a)

```
x = 13
if x < 10:
print ("Good morning")
elif x<12:
print ("Soon time for lunch")
elif x<18:
```

```
print ("Good day")
elif x<22:
print ("Good evening")
else:
print ("Good night")
```

Answer:

Good day

(b)

```
n1 = [0 for i in range(15)]
print (n1)
```

Answer:

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

If you replace the statement:

n1 = [0 for i in range(15)]

by the statement:

n1 = ["computer" for i in range(15)]

Then the output on the screen is:

```
['computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer',
'computer', 'computer', 'computer', 'computer', 'computer', 'computer', 'computer',
'computer']
```

(c)

```
for k in range(1,10):
print(k)
```

Answer:

```
1
2
3
```

```
4
5
6
7
8
9
```

If the statement:

for k in range(1,10):

by the statement:

for k in range(0,10):

Then the **output on the screen** is:

```
0
1
2
3
4
5
6
7
8
9
```

If the statement:

print(k)

is replaced by the statement:

print("computer")

i.e.,

```
for k in range(0,10):
print("computer")
```

Then the **output on the screen** is:

```
computer
computer
computer
computer
computer
computer
computer
computer
computer
computer
```

(d)

```
for day in ["Sunday","Monday","Tuesday","Wednesday","Thursday", "Friday","Saturday"]:
print(day)
```

Answer:

Sunday

Monday

Tuesday

Wednesday

Thursday

Friday

Saturday

(e)

```
n = 10
sum = 0
for i in range(1,n):
  sum = sum + i
print sum
```

Answer:

45

- **range(1,n) means:** generate integers starting from 1 up to, but not including, n.

- **How the execution takes place?**

```
i=1 (sum = 0 because the sum is initialized to 0 in the statement sum = 0)
Is i in range(1,n) true?
Yes, do this
sum = sum + i = 0 +1 =1
Now
```

```
i=2 (now the sum = 1)
Is i in range(1,n) true?
Yes, do this
sum = sum + i = 1 +2 =3
Now
i=3 (now the sum = 3)
Is i in range(1,n) true?
Yes, do this
sum = sum + i = 3 +3 = 6
Now
i=4 (now the sum = 6)
Is i in range(1,n) true?
Yes, do this
sum = sum + i = 6 + 4= 10
Now
i=5 (now the sum = 10)
Is i in range(1,n) true?
Yes, do this
sum = sum + i = 10 + 5= 15
Now
i=6 (now the sum = 15)
Is i in range(1,n) true?
Yes, do this
sum = sum + i = 15 + 6 = 21
Now
i=7 (now the sum = 21)
Is i in range(1,n) true?
Yes, do this
sum = sum + i = 21 + 7 = 28
Now
i=8 (now the sum = 28)
Is i in range(1,n) true?
Yes, do this
sum = sum + i = 28 + 8 = 36
Now
i=9 (now the sum = 36)
Is i in range(1,n) true?
Yes, do this
sum = sum + i = 36 + 9 = 45
stops because range(1,n) means: generate integers starting from i=1 up to, but not
including, i=10.
```

If you want to enter the value for n through the **keyboard**, then the above program should take the form:

```
n =input(" Please Enter the number: ")
sum = 0
for i in range(1,n): sum = sum + i
print sum
```

Output on the screen:

Please Enter the number:

If you enter 10

Then 45 will be outputted on the console screen.

(f)

```
for k in range(1,11):
print("5 x {0} = {1}".format(k, 5*k))
```

Answer:

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 50

If you replace the statement:

for k in range(1,11):

by the statement:

for k in range(0,11):

Then the **output on the screen** is:

```
5 x 0 = 0
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

Python String Methods

Method	Description
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals

isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Joins the elements of an iterable to the end of the string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified value and returns the last position of where it was found

rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case

zfill()	Fills the string with a specified number of 0 values at the beginning

Python List/Array Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Python Dictionary Methods

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Python Tuple Methods

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Python Set Methods

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)

isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Python File Methods

Method	Description
close()	Closes the file
detach()	Returns the separated raw stream from the buffer
fileno()	Returns a number that represents the stream, from the operating

	system's perspective
flush()	Flushes the internal buffer
isatty()	Returns whether the file stream is interactive or not
read()	Returns the file content
readable()	Returns whether the file stream can be read or not
readline()	Returns one line from the file
readlines()	Returns a list of lines from the file
seek()	Change the file position
seekable()	Returns whether the file allows us to change the file position
tell()	Returns the current file position
truncate()	Resizes the file to a specified size
writable()	Returns whether the file can be written to or not
write()	Writes the specified string to the file
writelines()	Writes a list of strings to the file

Python Built-in Exceptions

Exception	Description
ArithmeticError	Raised when an error occurs in numeric calculations
AssertionError	Raised when an assert statement fails
AttributeError	Raised when attribute reference or assignment fails
Exception	Base class for all exceptions
EOFError	Raised when the input() method hits an "end of file" condition (EOF)
FloatingPointError	Raised when a floating point calculation fails
GeneratorExit	Raised when a generator is closed (with the close() method)
ImportError	Raised when an imported module does not exist
IndentationError	Raised when indendation is not correct
IndexError	Raised when an index of a sequence does not exist
KeyError	Raised when a key does not exist in a dictionary
KeyboardInterrupt	Raised when the user presses Ctrl+c, Ctrl+z or Delete
LookupError	Raised when errors raised cant be found
MemoryError	Raised when a program runs out of memory
NameError	Raised when a variable does not exist
NotImplementedError	Raised when an abstract method requires an inherited class to override the method
OSError	Raised when a system related operation causes an error
OverflowError	Raised when the result of a numeric calculation is too large
ReferenceError	Raised when a weak reference object does not exist
RuntimeError	Raised when an error occurs that do not belong to any specific expections
StopIteration	Raised when the next() method of an iterator has no further values

SyntaxError	Raised when a syntax error occurs
TabError	Raised when indentation consists of tabs or spaces
SystemError	Raised when a system error occurs
SystemExit	Raised when the sys.exit() function is called
TypeError	Raised when two different types are combined
UnboundLocalError	Raised when a local variable is referenced before assignment
UnicodeError	Raised when a unicode problem occurs
UnicodeEncodeError	Raised when a unicode encoding problem occurs
UnicodeDecodeError	Raised when a unicode decoding problem occurs
UnicodeTranslateError	Raised when a unicode translation problem occurs
ValueError	Raised when there is a wrong value in a specified data type
ZeroDivisionError	Raised when the second operator in a division is zero

XML (standard)

Status	Published
Year started	1996; 24 years ago
Latest version	1.1 (Second Edition) September 29, 2006; 13 years ago
Editors	Tim Bray Jean Paoli C. M. Sperberg-McQueen Eve Maler François Yergeau John Cowan
Related standards	XML Schema

Domain	Data serialization
Abbreviation	XML
Website	w3.org/TR/xml11
	XML (file format)
Filename extension	`.xml`
Internet media type	`application/xml` `text/xml`
Uniform Type Identifier (UTI)	public.xml
UTI conformation	public.text
Magic number	`<?xml`
Developed by	World Wide Web Consortium
Type of format	Markup language
Extended from	SGML
Extended to	Numerous languages, including XHTML RSS Atom KML
Standard	1.0 (Fifth Edition) (November 26, 2008; 11 years ago) 1.1 (Second Edition) (August 16, 2006; 13 years ago)
Open format?	Yes

- **XML**

Extensible (**extendable**) Markup (**symbols and notations** like <, >, / etc.) Language (which is both human and machine understandable language) is a simple and very flexible text format designed to store and transport data through internet.

- **HTML (Hyper Text Markup Language)** = A text format designed to display data

- **XML to display the output:**

```
note
to people
from steve jobs
message Design is not just what it looks like and feels like. Design is how it works.
```

Answer:

```
<note>
<to> people </to>
<from> steve jobs </from>
<message> Design is not just what it looks like and feels like. Design is how it works.
</message>
</note>
```

- **Note:**

If the statement:

```
<message> Design is not just what it looks like and feels like. Design is how it works.
</message>
```

is replaced by the statement:

```
<Message> Design is not just what it looks like and feels like. Design is how it works.
</message>
```

Then there will be no display of the **output on the console screen**.

The statement:

- <to> people </to> **imply:** element
- <to> **imply:** start tag and </to> **imply:** end tag
- <note>   </note> **is termed:** parent element

```
<to> people </to>
<from> steve jobs </from>
<message> Design is not just what it looks like and feels like. Design is how it works.
</message>
```

are termed: child elements

- **XML to display the output:**

```
Book
Name of the book: Harry Potter
Author: J K. Rowling
Price: $255
Pages: 296
Year: 2002
Edition: 8
```

Answer:

```
<Book>
<Name>:Harry Potter </Name>
<Author>: J K. Rowling </Author>
<Price>: $255 </Price>
<Pages>: 296 </Pages>
<Year>: 2002</Year>
<Edition>: 8 </Edition>
</Book>
```

- **Note:**

What will be the output of the following:

```
<Book>
<Name>: Harry Potter </Name>
<Author> J K. Rowling </Author>
<Price> $255 </Price>
<rowling><Pages> 296 </Pages></rowling>
```

```
<Year> 2002</Year>
<Edition> 8 </Edition>
</Book>
```

Answer:

```
Book
Name of the book: Harry Potter Author: J K. Rowling
Price: 255$ Pages: 296
Year: 2002
Edition: 8
```

Note:

- `<rowling><Pages> 296 </Pages></rowling>` is termed: child element
- `<Pages> 296 </Pages>` is termed: sub child element.

PHP	
Paradigm	Imperative, functional, object-oriented, procedural, reflective
Designed by	Rasmus Lerdorf
Developer	The PHP Development Team, Zend Technologies
First appeared	1995; 25 years ago
Stable release	7.4.7 / June 9, 2020; 6 days ago
Typing discipline	Dynamic, weak

	since version 7.0: Gradual
Implementation language	C (primarily; some components C++)
OS	Unix-like, Windows
License	PHP License (most of Zend engine under Zend Engine License)
Filename extensions	`.php`, `.phtml`, `.php3`, `.php4`, `.php5`, `.php7`, `.phps`, `.php-s`, `.pht`, `.phar`
Website	www.php.net
Major implementations	
Zend Engine, HHVM, Phalanger, Quercus, Parrot	
Influenced by	
Perl, C, C++, Java, Tcl, JavaScript, Hack	
Influenced	
Hack	

PHP / Hypertext Preprocessor (designed by an **Greenlandic-Danish programmer** "Rasmus Lerdorf" in 1994 – as an efficient alternative to other scripting languages like Ruby, Perl and Microsoft’s ASP) is an relatively free (**not licensed by a major corporation**) popular efficient server side programming language (and relatively easy one to master and quick to learn) that carries out common website duties like accepting passwords, authenticating users, and managing forum posts and guest books. Without ***PHP*** Facebook, Yahoo, Google wouldn't have existed.

PHP Keywords				
__halt_compiler()	abstract	and	array()	as
break	callable (as of PHP 5.4)	case	catch	class
clone	const	continue	declare	default
die()	do	echo	else	elseif
empty()	enddeclare	endfor	endforeach	endif
endswitch	endwhile	eval()	exit()	extends
final	finally (as of PHP 5.5)	fn (as of PHP 7.4)	for	foreach
function	global	goto (as of PHP 5.3)	if	implements
include	include_once	instanceof	insteadof (as of PHP 5.4)	interface
isset()	list()	namespace (as of PHP 5.3)	new	or
print	private	protected	public	require
require_once	return	static	switch	throw
trait (as of PHP 5.4)	try	unset()	use	var
while	xor	yield (as of PHP 5.5)	yield from (as of PHP 7.0)	

```
<?php

// This is a single-line comment

# This is also a single-line comment


?>
```

- **A simple PHP program to print the word "Hello World!" on screen:**

```
<?php

echo "Hello World!";

?>
```

In the above example:

- <?php and ?> denote: opening and closing tags within which the execution of php codes takes place.
- echo "Hello World!"; → denote: the statement that makes provision to print the **output**:

Hello World!

on the screen.

Even If you replace the statement:

```
echo "Hello World!";
```

by the statement:

```
print "Hello World!";
```

i.e.,

```
<?php
print "Hello World!";
?>
```

There will be **no change in the output** on the screen (i.e., echo and print are more or less the same. They are both used to output data to the console screen).

If replace the opening tag <?php by <?

Then the output on the screen is:

```
<?
print "Hello World!";
?>
```

i.e., the entire program will be reflected on the console screen.

Even if you write the statement:

```
print ( "Hello World!");
```

instead of the statement:

print "Hello World!";

There will be **no change in the output** on the screen.

- **Note:**

  Even if you omit the **closing tag**

```
?>
```

in the above program. There will be **no change in the output** on the screen. But sometimes it reflects error. So omission of ?> is discouraged.

- **Program 5.7**

a)

```
<?php

echo "Hello World!";
echo "Hello World!";

?>
```

Output on the screen:

Hello World!Hello World!

b)

```
<?php

echo "\n Hello World!";
echo "\n Hello World!";

?>
```

Output on the screen:

Hello World!

Hello World!

c)

```
<?php

echo "Hello World!";
echo "\t Hello World!";

?>
```

Output on the screen:

```
Hello World!    Hello World!
```

- **Program 5.8**

  PHP program to add two numbers:

```
<?php

$num1 =1;
$num2=5;
$sum = $num1 + $num2;
echo "Sum of the two numbers is : $sum";

?>
```

Output on the screen:

```
Sum of the two numbers is : 6
```

- Equal sign implies: storage operator.

The statements:

```
$num1 =1;
$num2 =5;
$sum = $num1 + $num2;
```

imply: that we are creating the variables $num1, $num2 and $sum and storing the values for created variables (i.e., 1 for $num1, 5 for $num1 and $sum for $num1 + $num2).

The statement:

```
echo "Sum of the two numbers is : $sum";
```

make provision to print the output:

Sum of the two numbers is : 6 (which is 1+5)

on the screen.

Suppose if you omit the $ **symbol** before a variable name (whose purpose is to make it clear that the word following the symbol $ is a variable – the symbol $ distinguishes **variables** from other things) in the above program i.e.,

```
<?php
num1 = 1;
num2  =5;
sum = num1 + num2;
echo "Sum of the two numbers is : sum";
?>
```

Then

PHP Parse error: syntax error, unexpected ' = '

will be displayed on the console screen.

If you want to supply the integer values for $num1 and $num2 through the **key board**, then the statements:

```
$num1 =1;
$num2 =5;
```

should be replaced by the statements:

```
echo "Please enter the first number : ";

fscanf(STDIN, “%d\n”, $num1);

echo "Please enter the second number : ";

fscanf(STDIN, “%d\n”, $num2);
```

i.e.,

```
<?php
echo "Please enter the first number : ";
fscanf(STDIN, "%d\n", $num1);
echo "Please enter the second number : ";
fscanf(STDIN, "%d\n", $num2);
$sum = $num1 + $num2;
echo "Sum of the two numbers is : $sum";
?>
```

Output on the screen:

Please enter the first number :

If you enter 5

Please enter the second number :

If you enter 6

Sum of the two numbers is: 11 will be outputted on the screen.

- fscanf(STDIN, "%d\n", $num1); → denote: the statement that makes provision to enter an integer value for $num1 through the keyboard.
- fscanf(STDIN, "%d\n", $num2); → denote: the statement that makes provision to enter an integer value for $num2 through the keyboard.

Format string %d in the statement:

```
fscanf(STDIN, "%d\n", $num1);

or in the statement:

fscanf(STDIN, "%d\n", $num2);
```

tells the input function fscanf() to read the number entered through the keyboard (which is a integer) and STDIN (which stands for **Standard input**) means feed the number entered through the keyboard into the program.

If you want to enter the floating values (say 1.6 and 1.9) for $num1 and $num2, then your program should take the form:

```
<?php
echo "Please enter the first number : ";
fscanf(STDIN, "%f\n", $num1);
echo "Please enter the second number : ";
fscanf(STDIN, "%f\n", $num2);
$sum = $num1 + $num2;
echo "Sum of the two numbers is : $sum";
?>
```

Output on the screen:

Please enter the first number :

If you enter 1.6

Please enter the second number :

If you enter 1.9

Sum of the two numbers is: 3.5

will be outputted on the screen.

- **Program 5.9**

  PHP program to subtract two numbers:

```
<?php
$num1 =5;
$num2=1;
$sub = $num1 - $num2;
echo "difference of the two numbers is : $sub";
?>
```

Output on the screen:

difference of the two numbers is : 4

- **Program 6.0**

  PHP program to divide two numbers:

```
<?php

$num1 =6;
$num2=2;
$div = $num1 / $num2;
echo "the division of two numbers is : $div";

?>
```

Output on the screen:

the division of two numbers is : 3

- **Program 6.1**

  PHP program to multiply two numbers:

```
<?php
$num1 = 6;
$num2 = 2;
$mult = $num1 * $num2;
echo "the product of two numbers is : $mult";
?>
```

Output on the screen:

the product of two numbers is : 12

- **Program 6.2**

  PHP program to find the area of a circle:

```
<?php

$radius = 2.0;
$pi = 3.14159;
$area = $pi * $radius * $radius;
echo("\n radius = $radius centimeter");
echo("\n area = $area centimeter square");
?>
```

Output on the screen:

```
radius = 2 centimeter

area = 12.56636 centimeter square
```

- **Program 6.3**

PHP program to find the square root of a number

```php
<?php
$num1 = 4.0;
$num2 = sqrt($num1);
echo("The square root of a number = $num2");
?>
```

Output on the screen:

```
The square root of a number = 2
```

- **Program 6.4**

PHP program to find the square of a number

```php
<?php
$num1 = 2.0;
$num2 = $num1 * $num1;
echo("\n the square of a number = $num2");
?>
```

Output on the screen:

```
the square of a number = 4
```

If the statement:

```php
$num2 = $num1 * $num1;
```

is replaced by:

```php
$num2 = pow(($num1), 2);
```

i.e.,

```
<?php
$num1 = 2.0;
$num2 = pow(($num1), 2);
echo("\n the square of a number = $num2");
?>
```

Then there will be **no change in the output** on the screen i.e.,

the square of a number = 4

will be outputted on the screen.

This means:

```
$num2 = pow(($num1), 2); is the same as $num2 = $num1 * $num1;
```

- **Program 6.5**

  PHP program to find the cube root of a number

```
<?php
$num1 = 6.0;
$num2 = pow(($num1), 1/3);
echo("\n the cube root of a number = $num2");
?>
```

Output on the screen:

the cube root of a number = 1.8171205928321

- **Program 6.6**

  PHP program to round off a number

```
<?php
$num1 = 4.5;
$num2 = round ($num1);
echo("\n the round off of a number = $num2");
?>
```

Output on the screen:

the round off of a number = 5

- **Program 6.7**

  PHP program to find the incremented and decremented values of two numbers.

```
<?php
$num1 =2;
$num2=3;
$num3 = $num1 +1;
$num4 = $num1 - 1;
$num5 = $num2 +1;
$num6 = $num2 - 1;
echo ("\n The incremented value of $num1 = $num3 ");
echo ("\n The decremented value of $num1 = $num4 ");
echo ("\n The incremented value of $num2 = $num5 ");
echo ("\n The decremented value of $num2 = $num6 ");
?>
```

Output on the screen:

```
The incremented value of 2 = 3
The decremented value of 2 = 1
The incremented value of 3 = 4
The decremented value of 3 = 2
```

- **Program 6.8**

  PHP program to find the greatest of two numbers using *if – else* statement

  The syntax of *if – else* statement is:

  if (this condition is true)

  {

  print this statement;

  }

  else

  {

  print this statement;

}

```
<?php
$x = 4.5;
$y=5;
if($x>$y){
echo (" x is greater than y");
} else {
echo (" y is greater than x");
}
?>
```

Output on the screen:

y is greater than x

If the above program **is rewritten as:**

```
<?php
$x = 4.5;
$y=5;
if($x>$y){
echo (" $x is greater than $y");
} else {
echo (" $y is greater than $x");
}
?>
```

Then the output on the screen:

5 is greater than 4.5

- **Program 6.9**

  PHP program to find the greatest of three numbers using *else if* statement

  The syntax of *else if* statement is:

  if (this condition is true)

  {

  print this statement;

  }

  else if(this condition is true)

  {

```
print this statement;
}
else
{
print this statement;
}
```

```php
<?php
$x = 4.5;
$y = 5;
$z = 6;
if($x > $y && $x > z){
echo (" $x is greater than $y and $z");
} else if ($y>$z && $y>x){
echo (" $y is greater than $x and $z");
}
else
{
echo (" $z is greater than $x and $y");
}
?>
```

Output on the screen:

```
6 is greater than 4.5 and 5
```

- **Program 7.0**

  PHP program to print the first ten natural numbers using for loop statement

```php
<?php
for ($i=1; $i<=10; $i++)
echo (" \n $i");
?>
```

Output on the screen:

```
1

2

3

4

5
```

6

7

8

9

10

for ($i=1; $i<=10; $i++) denote the for loop statement for PHP and the syntax of the for loop statement is:

```
for (initialization; condition; increment)
```

Here:

- $i=1 denote initialization (i.e., from where to start)
- $i<=10 denote the condition (i.e., stop when the number 10 is reached)
- $i++ imply increment (which tells the value of **$i** to increase by 1 each time the loop is executed) and $i++ is the same as $i+1.

Since the initialization i.e., $i=1

The statement:

echo (" \n $i");

make provision to print the output:

1

on the screen.

After this, the following execution takes place:

```
Now,
$i= 1
Is the condition ($i<=10) is true?
Yes because $i=1
Do this
$i= 1+1 = 2
The statement
echo (" \n $i");
```

```
make provision to print the output:
2
Now,
$i= 2
Is the condition ($i<=10) is true?
Yes because $i=2
Do this
$i= 2+1 = 3
The statement
echo (" \n $i");
make provision to print the output:
3
Now,
$i= 3
Is the condition ($i<=10) is true?
Yes because $i=3
Do this
$i= 3+1 = 4
The statement
echo (" \n $i");
make provision to print the output:
4
Now,
$i= 4
Is the condition ($i<=10) is true?
Yes because $i=4
Do this
$i= 4+1 = 5
The statement
echo (" \n $i");
make provision to print the output:
5
Now,
$i= 5
Is the condition ($i<=10) is true?
Yes because $i=5
Do this
$i= 5+1 = 6
The statement
echo (" \n $i");
make provision to print the output:
6
Now,
$i= 6
Is the condition ($i<=10) is true?
Yes because $i=6
Do this
$i= 6+1 = 7
The statement
echo (" \n $i");
make provision to print the output:
7
Now,
$i= 7
Is the condition ($i<=10) is true?
Yes because $i=7
Do this
$i= 7+1 = 8
The statement
echo (" \n $i");
make provision to print the output:
8
Now,
$i= 8
Is the condition ($i<=10) is true?
Yes because $i=8
Do this
```

```
$i= 8+1 = 9
The statement
echo (" \n $i");
make provision to print the output:
9
Now,
$i= 9
Is the condition ($i<=10) is true?
Yes because $i=9
Do this
$i= 9+1 = 10
The statement
echo (" \n $i");
make provision to print the output:
10
stop because the condition $i<=10 is achieved.
```

If you replace the statement

for ($i=1; $i<=10; $i++)

by the statement

for ($i=1; $i= =10; $i++)

Then there will be **no display of output** on the screen.

If the statement

for ($i=1; $i<=10; $i++)

is replaced by the statement

for ($i=1; $i=10; $i++)

Then the output on the screen is:

```
10
10
10
10
10
10
10
10
10
10
10
10
10 …… continues
```

- **Program 7.1**

PHP program to print the first ten natural numbers using while loop statement

The syntax of while loop statement is:

while (this is the condition)

{

execute this statement;

}

```
<?php
$i = 1;
while($i <= 10) {
echo "\n $i ";
$i++;
}
?>
```

Output on the screen:

```
1
2
3
4
5
6
7
8
9
10
```

- **Program 7.2**

PHP program to print the first nine natural numbers using do while loop statement

The syntax of do while loop statement is:

do

{

execute this statement;

}

while(this is the condition);

```
<?php
$i = 1;
do {
echo "\n $i ";
$i++;
} while($i <= 9);
?>
```

Output on the screen:

```
1

2

3

4

5

6

7

8

9
```

- **Program 7.3**

  PHP program to print the average of the first10 numbers using for loop statement

```
<?php
$i;
$avg;
$sum = 0;
for( $i=1; $i<=10; $i++)
$sum = $sum + $i;
$avg = $sum/10;
echo "\n sum of the first 10 numbers = $sum ";
echo"\n average of the first 10 numbers = $avg  ";
?>
```

Output on the screen:

```
sum of the first 10 numbers = 55

average of the first 10 numbers = 5.5
```

- **Program 7.4**

  Switch case method

  a)

```
<?php
$ch ='3';
switch($ch)
{
case '1':
echo "Red";
break;
case '2':
echo "White";
break;
case  '3':
echo "Yellow";
break;
case '4':
echo "Green";
break;
default:
echo "Error";
break;
}
?>
```

Output on the screen:

```
Yellow
```

b)

```
<?php
$ch ='birds';
switch($ch)
{
case 'animal':
echo "elephant";
break;
case 'reptiles':
echo "crocodile";
break;
case 'birds':
echo "parrot";
break;
case 'mammals':
echo "cow";
break;
default:
echo "Error";
break;
}
?>
```

Output on the screen:

parrot

- **Program 7.5**

Addition of two numbers using PHP function

```
<?php
function addition($a, $b) {
return $a + $b;
}
$sum = addition(4, 3);
echo "the sum of two numbers = $sum ";
?>
```

Output on the screen:

```
the sum of two numbers = 7
```

JavaScript	
Paradigm	event-driven, functional, imperative
Designed by	Brendan Eich initially, plus other key contributors to the ECMAScript specification
First appeared	December 4, 1995; 24 years ago

Stable release	ECMAScript 2019 / June 2019; 1 year ago
Preview release	ECMAScript 2020
Typing discipline	Dynamic, duck

Filename extensions	`.js`
	`.mjs`

Major implementations	
V8, JavaScriptCore, SpiderMonkey, Chakra	
Influenced by	
AWK, C, HyperTalk, Java, Lua, Perl, Python, Scheme, Self	
Influenced	
ActionScript, AtScript, CoffeeScript, Dart, JScript .NET, LiveScript, Objective-J, Opa, QML, Raku, TypeScript	

JavaScript / Jscript (designed by an American technologist and co-founder of the Mozilla project, the Mozilla Foundation, and the Mozilla Corporation "***Brendan Eich***" and it was developed under the name Mocha, the language was officially called **LiveScript** when it first shipped in beta releases of Netscape Navigator 2.0 in September 1995, but it was renamed *JavaScript*) is an relatively popular object-oriented scripting interpreted programming language embedded in high level programming language of Hypertext Markup Language (the language that websites are rendered in and basically, everything you and your readers see on the "*front- end*" is HTML), commonly abbreviated as **HTML pages** – primarily used to design interactive websites with dynamic content and perform functions that the HTML cannot do, because of its reliability, simplicity and easy to understand, easy to use, write, modify and debug and quick to learn.

JavaScript Keywords:

abstract	arguments	boolean	break
byte	case	catch	char
const	continue	debugger	default
delete	do	double	else
eval	false	final	finally
float	for	function	goto
if	implements	in	instanceof
int	interface	let	long
native	new	null	package
private	protected	public	return
short	static	switch	synchronized
this	throw	throws	transient
true	try	typeof	var
void	volatile	while	with

lass	enum	export	extends
import	super		

A Simple JavaScript program to print the word "Hello World!" on screen:

```
<!DOCTYPE html>
<html>
<body>
<script>

document.write("Hello World!");

</script>
</body>
</html>
```

In the above example:

```
<!DOCTYPE html>

<html>

<body>



</body>

</html>
```

denote: **HTML** document and

<script>

document.write("Hello World!");

</script>

denote **JavaScript** code to print out the string "Hello World!" on the screen. In the above example, document.write () → denote function or method which print out the string "Hello World!" on the console screen.

If you fail to include the **tag**

<script>

</script> in the above example i.e.,

```
<!DOCTYPE html>
<html>
<body>
document.write("Hello World!");
</body>
</html>
```

Then

document.write("Hello World!");

will be outputted on the screen instead of Hello World!.

If you replace the text within the double quotation marks by the word "hello" i.e.,

```
<!DOCTYPE html>
<html>
<body>
document.write("hello");
</body>
</html>
```

Then the output on the screen is:

hello

Even if you write

```
<SCRIPT>

          </SCRIPT>
```

instead of

```
<script>

        </script>
```

```
Document.write("hello");
```

instead of

```
document.write("hello");
```

There will be **no change in the output** on the screen.

```
<!DOCTYPE html>
<html>
<body>
<script type="text/javascript">
document.write("Hello World!");
</script>
</body>
</html>


There will be no change in the output on the screen i.e., Hello World will be displayed
on the console screen.
```

- **Program 7.6**

  a)

```
<!DOCTYPE html>
<html>
<body>
<script>

document.write("Hello World!");
document.write("Hello World!");

</script>
</body>
</html>
```

Output on the screen:

Hello World!Hello World!

b)

```
<!DOCTYPE html>
<html>
<body>
<script>

document.write("\n Hello World!");
document.write("\n Hello World!");

</script>
</body>
</html>
```

Output on the screen:

Hello World!   Hello World!

c)

```
<!DOCTYPE html>
<html>
<body>
<script>
document.write("<br>Hello World!</br>");
document.write("Hello World!");
</script>
</body>
</html>
```

Output on the screen:

Hello World!

Hello World!

d)

```
<!DOCTYPE html>
<html>
<body>
<script>
document.write("<br>Hello World!</br>");
document.write("<br>Hello World!</br>");
</script>
</body>
</html>
```

Output on the screen:

Hello World!

Hello World!

e)

```
<!DOCTYPE html>
<html>
<body>
<script>
document.write("<b><br>Hello World!</br></b>");
document.write("<br>Hello World!</br>");
</script>
</body>
</html>
```

Output on the screen:

Hello World!

Hello World!

f)

```
<!DOCTYPE html>
<html>
<body>
<script>
document.write("<i><b><br>Hello World!</br></b></i>");
document.write("<br>Hello World!</br>");
```

```
</script>
</body>
</html>
```

Output on the screen:

Hello World!

Hello World!

- **Program 7.7**

JavaScript program to add two numbers:

```
<!DOCTYPE html>
<html>
<body>
<p>A typical addition operation adds two numbers and produces a new number.</p>
<script>
var x ;
var y;
var  z;
x =100;
y = 200;
z = x+ y;
document.write(" The sum of two numbers is:     " + z);
</script>
</body>
</html>
```

Output on the screen:

```
A typical addition operation adds two numbers and produces a new number.
```

The sum of two numbers is: 300

The statements:

```
var x ;
var y ;
var z ;
```

imply: that we are creating the variables x, y and z.

- Equal sign implies: storage operator

The statements:

```
x=100;
y = 200;
z = x+ y;
```

imply: that we are storing the values to the created variables (i.e., we are storing the value 100 for x and 200 for y and x + y for z).

The statement:

```
document.write(" The sum of two numbers is:     " + z);
```

make provision to print the output:

The sum of two numbers is: 300

on the screen.

In the statement:

```
document.write(" The sum of two numbers is:     " + z);
```

There are two strings:

- The sum of two numbers is:
- z

plus operator (+) functions as the concatenation operator (concatenation means connecting two statements to produce a single statement) – which (**here**) concatenates the string:

- "The sum of two numbers is: " and the string:
- "z (which is 100 + 200 = 300) " – producing a String statement

The sum of two numbers is: 300, which is displayed on the screen as the result.

The statement:

```
<p>A typical addition operation adds two numbers and produces a new number.</p>
```

make provision to print the output:

A typical arithmetic operation takes two numbers and produces a new number.

on the screen.

If the statement:

```
<p>A typical addition operation adds two numbers and produces a new number.</p>
```

is replaced by the statement:

```
<h1>A typical addition operation adds two numbers and produces a new number.</h1>
```

Then the output on the screen is:

A typical arithmetic operation takes two numbers and produces a new number.

The sum of two numbers is: 300

- **Program 7.8**

  JavaScript program to subtract two numbers:

```
<!DOCTYPE html>
<html>
<body>
<h1>A typical subtraction operation subtracts two numbers and produces a new
number.</h1>
<script>
var x ;
var y;
var z;
x=300;
y = 200;
z = x- y;
document.write(" The difference of two numbers is:     " + z);
</script>
</body>
</html>
```

Output on the screen:

A typical subtraction operation subtracts two numbers and produces a new number.

The difference of two numbers is: 100

- **Program 7.9**

  JavaScript program to divide two numbers:

```
<!DOCTYPE html>
<html>
<body>
<p>A typical division operation divides two numbers and produces a new number.</p>
<script>
var x ;
var y;
var z;
x=300;
y = 200;
z = x/ y;
document.write(" The division of two numbers is:     " + z);
</script>
</body>
</html>
```

Output on the screen:

A typical division operation divides two numbers and produces a new number.

The division of two numbers is: 1.5

- **Program 8.0**

JavaScript program to multiply two numbers:

```
<!DOCTYPE html>
<html>
<body>
<p>A typical multiplication operation multiplies two numbers and produces a new
number.</p>
<script>
var x ;
var y;
var z;
x=300;
y = 200;
z = x* y;
document.write(" The multiplication of two numbers is:     " + z);
</script>
</body>
</html>
```

Output on the screen:

A typical multiplication operation multiplies two numbers and produces a new number.

The multiplication of two numbers is: 60000

- **Program 8.1**

  JavaScript program to find the area of a circle

```
<!DOCTYPE html>
<html>
<body>
<script>
var r ;
var area;
r=3;
area = 3.14* r* r;
document.write(" The area of the circle is:     " + area +"\n centimeter square");
</script>
</body>
</html>
```

Output on the screen:

The area of the circle is: 28.27 centimeter square

- **Program 8.2**

  JavaScript program to find the square root of a number

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var z;
x=4;
z = Math.sqrt(x);
document.write(" The square root of a number z is: " + z);
</script>
</body>
</html>
```

Output on the screen:

The square root of a number z is: 2

- **Program 8.3**

JavaScript program to find the square of a number

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var z;
x=4;
z = x*x;
document.write(" The square of a number z is: " + z);
</script>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var z;
x=4;
z = Math.pow((x), 2);
document.write(" The square of a number z is: " + z);
</script>
</body>
</html>
```

Output on the screen:

The square of a number z is: 16

- **Program 8.4**

JavaScript program to find the cube root of a number

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var z;
x=4;
z = Math.cbrt(x);
document.write(" The cube root of a number z is: " + z);
```

```
</script>
</body>
</html>
```

Output on the screen:

```
The cube root of a number z is: 1.5874010519681996
```

- **Program 8.5**

  JavaScript program to round off a number

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var z;
x=4.5;
z = Math.round(x);
document.write(" The round off a number z is: " + z);
</script>
</body>
</html>
```

Output on the screen:

```
The round off a number z is: 5
```

- **Program 8.6**

  JavaScript program to find the incremented and decremented values of two numbers.

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var y;
var z;
var p;
var a;
var b;
x=4;
y=6;
z=x+1;
```

```
p=x-1;
a = y+1;
b= y-1;
document.write(" The incremented value of  x  is: " + z);
document.write(" The decremented value of  x  is: " + p);
document.write(" The incremented value of  y  is: " + a);
document.write(" The decremented value of  y  is: " + b);
</script>
</body>
</html>
```

Output on the screen:

```
The incremented value of x is: 5 The decremented value of x is: 3 The incremented value
of y is: 7 The decremented value of y is: 5
```

- **Program 8.7**

JavaScript program to find the greatest of two numbers using if – else statement

The syntax of if – else statement is:

if (this condition is true)

{

print this statement using document.write function;

}

else

{

print this statement using document.write function;

}

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var y;
x=4;
y=6;
if(x>y){
document.write(" x is greater than y");
} else {
```

```
document.write(" y is greater than x");
}
</script>
</body>
</html>
```

Output on the screen:

```
y is greater than x
```

- **Program 8.8**

JavaScript program to find the greatest of three numbers using *else-if* statement

The syntax of *else-if* statement is:

```
if (this condition is true)
{
print this statement using document.write function;
}
else if(this condition is true)
{
print this statement using document.write function;
}
else
{
print this statement using document.write function;
}
```

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var y;
var z;
x=4;
y=6;
z=12;
if(x>y&&x>z){
document.write(" x is greater than y and z");
} else if (y>x&&y>z)
{
document.write(" y is greater than x and z");
} else {
document.write(" z is greater than x and y");
}
</script>
</body>
</html>
```

Output on the screen:

```
z is greater than x and y
```

- **Program 8.9**

JavaScript program to print the first ten natural numbers using for loop statement

```
<!DOCTYPE html>
<html>
<body>
<script>
var i ;
for (i=1; i<=10; i++)
document.write("" +  i);
</script>
</body>
</html>
```

Output on the screen:

```
1 2 3 4 5 6 7 8 9 10
```

If you replace the statement:

```
for (i=1; i<=10; i++)
```

by the statement:

```
for (i=1; i= =10; i++)
```

Then there will be **no display of output** on the screen.

If the statement:

```
document.write(" " +  i);
```

is replaced by the statement

```
document.write("<br> </br>" + i);
```

Then the output on the screen is:

```
1

2

3

4

5

6

7

8

9

10
```

- **Program 9.0**

JavaScript program to print the first ten natural numbers using while loop statement

The syntax of while loop statement is:

while (this is the condition)

{

execute this statement;

}

```
<!DOCTYPE html>
<html>
<body>
<script>
var i=1 ;
while (i<=10)
document.write(" " +  i++);
</script>
</body>
</html>
```

Output on the screen:

```
1 2 3 4 5 6 7 8 9 10
```

- **Program 9.1**

JavaScript program to print the first nine natural numbers using do while loop statement

The syntax of do while loop statement is:

do

{

execute this statement;

}

while(this is the condition);

```
<!DOCTYPE html>
<html>
<body>
<script>
var i=1 ;
do{
document.write(" " +  i++);
} while (i<10)
</script>
</body>
</html>
```

Output on the screen:

```
1 2 3 4 5 6 7 8 9
```

- **Program 9.2**

  JavaScript program to print the average of the first10 numbers using for loop statement

```
<!DOCTYPE html>
<html>
<body>
<script>
var i, avg, sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
document.write("<br> sum of the first 10 numbers = </br>" + sum);
document.write("<br> average of the first10 numbers = </br>" + avg);
</script>
</body>
</html>
```

Output on the screen:

```
sum of the first 10 numbers =

55

average of the first10 numbers =

5.5
```

- **Program 9.3**

  Switch case method

  a)

```
<!DOCTYPE html>
<html>
<body>
<script>
var ch ='2';
switch(ch)
{
case '1':
document.write("Red");
break;
case '2':
document.write("White");
break;
case '3':
document.write("Yellow");
break;
case '4':
document.write("Green");
break;
default:
document.write("Error");
break;
}
</script>
</body>
</html>
```

Output on the screen:

```
White
```

b)

```
<!DOCTYPE html>
<html>
<body>
<script>
var ch ='animal';
switch(ch)
{
case 'animal':
document.write("elephant");
break;
case 'reptiles':
document.write("crocodile");
break;
case 'birds':
document.write("parrot");
break;
case 'mammals':
document.write("cow");
break;
default:
document.write("Error");
break;
}
</script>
</body>
</html>
```

Output on the screen:

elephant

- **Program 9.4**

  Addition of two numbers using JavaScript function

```
<!DOCTYPE html>
<html>
<body>
<script>
function addition(a, b) {
return a + b;
}
document.write(" " + addition(4, 3));
</script>
</body>
</html>
```

Output on the screen:

7

- **What is the mistake in the following program:**

```
<!DOCTYPE html>
<html>
<body>
<script>
function addition(a, b) {
return a % b;
}
document.write("" + function(4, 3));
</script>
</body>
</html>
```

Answer: There is mistake in the above program.

return a % b;

is written instead of:

return a + b;

The program:

```
<html>
<head>
<script>
function addNumbers()
{
var x = parseInt(document.getElementById("value1").value);
var y = parseInt(document.getElementById("value2").value);
var z = document.getElementById("answer");
z.value = x + y;
}
</script>
</head>
<body>
value1 = <input type ="text" id="value1"/>
value2 = <input type ="text" id="value2"/>
<input type="button" value="Click here" onclick="addNumbers()"/>
Answer = <input type="text" id = "answer"/>
</body>
</html>
```

demonstrates how to get two inputs from the user (i.e., value1 and value2) and have a button (i.e., click here) on the page to call a function:

function addNumbers()

to process the inputs and output the answer on the web screen.

```
var x = parseInt(document.getElementById("value1").value);
```

denote: that we are creating the **variable** x and assigning it a value = parseInt(document.getElementById("value1").value)

```
parseInt(document.getElementById("value1").value) --→ get the value (i.e., get the
integer value) for x entered in the input field which is defined with id value1.
```

```
parseInt(document.getElementById("value2").value) --→ get the value (i.e., get the
integer value) for y entered in the input field which is defined with id value2.
```

After entering the values for x and y in the input fields – you click on the button "click here" – after button click

"on click" in the statement:

```
<input type="button" value="Click here" onclick="addNumbers()"/>
```

call the function addNumbers() → within which

var z = document.getElementById("answer");

z.value = x + y;

is executed — i.e., the entered values for x and y are added and the answer is entered in the input field which is defined with id answer.

If you want to enter the floating values (say if you want enter 1.63 and 1.569) for x and y in the input fields, then you need to replace the statements:

```
var x = parseInt(document.getElementById("value1").value);
var y = parseInt(document.getElementById("value2").value);
```

in the above program by the statements:

```
var x = parseFloat(document.getElementById("value1").value);
var y = parseFloat(document.getElementById("value2").value);
```

- **What will be the output of the following programs:**

a)

```
<html>
<head>
<script>
function area()
{
var r = parseFloat(document.getElementById("value1").value);
var a = document.getElementById("answer");
a.value = 4*3.14* r* r;
}
</script>
</head>
<body>
radius = <input type="text" id="value1"/>
<input type="button" value="Click here" onclick="area()"/>
Area of the circle = <input type="text" id = "answer"/>
</body>
</html>
```

b)

```
<html>
<head>
<script>

function ifelse()
{
var x = parseInt(document.getElementById("value1").value);

var y = parseInt(document.getElementById("value2").value);

var z = parseInt(document.getElementById("value3").value);

if(x>y&&x>z) {

document.write(" x is greater than y and z");

} else if (y>x&&y>z) {

document.write(" y is greater than x and z");
} else {
document.write(" z is greater than x and y");
}
}
</script>
</head>
<body>
x = <input type="text" id="value1"/>

y = <input type="text" id="value2"/>
```

```
z = <input type="text" id="value3"/>

<input type="button" value="Click here" onclick="ifelse()"/>

</body>
</html>
```

c)

```
<html>
<head>
<script>
function switchcase()
{
var ch = parseInt(document.getElementById("value1").value);

switch(ch)
{
case 1:
document.write("Red");
break;
case 2:
document.write("White");
break;
case 3:
document.write("Yellow");
break;
case 4:
document.write("Green");
break;
default:
document.write("Error");
break;
}
}
</script>
</head>
<body>
ch = <input type="text" id="value1"/>

<input type="button" value="Click here" onclick="switchcase()"/>

</body>
</html>
```

> "Object-oriented programming offers a sustainable way to write spaghetti code. It lets you accrete programs as a series of patches."
>
> — Paul Graham

Python Glossary

Feature	Description
Indentation	Indentation refers to the spaces at the beginning of a code line
Comments	Comments are code lines that will not be executed
Multi Line Comments	How to insert comments on multiple lines
Creating Variables	Variables are containers for storing data values
Variable Names	How to name your variables
Assign Values to Multiple Variables	How to assign values to multiple variables
Output Variables	Use the print statement to output variables
String Concatenation	How to combine strings
Global Variables	Global variables are variables that belongs to the global scope
Built-In Data Types	Python has a set of built-in data types
Getting Data Type	How to get the data type of an object

Setting Data Type	How to set the data type of an object
Numbers	There are three numeric types in Python
Int	The integer number type
Float	The floating number type
Complex	The complex number type
Type Conversion	How to convert from one number type to another
Random Number	How to create a random number
Specify a Variable Type	How to specify a certain data type for a variable
String Literals	How to create string literals
Assigning a String to a Variable	How to assign a string value to a variable
Multiline Strings	How to create a multi line string
Strings are Arrays	Strings in Python are arrays of bytes representing Unicode characters
Slicing a String	How to slice a string
Negative Indexing on a String	How to use negative indexing when accessing a string

String Length	How to get the length of a string
Check In String	How to check if a string contains a specified phrase
Format String	How to combine two strings
Escape Characters	How to use escape characters
Boolean Values	True or False
Evaluate Booleans	Evaluate a value or statement and return either True or False
Return Boolean Value	Functions that return a Boolean value
Operators	Use operator to perform operations in Python
Arithmetic Operators	Arithmetic operator are used to perform common mathematical operations
Assignment Operators	Assignment operators are use to assign values to variables
Comparison Operators	Comparison operators are used to compare two values
Logical Operators	Logical operators are used to combine conditional statements
Identity Operators	Identity operators are used to see if two objects are in fact the same object

Membership Operators	Membership operators are used to test is a sequence is present in an object
Bitwise Operators	Bitwise operators are used to compare (binary) numbers
Lists	A list is an ordered, and changeable, collection
Access List Items	How to access items in a list
Change List Item	How to change the value of a list item
Loop Through List Items	How to loop through the items in a list
Check if List Item Exists	How to check if a specified item is present in a list
List Length	How to determine the length of a list
Add List Items	How to add items to a list
Remove List Items	How to remove list items
Copy a List	How to copy a list
Join Two Lists	How to join two lists
Tuple	A tuple is an ordered, and unchangeable, collection
Access Tuple Items	How to access items in a tuple

Change Tuple Item	How to change the value of a tuple item
Loop List Items	How to loop through the items in a tuple
Check if Tuple Item Exists	How to check if a specified item is present in a tuple
Tuple Length	How to determine the length of a tuple
Tuple With One Item	How to create a tuple with only one item
Remove Tuple Items	How to remove tuple items
Join Two Tuples	How to join two tuples
Set	A set is an unordered, and unchangeable, collection
Access Set Items	How to access items in a set
Add Set Items	How to add items to a set
Loop Set Items	DETTE KAPITTELET MANGLER
Check if Set Item Exists	DETTE KAPITTELET MANGLER
Set Length	How to determine the length of a set
Remove Set Items	How to remove set items

Join Two Sets	How to join two sets
Dictionary	A dictionary is an unordered, and changeable, collection
Access Dictionary Items	How to access items in a dictionary
Change Dictionary Item	How to change the value of a dictionary item
Loop Dictionary Items	How to loop through the items in a tuple
Check if Dictionary Item Exists	How to check if a specified item is present in a dictionary
Dictionary Length	How to determine the length of a dictionary
Add Dictionary Item	How to add an item to a dictionary
Remove Dictionary Items	How to remove dictionary items
Copy Dictionary	How to copy a dictionary
Nested Dictionaries	A dictionary within a dictionary
If Statement	How to write an if statement
If Indentation	If statemnts in Python relies on indentation (whitespace at the beginning of a line)

Elif	elif is the same as "else if" in other programming languages
Else	How to write an if...else statement
Shorthand If	How to write an if statement in one line
Shorthand If Else	How to write an if...else statement in one line
If AND	Use the and keyword to combine if statements
If OR	Use the or keyword to combine if statements
Nested If	How to write an if statement inside an if statement
The pass Keyword in If	Use the pass keyword inside empty if statements
While	How to write a while loop
While Break	How to break a while loop
While Continue	How to stop the current iteration and continue wit the next
While Else	How to use an else statement in a while loop
For	How to write a for loop
Loop Through a String	How to loop through a string

For Break	How to break a for loop
For Continue	How to stop the current iteration and continue wit the next
Looping Through a rangee	How to loop through a range of values
For Else	How to use an else statement in a for loop
Nested Loops	How to write a loop inside a loop
For pass	Use the pass keyword inside empty for loops
Function	How to create a function in Python
Call a Function	How to call a function in Python
Function Arguments	How to use arguments in a function
*args	To deal with an unknown number of arguments in a function, use the * symbol before the parameter name
Keyword Arguments	How to use keyword arguments in a function
*kwargs	To deal with an unknown number of keyword arguments in a function, use the * symbol before the parameter name
Default Parameter Value	How to use a default parameter value

Passing a List as an Argument	How to pass a list as an argument
Function Return Value	How to return a value from a function
The pass Statement i Functions	Use the pass statement in empty functions
Function Recursion	Functions that can call itself is called recursive functions
Lambda Function	How to create anonymous functions in Python
Why Use Lambda Functions	Learn when to use a lambda function or not
Array	Lists can be used as Arrays
What is an Array	Arrays are variables that can hold more than one value
Access Arrays	How to access array items
Array Length	How to get the length of an array
Looping Array Elements	How to loop through array elements
Add Array Element	How to add elements from an array
Remove Array Element	How to remove elements from an array
Array Methods	Python has a set of Array/Lists methods

Class	A class is like an object constructor
Create Class	How to create a class
The Class __init__() Function	The __init__() function is executed when the class is initiated
Object Methods	Methods in objects are functions that belongs to the object
self	The self parameter refers to the current instance of the class
Modify Object Properties	How to modify properties of an object
Delete Object Properties	How to modify properties of an object
Delete Object	How to delete an object
Class pass Statement	Use the pass statement in empty classes
Create Parent Class	How to create a parent class
Create Child Class	How to create a child class
Create the __init__() Function	How to create the __init__() function
super Function	The super() function make the child class inherit the parent class

Add Class Properties	How to add a property to a class
Add Class Methods	How to add a method to a class
Iterators	An iterator is an object that contains a countable number of values
Iterator vs Iterable	What is the difference between an iterator and an iterable
Loop Through an Iterator	How to loop through the elements of an iterator
Create an Iterator	How to create an iterator
StopIteration	How to stop an iterator
Global Scope	When does a variable belong to the global scope?
Global Keyword	The global keyword makes the variable global
Create a Module	How to create a module
Variables in Modules	How to use variables in a module
Renaming a Module	How to rename a module
Built-in Modules	How to import built-in modules
Using the dir() Function	List all variable names and function names in a

	module
Import From Module	How to import only parts from a module
Datetime Module	How to work with dates in Python
Date Output	How to output a date
Create a Date Object	How to create a date object
The strftime Method	How to format a date object into a readable string
Date Format Codes	The datetime module has a set of legal format codes
JSON	How to work with JSON in Python
Parse JSON	How to parse JSON code in Python
Convert into JSON	How to convert a Python object in to JSON
Format JSON	How to format JSON output with indentations and line breaks
Sort JSON	How to sort JSON
RegEx Module	How to import the regex module
RegEx Functions	The re module has a set of functions
Metacharacters in RegEx	Metacharacters are characters with a special meaning

RegEx Special Sequences	A backslash followed by a a character has a special meaning
RegEx Sets	A set is a set of characters inside a pair of square brackets with a special meaning
RegEx Match Object	The Match Object is an object containing information about the search and the result
Install PIP	How to install PIP
PIP Packages	How to download and install a package with PIP
PIP Remove Package	How to remove a package with PIP
Error Handling	How to handle errors in Python
Handle Many Exceptions	How to handle more than one exception
Try Else	How to use the else keyword in a try statement
Try Finally	How to use the finally keyword in a try statement
raise	How to raise an exception in Python

Beginner C Exercises

Question 1

Question:

Write a program to print Hello World!.

Solution:

```
#include<stdio.h>
int main()
{
printf("Hello,world!");
return 0;
}
```

Question 2

Question:

Write a program to find the area of a circle.

Solution:

```
#include<stdio.h>
int main()
{
int r, area;
r = 2;
area = 3.14 * r * r;
printf("The area of the circle = %d", area);
return 0;
}
```

Question 3

Question:

Write a program to find the sum of two numbers.

Solution:

```
#include<stdio.h>
int main()
{
int a, b, sum;
a=1;
b=2;
sum = a + b;
printf("The sum of a and b = %d", sum);
return 0;
}
```

Question 4

Question:

Write a program to find the square of a number.

Solution:

```
#include<stdio.h>
#include<math.h>
int main()
{
int a, b;
a=2;
b = pow((a), 2);
printf("The square of a = %d", b);
return 0;
}
```

Question 5

Question:

Write a program to find the greatest of two numbers.

Solution:

```
#include<stdio.h>
int main()
{
int a, b;
a = 2;
b = 3;
if(a>b)
{
printf("a is greater than b");
}
else
{
printf("b is greater than a");
}
return 0;
}
```

Question 6

Question:

Write a program to print the average of the elements in the array.

Solution:

```
#include<stdio.h>
int main()
{
int i, avg, sum = 0;
int num [5] = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
sum = sum + num [i];
avg = sum/5;
printf("Sum of the Elements in the array = %d", sum);
printf("Average of the elements in the array= %d", avg);
return 0;
}
```

Question 7

Question:

Write a program such that a Switch (case) allows to make a decision from the number of choices, i.e., from the number of cases.

Solution:

```
#include<stdio.h>
int main()
{
char ch;
printf("Enter any character:");
scanf("%c", &ch);
switch(ch)
{
case 'R':
printf("Red");
break;
case 'W':
printf("White");
break;
case 'Y':
printf("Yellow");
break;
case 'G':
printf("Green");
break;
default:
printf("Error");
break;
}
return 0;
}
```

Question 8

Question:

Write a program to find the greatest of two numbers using pointers.

Solution:

```
#include<stdio.h>
int main()
{
int x, y, *p, *q;
printf("Enter any integer:");
scanf("%d", &x);
printf("Enter any integer:");
scanf("%d", &y);
p = &x;
q = &y;
if(*p>*q)
{
printf("x is greater than y");
}
if(*q>*p)
{
printf("y is greater than x");
}
return 0;
}
```

Question 9

Question:

Write a program to print the address of x and the value assigned to x.

Solution:

```
#include <stdio.h>
int main()
{
int x, *p;
x = 1;
p = &x;
printf("The address of the variable x =%d", p);
printf("The value of the variable x =%d", *p);
return 0;
}
```

Question 10

Question:

Write a program to print the first 10 numbers starting from one together with their squares and cubes.

Solution:

```
#include<stdio.h>
int main()
{
int i;
for( i=1; i<=10; i++)
printf("Number=%d its square=%d its cube=%d\n", i , i*i, i*i*i);
return 0;
}
```

Question 11

Question:

Write a program:

If you enter a character M

Output must be: ch = M.

Solution:

```
#include<stdio.h>
int main()
{
char M;
printf("Enter any character:");
scanf("%c", &M);
printf("ch=%c", M);
return 0;
}
```

Question 12

Question:

Write a program to print the multiplication table of a number.

Solution:

```
#include<stdio.h>
int main()
{
int n, i;
printf("Enter any number:");
scanf("%d", &n);
for( i=1; i<=5; i++)
printf("%d * %d = %d\n", n, i, n*i);
return 0;
}
```

Question 13

Question:

Write a program to print the product of the first 10 digits.

Solution:

```
#include<stdio.h>
int main()
{
int i, product = 1;
for( i=1; i<=10; i++)
product = product * i;
printf("The product of the first 10 digits =%d", product);
return 0;
}
```

Question 14

Question:

Write a program to print whether the given number is positive or negative.

Solution:

```
#include<stdio.h>
int main()
{
int a;
a = -35;
if(a>0)
{
printf("Number is positive");
}
else
{
printf("Number entered is negative");
}
return 0;
}
```

Question 15

Question:

Write a program to check the equivalence of two numbers.

Solution:

```
#include<stdio.h>
int main()
{
int x, y;
printf("Enter any number:");
scanf ("%d", &x);
printf("Enter any number:");
scanf ("%d", &y);
if(x-y==0)
{
printf("The two numbers are equivalent");
}
else
{
printf("The two numbers are not equivalent");
}
return 0;
}
```

Question 16

Question:

Write a program to print the remainder of two numbers.

Solution:

```
#include<stdio.h>
int main()
{
```

```
int a, b, c;
printf("Enter any number:");
scanf("%d", &a);
printf("Enter any number:");
scanf("%d", &b);
c = a%b;
printf("The remainder of a and b = %d", c);
return 0;
}
```

Question 17

Question:

Write a program to print the given number is even or odd.

Solution:

```
#include<stdio.h>
int main()
{
int a;
printf("Enter any number:");
scanf ("%d", &a);
if(a%2 = = 0)
{
printf("The number is even");
}
else
{
printf("The number is odd");
}
return 0;
}
```

Question 18

Question:

Write a program to print the characters from A to Z.

Solution:

```
#include<stdio.h>
int main()
{
char a;
for( a='A'; a<='Z'; a++)
printf("%c\n", a);
return 0;
}
```

Question 19

Question:

Write a program to find the incremented and decremented values of two numbers.

Solution:

```
#include<stdio.h>
int main()
```

```
{
int a, b, c, d, e, f;
a = 10;
b=12;
c=a+1;
d=b+1;
e=a-1;
f=b-1;
printf("The incremented value of a =%d", c);
printf("The incremented value of b =%d", d);
printf("The decremented value of a =%d", e);
printf("The decremented value of b =%d", f);
return 0;
}
```

Question 20

Question:

Write a program to calculate the simple interest.

Solution:

```
#include<stdio.h>
int main()
{
int P,T, R, SI;
P = 1000;
T = 2;
R = 3;
SI = P*T*R/100;
printf("The simple interest = %d", SI);
return 0;
}
```

Question 21

Question:

Write a program to Find the largest of three numbers.

Solution:

```
#include<stdio.h>
int main()
{
int a, b, c;
printf("Enter any number:");
scanf("%d", &a);
printf("Enter any number:");
scanf("%d", &b);
printf("Enter any number:");
scanf("%d", &c);
if(a>b&&a>c)
{
printf("%d is greater than %d and %d", a, b, c);
}
else if (b>a&&b>c)
{
printf("%d is greater than %d and %d", b, a, c);
}
else
{
printf("%d is greater than %d and %d", c, b, a);
}
return 0;
}
```

Question 22

Question:

Write a program to print the factorial of the entered number.

Solution:

```
#include<stdio.h>
int main()
{
int i, n, fact=1 ;
printf("Enter any number:");
scanf("%d", &n);
for(i=1; i<=n; i++)
fact = fact *i;
printf("\n Entered number is: %d", n);
printf("\n The factorial of the entered number %d is: %d", n, fact);
return 0;
}
```

Question 23

Question:

Write a program to print the length of the entered string.

Solution:

```
#include<stdio.h>
#include<string.h>
int main()
{
char ch[4];
printf("Enter any word: ");
scanf("%c", &ch);
printf("The length of the string = %d", strlen(ch));
return 0;
}
```

Question 24

Question:

Write a program to print the ASCII value of the entered character.

Solution:

```
#include<stdio.h>
int main()
{
char ch ='A';
printf("The ASCII value of ch is: %d", ch);
return 0;
}
```

Question 25

Question:

Write a program to check whether the entered character is a lower case letter or not.

Solution:

```
#include<stdio.h>
int main()
{
char ch = 'a';
if(islower(ch))
printf("you have entered the lower case letter");
else
printf("you have entered the upper case letter");
return 0;
}
```

Question 26

Question:

Write a program to check whether the entered character is a upper case letter or not.

Solution:

```
#include<stdio.h>
int main()
{
char ch = 'a';
if(isupper(ch))
printf("you have entered the upper case letter");
else
printf("you have entered the lower case letter");
return 0;
}
```

Question 27

Question:

Write a program to convert the lower case letter to upper case letter.

Solution:

```
#include<stdio.h>
int main()
{
char ch = 'a';
char b = toupper(ch);
printf("lower case letter %c is converted to upper case letter %c", ch, b);
return 0;
}
```

Question 28

Question:

Write a program to print the output:

Einstein [0] = E

Einstein [1] = I

Einstein [2] = N

Einstein [3] = S

Einstein [4] = T

Einstein [5] = E

Einstein [6] = I

Einstein [7] = N

Solution:

```
#include<stdio.h>
int main()
{
int i;
char name [8] = {' E' , ' I', ' N', ' S', ' T ', ' E', ' I', ' N'};
for(i=0; i<8; i++)
printf("\n Element [%d] = %c", i, name[i]);
return 0;
}
```

Question 29

Question:

Write a program to print the output:

Name of the book = B

Price of the book = 135.00

Number of pages = 300

Edition = 8

using structures.

Solution:

```
#include<stdio.h>
int main()
{
struct book {
char name;
float price;
int pages;
int edition;
};
struct book b1;
b1.name = 'B';
b1.price = 135.00;
b1.pages = 300;
b1.edition = 8;
printf("\n Name of the book = %c", b1.name);
printf("\n Price of the book = %f", b1.price);
printf("\n Number of pages = %d", b1.pages);
printf("\n Edition of the book = %d", b1.edition);
return 0;
}
```

Question 30

Question:

Write a program to find square of a number using functions.

Solution:

```
#include<stdio.h>
int square();
int main()
```

```
{
int answer;
answer = square();
printf("Square of the given number=%d", answer);
return(0);
}
int square()
{
int x;
printf("Enter any integer:");
scanf("%d", &x);
return x*x;
}
```

Question 31

Question:

Write a program To print "hello world" 10 times.

Solution:

```
#include<stdio.h>
int main()
{
int i;
for (i =1; i<=10; i ++)
printf("hello world \n");
return 0;
}
```

Question 32

Question:

Write a program to print first 5 numbers using do while loop statement.

Solution:

```
#include<stdio.h>
int main()
{
int i =1;
do
{
printf("%d\n", i++);
} while (i<=5);
return 0;
}
```

Question 33

Question:

Write a program to print the output:

body [b] = b

body [o] = o

body [d] = d

body [y] = y

Solution:

```
#include <stdio.h>
int main()
{
char i;
char body [4] = {'b', 'o', 'd', 'y'};
for(i=0; i<4; i++)
printf("\n body[%c] = %c", body[i] , body[i]);
return 0;
}
```

Question 34

Question:

What will be the output of the below program:

```
#include<stdio.h>
#include<stdlib.h>
int main () {
printf("linux\n");
exit (0);
printf("php\n");
return 0;
}
```

Solution:

```
linux
```

Question 35

Question:

Write a program to check whether a character is an alphabet or not.

Solution:

```
#include <stdio.h>
#include <ctype.h>
int main()
{
int a =2;
if(isalpha(a))
{
printf("The character a is an alphabet");
}
else
{
printf("The character a is not an alphabet");
}
return 0;
}
```

Question 36

Question:

Write a program to calculate the discounted price and the total price after discount

Given:

If purchase value is greater than 1000, 10% discount

If purchase value is greater than 5000, 20% discount

If purchase value is greater than 10000, 30% discount.

Solution:

```
#include<stdio.h>
int main()
{
double PV;
printf("Enter purchased value:");
scanf("%lf", &PV);
if(PV>1000)
{
printf("\n Discount=%lf", PV* 0.1);
printf("\n Total=%lf", PV - PV* 0.1);
}
else if(PV>5000)
{
printf("\n Discount =%lf", PV* 0.2);
printf("\n Total=%lf", PV - PV* 0.1);
}
else
{
printf("\n Discount=%lf", PV* 0.3);
printf("\n Total=%lf", PV - PV* 0.1);
}
return 0;
}
```

Question 37

Question:

Write a program to print the first ten natural numbers using while loop statement.

Solution:

```
#include<stdio.h>
int main()
{
int i = 1;
while (i<=10)
{
printf("%d\n", i++);
}
return 0;
}
```

Question 38

Question:

What will be the output of the below program:

```
#include <stdio.h>
int main()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
continue;
}
printf("%d\n ", i);
}
return 0;
}
```

Solution:

```
1
2
4
5
```

Question 39

Question:

Write a program to find the size of an array.

Solution:

```
#include <stdio.h>
int main()
{
   int num [] = {11, 22, 33, 44, 55, 66};
    int n;
    /* Calculating the size of the array with this formula.
     * n = sizeof(array_name) / sizeof(array_name[0])
     * This is a universal formula to find number of elements in
     * an array, which means it will work for arrays of all data
     * types such as int, char, float etc.
     */
    n = sizeof(num) / sizeof(num [0]);
    printf("Size of the array is: %d\n", n);
    return 0;
}
```

Question 40

Question:

What would be the output of the following programs:

```
#include <stdio.h>
int main()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
break;
}
printf("%d\n", i);
}
return 0;
}
```

Solution:

```
1
2
```

```
#include <stdio.h>
int main()
{
int i;
for(i=1;i<=5;i++)
{
if(i==3)
{
goto HAI;
}
printf("\n %d ",i);
}
HAI : printf("\n Linux");
}
```

Solution:

```
1
2
Linux
```

```
#include<stdio.h>
int main()
{
int i = 54;
int y = i<<1;
printf("The value of y = %d", y);
return 0;
}
```

Solution:

```
The value of y = 108
```

```
#include<stdio.h>
int main()
{
int i = 54;
int y = i>>1;
printf("The value of y = %d", y);
return 0;
}
```

Solution:

```
The value of y = 27
```

```
#include<stdio.h>
#include <stdlib.h>
int main()
{
int a, b;
a= - 2;
b= abs(a);
printf("Absolute value = %d", b);
return 0;
}
```

Solution:

```
Absolute value = 2
```

```
#include <stdio.h>
int main()
{
for( ; ; )
{
printf("This loop will run forever.\n");
}
return 0;
}
```

Solution:

```
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever. .........
```

```
#include<stdio.h>
int main()
{
printf("Hello,world!");
return 0;
printf("Hello,world!");
}
```

Solution:

```
Hello,world!
```

Question 41

Question:

Write a program to check whether the person is a senior citizen or not.

Solution:

```
#include<stdio.h>
int main()
{
int age;
printf("Enter age:");
scanf("%d", &age);
if(age>=60)
{
printf("senior citizen");
}
else
{
printf("not a senior citizen");
}
return 0;
}
```

Question 1

Question:

Write a program to print Hello World!.

Solution:

```
#include<iostream>
int main()
{
std::cout<<"Hello World!";
return 0;
}
```

Solution:

```
#include<iostream>
using namespace std;
int main()
{
cout<<"Hello World!";
return 0;
}
```

Question 2

Question:

Write a program to find the area of a circle.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
float r, area;
cout<<"Enter any number:";
cin>>r;
area = 3.14 * r * r;
cout<<"The area of the circle = "<< area;
return 0;
}
```

Question 3

Question:

Write a program to find the sum of two numbers.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
float a, b, sum;
cout<<"Enter any two numbers:";
cin>>a;
cin>>b;
sum = a+ b;
cout<<"The sum of a and b = "<< sum;
return 0;
}
```

Question 4

Question:

Write a program to find the square of a number.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int a, b;
a=2;
b = a * a;
cout<<"The square of a = "<< b;
return 0;
}
```

Question 5

Question:

Write a program to find the greatest of two numbers.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int a, b;
a = 2;
b = 3;
if(a>b)
{
cout<<"a is greater than b";
}
else
{
cout<<"b is greater than a";
}
return 0;
}
```

Question 6

Question:

Write a program to print the average of the elements in the array.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int i, avg, sum = 0;
int num [5] = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
sum = sum + num [i];
avg = sum/5;
cout<<"Sum of the Elements in the array = "<< sum <<endl;
cout<<"Average of the elements in the array= "<< avg<<endl;
return 0;
}
```

Question 7

Question:

Write a program such that a Switch (case) allows to make a decision from the number of choices, i.e., from the number of cases.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
char ch;
cout<<"Enter any character:";
cin>>ch;
switch(ch)
{
case 'R':
cout<<"Red";
break;
case 'W':
cout<<"White";
break;
case 'Y':
cout<<"Yellow";
break;
case 'G':
cout<<"Green";
break;
default:
cout<<"Error";
break;
}
return 0;
}
```

Question 8

Question:

Write a program to find the greatest of two numbers using pointers.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int x, y, *p, *q;
cout<<"Enter any integer:";
cin>> x;
cout<<"Enter any integer:";
cin>> y;
p = &x;
q = &y;
if(*p>*q)
{
cout<<"x is greater than y";
}
else
{
cout<<"y is greater than x";
}
return 0;
}
```

Question 9

Question:

Write a program to print the address of x and the value assigned to x.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int x, *p;
cout<<"Enter any integer:";
cin>>x;
p = &x;
cout<<"The address of the variable x = "<< p<<endl;
cout<<"The value of the variable x = "<< *p<<endl;
return 0;
}
```

Question 10

Question:

Write a program to print the first 10 numbers starting from one together with their squares and cubes.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int i;
for( i=1; i<=10; i++)
cout<<"number = "<< i <<" its square = "<< i*i <<" its cube = "<< i*i*i<< endl;
return 0;
}
```

Question 11

Question:

Write a program:

If you enter a character M

Output must be: ch = M.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
char M;
cout<<"Enter any character:";
cin>>M;
cout<<"ch= "<< M;
return 0;
}
```

Question 12

Question:

Write a program to print the multiplication table of a number.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int n, i;
cout<<"Enter any number:";
cin>>n;
for( i=1; i<=5; i++)
cout<< n <<" * "<< i <<" = "<< n*i <<endl;
return 0;
}
```

Question 13

Question:

Write a program to print the product of the first 10 digits.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int i, product = 1;
for( i=1; i<=10; i++)
product = product * i;
cout<<"The product of the first 10 digits = " << product;
return 0;
}
```

Question 14

Question:

Write a program to print whether the given number is positive or negative.

Solution:

```
#include<iostream>
using namespace std;
```

```
int main()
{
int a;
a = -35;
if(a>0)
{
cout<<"Number is positive";
}
else
{
cout<<"Number entered is negative";
}
return 0;
}
```

Question 15

Question:

Write a program to check the equivalence of two numbers.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int x, y;
cout<<"Enter any number:";
cin>>x;
cout<<"Enter any number:";
cin>>y;
if(x-y==0)
{
```

```
cout<<"The two numbers are equivalent";
}
else
{
cout<<"The two numbers are not equivalent";
}
return 0;
}
```

Question 16

Question:

Write a program to print the remainder of two numbers.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int a, b, c;
cout<<"Enter any number:";
cin>>a;
cout<<"Enter any number:";
cin>>b;
c = a % b;
cout<<"The remainder of a and b = "<< c;
return 0;
}
```

Question 17

Question:

Write a program to print the given number is even or odd.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int a;
cout<<"Enter any number:";
cin>>a;
if(a%2 = = 0)
{
cout<<"The number is even";
}
else
{
cout<<"The number is odd";
}
return 0;
}
```

Question 18

Question:

Write a program to print the characters from A to Z.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
char a = 'A';
while (a<='Z')
{
cout<<" \n"<< a++;
}
return 0;
}
```

Question 19

Question:

Write a program to find the incremented and decremented values of two numbers.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int a, b, c, d, e, f;
a = 10;
b=12;
c=a+1;
d=b+1;
e=a-1;
f=b-1;
cout<<"The incremented value of a = "<< c << endl;
cout<<"The incremented value of b = "<< d << endl;
cout<<"The decremented value of a = "<< e << endl;
cout<<"The decremented value of b = "<< f << endl;
return 0;
}
```

Question 20

Question:

Write a program to calculate the simple interest.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int P,T, R, SI;
cout<<"Enter principal amount:";
cin>>P;
cout<<"Enter time:";
cin>>T;
cout<<"Enter rate of interest:";
cin>>R;
SI = P*T*R/100;
cout<<"the simple interest = "<<SI;
return 0;
}
```

Question 21

Question:

Write a program to Find the largest of three numbers.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int a, b, c;
cout<<"Enter any number:";
cin>>a;
cout<<"Enter any number:";
cin>>b;
cout<<"Enter any number:";
cin>>c;
if(a>b&&a>c)
{
cout<< a<<" is greater than "<< b<<" and "<<c;
}
else if (b>a&&b>c)
{
cout<< b<<" is greater than "<< a <<" and "<<c;
}
else
{
cout<< c<<" is greater than "<< b<<" and "<< a;
}
return 0;
}
```

Question 22

Question:

Write a program to print the factorial of the entered number.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int i, n, fact=1 ;
cout<<"Enter any number:";
cin>>n;
for(i=1; i<=n; i++)
fact = fact *i;
cout<<"\n Entered number is: "<< n;
cout<<"\n The factorial of the entered number "<< n <<" is: "<< fact;
return 0;
}
```

Question 23

Question:

Write a program to print the length of the entered string.

Solution:

```
#include<iostream>
#include<string.h>
using namespace std;
int main()
{
char ch[4];
cout<<"Enter any word: ";
cin>>ch;
cout<<"The length of the string = "<< strlen(ch);
return 0;
}
```

Question 24

Question:

Write a program to print the ASCII value of the entered character.

Solution:

```
#include <iostream>
using namespace std;
int main()
{
 char c;
 cout << "Enter a character: ";
 cin >> c;
 cout << "ASCII Value of " << c << " is " << int(c);
 return 0;
}
```

Question 25

Question:

Write a program to check whether the entered character is a lower case letter or not.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
char ch = 'a';
if(islower(ch))
cout<<"you have entered the lower case letter";
else
cout<<"you have entered the upper case letter";
return 0;
}
```

Question 26

Question:

Write a program to check whether the entered character is a upper case letter or not.

Solution:

```
#include<iostream>
```

```
using namespace std;
int main()
{
char ch = 'a';
if(isupper(ch))
cout<<"you have entered the upper case letter";
else
cout<<"you have entered the lower case letter";
return 0;
}
```

Question 27

Question:

Write a program to convert the lower case letter to upper case letter.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
char ch = 'a';
char b = toupper(ch);
cout<<" lower case letter "<<ch<<" is converted to upper case letter "<<b;
return 0;
}
```

Question 28

Question:

Write a program to print the output:

Einstein [0] = E

Einstein [1] = I

Einstein [2] = N

Einstein [3] = S

Einstein [4] = T

Einstein [5] = E

Einstein [6] = I

Einstein [7] = N

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int i;
char name [8] = {'E' , 'I', 'N', 'S', 'T', 'E', 'I', 'N'};
for(i=0; i<8; i++)
cout<<"Element ["<< i <<" ] = "<< name[i] << endl;
return 0;
}
```

Question 29

Question:

Write a program to print the output:

Name of the book = B

Price of the book = 135.00

Number of pages = 300

Edition = 8

using structures.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
struct book {
char name;
float price;
int pages;
int edition;
};
struct book b1= {'B', 135.00, 300, 8};
cout<<"Name of the book = "<< b1.name<< endl;
cout<<"Price of the book = "<< b1.price<<endl;
cout<<"Number of pages = "<< b1.pages<<endl;
cout<<"Edition of the book = "<< b1.edition<< endl;
return 0;
}
```

Question 30

Question:

Write a program to find square of a number using functions.

Solution:

```
#include<iostream>
using namespace std;
int square();
int main()
{
int answer;
answer = square();
cout<<"Square of the given number = "<< answer;
return 0;
}
int square()
{
int x;
cout<<"Enter any integer:";
cin>>x;
return x*x;
}
```

Question 31

Question:

Write a program To print "hello world" 10 times.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int i;
for (i =1; i<=10; i ++)
cout<<"\n hello world";
return 0;
}
```

Question 32

Question:

Write a program to print first 5 numbers using do while loop statement.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int i =1;
do
{
cout<<" \n i= "<< i++;
} while (i<=5);
return 0;
}
```

Question 33

Question:

Write a program to print the output:

body [b] = b

body [o] = o

body [d] = d

body [y] = y

Solution:

```
#include <iostream>
using namespace std;
int main()
{
char i;
char body [4] = {'b', 'o', 'd', 'y'};
for(i=0; i<4; i++)
cout<<"\n body ["<<body[i] <<" ] = "<< body[i] << endl;
return 0;
}
```

Question 34

Question:

What will be the output of the below program:

```
#include <iostream>
using namespace std;
int main()
{
cout<<"linux\n";
exit (0);
cout<<"php\n";
return 0;
}
```

Solution:

```
linux
```

Question 35

Question:

Write a program to check whether a character is an alphabet or not.

Solution:

```
#include <iostream>
using namespace std;
int main()
{
int a =2;
if(isalpha(a))
{
   cout<<"The character a is an alphabet";
}
else
{
cout<<"The character a is not an alphabet";
}
return 0;
}
```

Question 36

Question:

Write a program to calculate the discounted price and the total price after discount

Given:

If purchase value is greater than 1000, 10% discount

If purchase value is greater than 5000, 20% discount

If purchase value is greater than 10000, 30% discount.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
double PV;
cout<<"Enter purchased value:";
cin>>PV;
if(PV>1000)
{
cout<<"Discount = "<< PV* 0.1 << endl;
cout<<"Total= "<< PV - PV* 0.1 << endl;
}
else if(PV>5000)
{
cout<<"Discount = "<< PV* 0.2 << endl;
cout<<"Total= "<< PV - PV* 0.1 << endl;
}
else
{
cout<<"Discount = "<< PV* 0.3 << endl;
cout<<"Total= "<< PV - PV* 0.1 << endl;
}
return 0;
}
```

Question 37

Question:

Write a program to print the first ten natural numbers using while loop statement.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int i = 1;
while (i<=10)
{
cout<<"\n "<< i++;
}
return 0;
}
```

Question 38

Question:

What will be the output of the below program:

```
#include <iostream>
using namespace std;
int main()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
continue;
}
cout<<"\n "<< i;
}
return 0;
}
```

Solution:

```
1
2
4
5
```

Question 39

Question:

Write a program to find the size of an array.

Solution:

```
#include <iostream>
using namespace std;
int main()
{
   int num [] = {11, 22, 33, 44, 55, 66};
    int n;

    /* Calculating the size of the array with this formula.
     * n = sizeof(array_name) / sizeof(array_name[0])
     * This is a universal formula to find number of elements in
     * an array, which means it will work for arrays of all data
     * types such as int, char, float etc.
     */
    n = sizeof(num) / sizeof(num [0]);
    cout<<"Size of the array is:"<<n;
    return 0;
}
```

Question 40

Question:

What would be the output of the following programs:

```
#include <iostream>
using namespace std;
int main()
{
int i;
for (i=1; i<=5; i++)
{
if (i==3)
{
break;
}
cout<<"\n "<< i;
}
return 0;
}
```

Solution:

```
1
2
```

```
#include<iostream>
using namespace std;
int main()
{
int i;
for(i=1;i<=5;i++)
{
if(i==3)
{
goto HAI;
}
cout<<"\n "<< i;
}
HAI : cout<<"\n Linux";
}
```

Solution:

```
1
2
Linux
```

```
#include<iostream>
using namespace std;
int main()
{
int i = 54;
int y = i<<1;
cout<<"The value of y = "<< y;
return 0;
}
```

Solution:

```
The value of y = 108
```

```
#include<iostream>
using namespace std;
int main()
{
int i = 54;
int y = i>>1;
cout<<"The value of y = "<< y;
return 0;
}
```

Solution:

```
The value of y = 27
```

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
int a, b;
a= - 2;
b= abs(a);
cout<<" Absolute value = "<< b<< endl;
return 0;
}
```

Solution:

```
Absolute value = 2
```

```
#include <iostream>
using namespace std;
int main()
{
for( ; ; ) {
cout<<"This loop will run forever.\n";
}
return 0;
}
```

Solution:

```
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever. .........
```

```
#include<iostream>
using namespace std;
int main()
{
cout<<"Hello World!";
return 0;
cout<<"Hello World!";
}
```

Solution:

```
Hello,world!
```

Question 41

Question:

Write a program to check whether the person is a senior citizen or not.

Solution:

```
#include<iostream>
using namespace std;
int main()
{
int age;
age=20;
if(age > = 60)
{
cout<<"Senior citizen";
}
if(age<60)
{
cout<<"Not a senior citizen";
}
return 0;
}
```

Question 42

Question:

Write a program to compute inverse of tan x.

Solution:

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
int x = 20;
cout<<"Inverse of tan x = "<< atan(x);
return 0;
}
```

JavaScript Exercises for Beginners

Question 1

Question:

Write a program to print Hello World!.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
document.write("Hello World!");
</script>
</body>
</html>
```

Question 2

Question:

Write a program to add two numbers.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<p>A typical addition operation adds two numbers and produces a new number.</p>
<script>
var x ;
var y;
var z;
x =100;
y = 200;
z = x+ y;
document.write(" The sum of two numbers is:     " + z);
</script>
</body>
</html>
```

Question 3

Question:

Write a program to subtract two numbers.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<p>A typical subtraction operation subtracts two numbers and produces a new number.</p>
<script>
var x ;
var y;
var z;
x=300;
y = 200;
z = x-y;
document.write(" The difference of two numbers is:     " + z);
</script>
</body>
</html>
```

Question 4

Question:

Write a program to divide two numbers.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<p>A typical division operation divides two numbers and produces a new number.</p>
<script>
var x ;
var y;
var z;
x=300;
y = 200;
z = x/y;
document.write(" The division of two numbers is:     " + z);
</script>
</body>
</html>
```

Question 5

Question:

Write a program to multiply two numbers.

Solution:

```
<!DOCTYPE html>
```

```
<html>
<body>
<p> A typical multiplication operation multiplies two numbers and produces a new number.</p>
<script>
var x ;
var y;
var z;
x=300;
y = 200;
z = x* y;
document.write(" The multiplication of two numbers is:     " + z);
</script>
</body>
</html>
```

Question 6

Question:

Write a program to find the area of a circle.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
var r ;
var area;
r=3;
```

```
area = 3.14* r* r;
document.write(" The area of the circle is:     " + area +" centimeter square");
</script>
</body>
</html>
```

Question 7

Question:

Write a program to find the square root of a number.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var z;
x=4;
z = Math.sqrt(x);
document.write(" The square root of a number z is: " + z);
</script>
</body>
</html>
```

Question 8

Question:

Write a program to find the cube root of a number.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var z;
x=4;
z = Math.cbrt(x);
document.write(" The cube root of a number z is: " + z);
</script>
</body>
</html>
```

Question 9

Question:

Write a program to round off a number.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var z;
x=4.5;
z = Math.round(x);
document.write(" The round off a number z is: " + z);
</script>
</body>
</html>
```

Question 10

Question:

Write a program to find the incremented and decremented values of two numbers.

Solution:

```
<!DOCTYPE html>
```

```
<html>
<body>
<script>
var x ;
var y;
var z;
var p;
var a;
var b;
x=4;
y=6;
z=x+1;
p=x-1;
a = y+1;
b= y-1;
document.write(" The incremented value of  x  is: " + z);
document.write(" The decremented value of  x  is: " + p);
document.write(" The incremented value of  y  is: " + a);
document.write(" The decremented value of  y  is: " + b);
</script>
</body>
</html>
```

Question 11

Question:

Write a program to find the greatest of two numbers using if – else statement.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
var x ;
var y;
x=4;
y=6;
if(x>y){
document.write(" x is greater than y");
} else {
document.write(" y is greater than x");
}
</script>
</body>
</html>
```

Question 12

Question:

Write a program to print the first ten natural numbers using for loop statement.

Solution:

```
<!DOCTYPE html>
<html>
<body>
```

```
<script>
var i ;
for (i=1; i<=10; i++)
document.write("" +  i);
</script>
</body>
</html>
```

Question 13

Question:

Write a program to print the first ten natural numbers using while loop statement.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
var i=1 ;
while (i<=10)
document.write("" +  i++);
</script>
</body>
</html>
```

Question 14

Question:

Write a program to print the first nine natural numbers using do while loop statement.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
var i=1 ;
do{
document.write("" +  i++);
} while (i<10)
</script>
</body>
</html>
```

Question 15

Question:

Write a program to print the average of the first 10 numbers using for loop statement.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
var i, avg, sum = 0;
for( i=1; i<=10; i++)
sum = sum + i;
avg = sum/10;
document.write("<br> sum of the first 10 numbers =  </br>" + sum);
document.write("<br> average of the first10 numbers =  </br>" + avg);
</script>
</body>
</html>
```

Question 16

Question:

Write a program to add two numbers using JavaScript function.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
function addition(a, b) {
return a + b;
}
document.write("" + addition(4, 3));
</script>
</body>
</html>
```

Question 17

Question:

Write a program to display the current date and time.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<button type="button"
onclick="document.getElementById('demo').innerHTML = Date()">
Click me to display Date and Time.</button>
<p id="demo"></p>
</body>
```

```
</html>
```

Question 18

Question:

Write a program to find the area of a triangle where lengths of the three of its sides are 5, 6, 7.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
var side1 = 5;
var side2 = 6;
var side3 = 7;
var s = (side1 + side2 + side3)/2;
var area =  Math.sqrt(s*((s-side1)*(s-side2)*(s-side3)));
document.write(area);
</script>
</body>
</html>
```

Question 19

Question:

Write a program to convert temperature from fahrenheit to celsius.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
function toCelsius(f) {
  return (5/9) * (f-32);
}
document.getElementById("demo").innerHTML = toCelsius(77);
</script>
</body>
</html>
```

Question 20

Question:

Write a program to get the website URL (loading page).

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
alert(document.URL);
</script>
</body>
</html>
```

Question 21

Question:

Write a program to reverse a given string.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
function string_reverse(str)
{
    return str.split("").reverse().join("");
}
document.write(string_reverse("JavaScript"));
</script>
</body>
</html>
```

Question 22

Question:

Write a program to count the number of vowels in a given string.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
function vowel_Count(str)
{

  return str.replace(/[^aeiou]/g, "").length;
}
document.write(vowel_Count("Python"));
</script>
</body>
</html>
```

Question 23

Question:

Write a program to find sum of array elements.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
var num = [1,2,3,4]
var sum = 0;
for(var i = 0; i < num.length; i++){
  sum += num[i]
}
document.write(sum);
</script>
</body>
</html>
```

Question 24

Question:

Write a program to create the dot products of two given 3D vectors.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
function dot_product(vector1, vector2) {
  var result = 0;
  for (var i = 0; i < 3; i++) {
    result += vector1[i] * vector2[i];
  }
  return result;
}
document.write(dot_product([1,2,3], [1,2,3]))
</script>
</body>
</html>
```

Question 25

Question:

Write a program to find the number of even digits in a given integer.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
function even_digits(num) {
  var ctr = 0;
  while (num) {
    ctr += num % 2 === 0;
    num = Math.floor(num / 10);
  }
  return ctr;
}
document.write(even_digits(124));
</script>
</body>
</html>
```

Question 26

Question:

Write a program to change the capitalization of all letters in a given string.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
function change_case(txt) {
    var str1 = "";
    for (var i = 0; i < txt.length; i++) {
        if (/[A-Z]/.test(txt[i])) str1 += txt[i].toLowerCase();
        else str1 += txt[i].toUpperCase();
    }
    return str1;
}
document.write(change_case("germany"));
</script>
</body>
</html>
```

Question 27

Question:

Write a program to remove all characters from a given string that appear more than once.

Solution:

```
<!DOCTYPE html>
<html>
<body>
```

```
<script>
function remove_duplicate_cchars(str) {
  var arr_char = str.split("");
  var result_arr = [];

  for (var i = 0; i < arr_char.length; i++) {
    if (str.indexOf(arr_char[i]) === str.lastIndexOf(arr_char[i]))
      result_arr.push(arr_char[i]);
    }

  return result_arr.join("");
}
document.write(remove_duplicate_cchars("abcdabc"));
</script>
</body>
</html>
```

Question 28

Question:

Write a program to sort an array of all prime numbers between 1 and a given integer.

Solution:

```
<!DOCTYPE html>
<html>
<body>
```

```
<script>
function sort_prime(num) {

  var prime_num1 = [],
      prime_num2 = [];
  for (var i = 0; i <= num; i++) {
    prime_num2.push(true);
  }
  for (var i = 2; i <= num; i++) {
    if (prime_num2[i]) {
      prime_num1.push(i);
      for (var j = 1; i * j <= num; j++) {
        prime_num2[i * j] = false;
      }
    }
  }

  return prime_num1;
}
document.write(sort_prime(11));
</script>
</body>
</html>
```

Question 29

Question:

Write a program to check whether there is at least one element which occurs in two given sorted arrays of integers.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
function check_common_element(arra1, arra2) {
  for (var i = 0; i < arra1.length; i++)
  {
    if (arra2.indexOf(arra1[i]) != -1)
      return true;
  }
  return false;
}
document.write(check_common_element([1,2,3], [3,4,5]));
</script>
</body>
</html>
```

Question 30

Question:

Write a program to find the maximum difference between any two adjacent elements of a given array of integers.

Solution:

```
<!DOCTYPE html>
<html>
<body>
<script>
function max_difference(arr) {
         var max = -1;
    var temp;
         for (var i = 0; i < arr.length - 1; i++)
      {
                  temp = Math.abs(arr[i] - arr[i + 1]);
                  max = Math.max(max, temp);
           }
         return max;
}

document.write(max_difference([1, 2, 3, 8, 9]));
</script>
</body>
</html>
```

PHP Exercises for Beginners

Question 1

Question:

Write a program to print Hello World!.

Solution:

```
<?php
echo "Hello World!";
?>
```

Question 2

Question:

Write a program to add two numbers.

Solution:

```
<?php
$num1 =1;
$num2=5;
$sum = $num1 + $num2;
echo "Sum of the two numbers is : $sum";
?>
```

Question 3

Question:

Write a program to subtract two numbers.

Solution:

```php
<?php
$num1 =5;
$num2=1;
$sub = $num1 - $num2;
echo "difference of the two numbers is : $sub";
?>
```

Question 4

Question:

Write a program to divide two numbers.

Solution:

```php
<?php
```

```
$num1 =6;
$num2=2;
$div = $num1 / $num2;
echo "the division of two numbers is : $div";
?>
```

Question 5

Question:

Write a program to multiply two numbers.

Solution:

```
<?php
$num1 = 6;
$num2 = 2;
$mult = $num1 * $num2;
echo "the product of two numbers is : $mult";
?>
```

Question 6

Question:

Write a program to find the area of a circle.

Solution:

```
<?php
$radius = 2.0;
$pi = 3.14159;
$area = $pi * $radius * $radius;
echo("\n radius = $radius centimeter");
echo("\n area = $area centimeter square");
?>
```

Question 7

Question:

Write a program to find the square root of a number.

Solution:

```
<?php
```

```
$num1 = 4.0;
$num2 = sqrt($num1);
echo("The square root of a number = $num2");
?>
```

Question 8

Question:

Write a program to find the cube root of a number.

Solution:

```
<?php
$num1 = 6.0;
$num2 = pow(($num1), 1/3);
echo("\n the cube root of a number = $num2");
?>
```

Question 9

Question:

Write a program to round off a number.

Solution:

```
<?php
$num1 = 4.5;
$num2 = round ($num1);
echo("\n the round off of a number = $num2");
?>
```

Question 10

Question:

Write a program to find the incremented and decremented values of two numbers.

Solution:

```
<?php
$num1 =2;
$num2=3;
$num3 = $num1 +1;
```

```
$num4 = $num1 - 1;
$num5 = $num2 +1;
$num6 = $num2 - 1;
echo ("\n The incremented value of  $num1 = $num3 ");
echo ("\n The decremented value of  $num1 = $num4 ");
echo ("\n The incremented value of  $num2 = $num5 ");
echo ("\n The decremented value of  $num2 = $num6 ");
?>
```

Question 11

Question:

Write a program to find the greatest of two numbers using if – else statement.

Solution:

```
<?php
$x = 4.5;
$y=5;
if($x>$y){
echo (" x is greater than y");
} else {
echo (" y is greater than x");
}
?>
```

Question 12

Question:

Write a program to print the first ten natural numbers using for loop statement.

Solution:

```
<?php
for ($i=1; $i<=10; $i++)
echo (" \n $i");
?>
```

Question 13

Question:

Write a program to print the first ten natural numbers using while loop statement.

Solution:

```php
<?php
$i = 1;
while($i <= 10) {
echo "\n  $i ";
$i++;
}
?>
```

Question 14

Question:

Write a program to print the first nine natural numbers using do while loop statement.

Solution:

```php
<?php
$i = 1;
do {
echo "\n $i ";
$i++;
} while($i <= 9);
?>
```

Question 15

Question:

Write a program to print the average of the first 10 numbers using for loop statement.

Solution:

```php
<?php
$i;
$avg;
$sum = 0;
for( $i=1; $i<=10; $i++)
$sum = $sum + $i;
$avg = $sum/10;
echo "\n sum of the first 10 numbers = $sum ";
echo"\n average of the first 10 numbers = $avg  ";
?>
```

Question 16

Question:

Write a program to add two numbers using PHP function.

Solution:

```php
<?php
function addition($a, $b) {
return $a + $b;
}
$sum = addition(4, 3);
echo "the sum of two numbers = $sum ";
?>
```

Question 17

Question:

Write a program to display the current date and time.

Solution:

```php
<?php
    $currentDateTime = date('Y-m-d H:i:s');
    echo $currentDateTime;
```

```
?>
```

Question 18

Question:

Write a program to calculate area of a triangle with base as 20 and height as 25.

Solution:

```
<?php
 $base = 10;
 $height = 15;
 echo "area  = " . ($base * $height) / 2;
 ?>
```

Question 19

Question:

Write a program to convert temperature from fahrenheit to celsius.

Solution:

```
<?php
 $far = 89;
 $cel=($far - 32) * (5/9);
 echo "Temperature in Fahrenheit is : $far". "<br />";
 echo "Temperature in Celcius is : $cel" ;
  ?>
```

Question 20

Question:

Write a program to print factorial of a number..

Solution:

```
<?php
$num = 5;
$fact = 1;
for ($x=$num; $x>=1; $x--)
{
  $fact = $fact * $x;
}
```

```
echo "$fact";
?>
```

Question 21

Question:

Write a program to reverse a given string.

Solution:

```
<?php
$str = "JavaScript";
$len = strlen($str);
for ($i=($len-1) ; $i >= 0 ; $i--)
{
  echo $str[$i];
}
?>
```

Question 22

Question:

Write a program to count the number of vowels in a given string.

Solution:

```
<?php

function vowel_Count($string)
{
    preg_match_all('/[aeiou]/i', $string, $matches);
    return count($matches[0]);
}
print_r(vowel_Count('Python'));



?>
```

Question 23

Question:

Write a program to find sum of array elements.

Solution:

```
<?php
$a=array(1,2,3,4);
echo array_sum($a);
?>
```

Question 24

Question:

Write a program to check whether a number is Even or Odd.

Solution:

```
<?php
$num=253;
if($num%2==0)
{
 echo "$num is Even Number";
}
else
{
 echo "$num is Odd Number";
}
?>
```

Question 25

Question:

Write a program to list the first 15 prime numbers.

Solution:

```
<?php
$count = 0;
$num = 2;
while ($count < 15 )
{
$div=0;
for ( $i=1; $i<=$num; $i++)
{
if (($num%$i)==0)
{
$div++;
}
}
if ($div<3)
{
echo $num." </br> ";
$count=$count+1;
}
$num=$num+1;
}
?>
```

Question 26

Question:

Write a program to reverse a given number.

Solution:

```
<?php
$num = 253;
$rev = 0;
while ($num > 1)
{
$rem = $num % 10;
$rev = ($rev * 10) + $rem;
$num = ($num / 10);
}
echo "$rev";
?>
```

Question 27

Question:

Write a program to check whether 507 is Armstrong or not.

Solution:

```php
<?php
$num=507;
$total=0;
$x=$num;
while($x!=0)
{
$rem=$x%10;
$total=$total+$rem*$rem*$rem;
$x=$x/10;
}
if($num==$total)
{
echo "Armstrong number";
}
else
{
echo "No it is not an armstrong number";
}
?>
```

Question 28

Question:

Write a program to print table of a number.

Solution:

```
<?php
define('a', 7);
for($i=1; $i<=10; $i++)
{
  echo "7 * $i = ", $i*a;
  echo '<br>';
}
?>
```

Question 29

Question:

Write a program to print the first 12 numbers of a Fibonacci series.

Solution:

```
<?php
$num = 0;
$n1 = 0;
$n2 = 1;
echo $n1.' '.$n2.' ';
while ($num < 10 )
{
    $n3 = $n2 + $n1;
    echo $n3.' ';
    $n1 = $n2;
    $n2 = $n3;
    $num = $num + 1;
    }
?>
```

Question 30

Question:

Write a program to swap two numbers 65 and 98 using a third variable.

Solution:

```
<?php
$a = 65;
$b = 98;
$c = $a;
$a = $b;
$b = $c;
echo "After swapping:<br><br>";
echo "a =".$a."  b=".$b;
?>
```

Beginner Python Exercises

Question 1

Question:

Write a program to Add Two Numbers.

Solution:

```
a = 1
b = 2
c= a+b
print(c)
```

```
a = int(input("enter a number: "))
b = int(input("enter a number: "))
c= a+b
print(c)
```

Question 2

Question:

Write a program to find whether a given number (accept from the user) is even or odd, print out an appropriate message to the user.

Solution:

```
a = int(input("enter a number: "))
if a % 2 == 0:
    print("This is an even number.")
else:
    print("This is an odd number.")
```

Question 3

Question:

Write a program to check whether a number entered by the user is positive, negative or zero.

Solution:

```
a = int(input("Enter a number: "))
if a > 0:
   print("Positive number")
elif a == 0:
   print("Zero")
else:
   print("Negative number")
```

Question 4

Question:

Write a program to display the calendar of a given date.

Solution:

```
import calendar
yy = int(input("Enter year: "))
mm = int(input("Enter month: "))
print(calendar.month(yy, mm))
```

Question 5

Question:

Write a program to ask the user to enter the string and print that string as output of the program.

Solution:

```
string = input("Enter string: ")
print("You entered:",string)
```

Question 6

Question:

Write a program to Concatenate Two Strings.

Solution:

```
string1 = input("Enter first string to concatenate: ")
string2 = input("Enter second string to concatenate: ")
string3 = string1 + string2
print("String after concatenation = ",string3)
```

Question 7

Question:

Write a program to Check if an item exists in the list.

Solution:

```
list_of_items = ["ball", "book", "pencil"]
item = input("Type item to check: ")
if item in list_of_items:
 print("Item exists in the list.")
else:
  print("Item does not exist in the list.")
```

Question 8

Question:

Write a program to Join two or more lists.

Solution:

```
list1 = ["This" , "is", "a", "sample", "program"]
list2 = [10, 2, 45, 3, 5, 7, 8, 10]
finalList = list1 + list2
print(finalList)
```

Question 9

Question:

Write a program to Calculate Cube of a Number.

Solution:

```
import math
a = int(input("Enter a number: "))
b=math.pow(a,3)
print (b)
```

Question 10

Question:

Write a program to Calculate Square root of a Number.

Solution:

```
import math
a = int(input("Enter a number: "))
b=math.sqrt(a)
print (b)
```

Question 11

Question:

Write a program that takes a list of numbers (for example, a = [5, 10, 15, 20, 25]) and makes a new list of only the first and last elements of the given list.

Solution:

```
a = [5, 10, 15, 20, 25]
print([a[0], a[4]])
```

Question 12

Question:

Take a list, say for example this one: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] and write a program that prints out all the elements of the list that are less than 5.

Solution:

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
for i in a:
    if i < 5:
        print(i)
```

Question 13

Question:

Let's say I give you a list saved in a variable: a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]. Write one line of Python that takes this list 'a' and makes a new list that has only the even elements of this list in it.

Solution:

```
a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
b = [number for number in a if number % 2 == 0]
print(b)
```

Question 14

Question:

Ask the user for a string and print out whether this string is a palindrome or not (A palindrome is a string that reads the same forwards and backwards).

Solution:

```
a=input("Please enter a word: ")
c = a.casefold()
b = reversed(c)
if list(c) == list(b):
   print("It is palindrome")
else:
   print("It is not palindrome")
```

Question 15

Question:

Take two lists, say for example these two: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89] b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13] and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.

Solution:

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
result = [i for i in set(a) if i in b]
print(result)
```

Question 16

Question:

Write a program to add a string to text file.

Solution:

```
file = open("testfile.txt","w")
file.write("Hello World")
file.write("This is our new text file")
file.write("and this is another line.")
file.write("Why? Because we can.")
file.close()
```

Question 17

Question:

Write a program to read a file and display its contents on console.

Solution:

```
with open('testfile.txt') as f:
        line = f.readline()
        while line:
                print(line)
                line = f.readline()
```

Question 18

Question:

Take two sets, say for example these two: a = {1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89} b = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13} and write a program that returns a set that contains only the elements that are common between the sets.

Solution:

```
a = {1, 1, 2, 2, 3, 5, 8, 13, 21, 34, 55, 89}
b = {1, 2, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13}
c = set(a) & set(b)
print(c)
```

Question 19

Question:

Write a program to split the characters of the given string into a list.

Solution:

```
s = "mystring"
l = list(s)
print (l)
```

Question 20

Question:

Create a program that asks the user for a number and then prints out a list of all the divisors of that number.

Solution:

```
n=int(input("Enter an integer: "))
print("The divisors of the number are: ")
for i in range(1,n+1):
    if(n%i==0):
        print(i)
```

Question 21

Question:

Write a program to Find the largest of three numbers.

Solution:

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
c = int(input("Enter third number: "))
if (a > b) and (a > c):
   largest = a
elif (b > a) and (b > c):
   largest = b
else:
   largest = c
print("The largest number is", largest)
```

Question 22

Question:

Write a Program to Find Absolute value of a Number.

Solution:

```
num = int(input("Enter a number: "))
if num >= 0:
                   print(num)
else:
                   print(-num)
```

Question 23

Question:

Write a program to Find the length of a String.

Solution:

```
print("Enter 'y' for exit.")
string = input("Enter a string: ")
if string == 'y':
    exit()
else:
    print("Length of the string =", len(string))
```

Question 24

Question:

Write a program to Print Natural Numbers from 1 to N.

Solution:

```
N = int(input("Please Enter any Number: "))
for i in range(1, N+1):
    print (i)
```

Question 25

Question:

Write a program to calculate the sum and average of Natural Numbers from 1 to N.

Solution:

```
N = int(input("Please Enter any Number: "))
sum = 0
for i in range(1,N+1):
  sum = sum + i
print(sum)
average = sum / N
print(average)
```

Question 26

Question:

Write a program to Print a Statement Any Number of Times.

Solution:

```
n = int(input("Please Enter any Number: "))
for i in range(n):
    print("hello world")
```

Question 27

Question:

Write a program To Multiply Two Numbers Using Function.

Solution:

```
def my_function():
    a = int(input("enter a number: "))
    b=int(input("enter a number: "))
    c= a*b
    return c
d = my_function()
print (d)
```

Question 28

Question:

Write a program To add an item to the end of the list.

Solution:

```
list1 = ["pen", "book", "ball"]
list1.append("bat")
print(list1)
```

Question 29

Question:

Write a program To remove an item from the list.

Solution:

```
list1 = ["pen", "book", "ball"]
list1.remove("ball")
print(list1)
```

Question 30

Question:

Write a program To print the number of elements in an array.

Solution:

```
list1 = ["pen", "book", "ball"]
a = len(list1)
print(a)
```

Question 31

Question:

Write a program To calculate the variance and standard deviation of the elements of the list.

Solution:

```
import numpy as np
a= [2,6,8,12,18,24,28,32]
```

```
variance= np.var(a)
std = np.std(a)
print(variance)
print(std)
```

Question 32

Question:

Write a program to get the difference between the two lists.

Solution:

```
list1 = [4, 5, 6, 7]
list2 = [4, 5]
print(list(set(list1) - set(list2)))
```

Question 33

Question:

Write a program to select an item randomly from a list.

Solution:

```
import random
list = ['Paper', 'Pencil', 'Book', 'Bag', 'Pen']
```

```
print(random.choice(list))
```

Question 34

Question:

Write a program that prints all the numbers from 0 to 6 except 2 and 6.

Solution:

```
for x in range(6):
    if (x == 2 or x==6):
        continue
    print(x)
```

Question 35

Question:

Write a program that takes input from the user and displays that input back in upper and lower cases.

Solution:

```
a = input("What's your name? ")
print(a.upper())
print(a.lower())
```

Question 36

Question:

Write a program to check whether a string starts with specified characters.

Solution:

```
string = "myw3schools.com"
print(string.startswith("w3s"))
```

Question 37

Question:

Write a program to create the multiplication table (from 1 to 10) of a number.

Solution:

```
n = int(input("Enter a number: "))
for i in range(1,11):
   print(n,'x',i,'=',n*i)
```

Question 38

Question:

Write a program to check a triangle is equilateral, isosceles or scalene.

Solution:

```
print("Enter lengths of the triangle sides: ")
a = int(input("a: "))
b = int(input("b: "))
c = int(input("c: "))
if a == b == c:
        print("Equilateral triangle")
elif a==b or b==c or c==a:
        print("isosceles triangle")
else:
        print("Scalene triangle")
```

Question 39

Question:

Write a program to sum of two given integers. However, if the sum is between 15 to 20 it will return 20.

Solution:

```
a = int(input("enter a number: "))
```

```
b = int(input("enter a number: "))
c= a+b
if c in range(15, 20):
        print (20)
else:
        print(c)
```

Question 40

Question:

Write a program to convert degree to radian.

Solution:

```
pi=22/7
degree = int(input("Input degrees: "))
radian = degree*(pi/180)
print(radian)
```

Question 41

Question:

Write a program to Generate a Random Number.

Solution:

```
import random
print(random.randint(0,9))
```

Question 42

Question:

Write a Program to find the semi-perimeter of triangle.

Solution:

```
a = int(input('Enter first side: '))
b = int(input('Enter second side: '))
c = int(input('Enter third side: '))
s = (a + b + c) / 2
print(s)
```

Question 43

Question:

Given a list of numbers, Iterate it and print only those numbers which are divisible of 2.

Solution:

```
List = [10, 20, 33, 46, 55]
for i in List:
    if (i % 2 == 0):
      print(i)
```

Question 44

Question:

Write a program to Multiply all numbers in the list.

Solution:

```
import numpy
list = [1, 2, 3]
result = numpy.prod(list)
print(result)
```

Question 45

Question:

Write a program to print ASCII Value of a character.

Solution:

```
a = 'j'
print("The ASCII value of '" + a + "' is", ord(a))
```

Beginner Java Exercises

Question 1

Question:

Write a program to print Hello World!.

Solution:

```
public class MyClass {
public static void main(String [] args) {
System.out.println("Hello, World!");
}
}
```

Question 2

Question:

Write a program to find the area of a circle.

Solution:

```
public class MyClass {
public static void main (String [] args) {
int r, area;
r = 2;
area = 3.14 * r * r;
System.out.println("The area of the circle = " + area);
}
}
```

Question 3

Question:

Write a program to find the sum of two numbers.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int a, b, sum;
a=1;
b=2;
sum = a + b;
System.out.println("The sum of a and b = " + sum);
}
}
```

Question 4

Question:

Write a program to find the square of a number.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int a, b;
a=2;
b = a * a;
System.out.println("The square of a = " + b);
}
}
```

Question 5

Question:

Write a program to find the greatest of two numbers.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int a, b;
a=2;
b =3;
if(a>b)
{
System.out.println("a is greater than b");
}
else
{
System.out.println("b is greater than a");
}
}
}
```

Question 6

Question:

Write a program to print the average of the elements in the array.

Solution:

```
public class MyClass {
public static void main(String[] args) {
int i, avg, sum = 0;
int [] num = {16, 18, 20, 25, 36};
for(i=0; i<5; i++)
sum = sum + num[i];
avg = sum/5;
System.out.println("Sum of the Elements in the array = " + sum);
System.out.println("Average of the Elements in the array = " + avg);
}
}
```

Question 7

Question:

Write a program such that a Switch (case) allows to make a decision from the number of choices, i.e., from the number of cases.

Solution:

```
public class MyClass {
public static void main(String[] args)throws Exception {
char ch;
System.out.print("Enter a character:");
ch = (char)System.in.read();
switch(ch)
{
case 'R':
System.out.print("Red");
break;
case 'W':
System.out.print("White");
break;
case 'Y':
System.out.print("Yellow");
break;
case 'G':
System.out.print("Green");
break;
default:
System.out.print("Error");
break;
}
}
}
```

Question 8

Question:

Write a program to read 10 numbers from the keyboard and find their sum and average.

Solution:

```
import java.util.Scanner;
public class MyClass {
public static void main(String [] args) {
int N1, N2, N3, N4, N5, N6, N7, N8, N9, N10, sum;
float X;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any ten Numbers: ");
N1 = scan.nextInt();
N2 = scan.nextInt();
N3 = scan.nextInt();
N4 = scan.nextInt();
N5 = scan.nextInt();
N6 = scan.nextInt();
N7 = scan.nextInt();
N8 = scan.nextInt();
N9 = scan.nextInt();
N10 = scan.nextInt();
sum = N1 + N2 + N3 + N4 + N5 + N6 + N7 + N8 + N9 + N10;
X = sum /10;
System.out.println("The sum of 10 numbers = " + sum);
System.out.println("The average of 10 numbers = " + X);
}
}
```

Question 9

Question:

Write a program to print the first 10 numbers starting from one together with their squares and cubes.

Solution:

```
public class MyClass {
public static void main(String[] args) throws Exception {
int i;
for( i=1; i<=10; i++)
System.out.println(" \n number = " + i + " its square = " + i*i + " its cube = " + i*i*i);
}
}
```

Question 10

Question:

Write a program:

If you enter a character M

Output must be: ch = M.

Solution:

```
public class MyClass {
```

```
public static void main(String[] args) throws Exception {
char c;
System.out.print("Enter a character:");
c = (char)System.in.read();
System.out.println("ch= " + c);
}
}
```

Question 11

Question:

Write a program to print the multiplication table of a number.

Solution:

```
import java.util.Scanner;
public class MyClass {
public static void main(String [] args) {
int n, i;
Scanner scan = new Scanner(System.in);
System.out.println("Enter a number: ");
n = scan.nextInt();
for( i=1; i<=5; i++)
System.out.println (n + " * " + i + " = " + n * i);
}
}
```

Question 12

Question:

Write a program to print the product of the first 10 digits.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int i, product = 1;
for( i=1; i<=10; i++)
product = product * i;
System.out.println("The product of the first 10 digits = " + product);
}
}
```

Question 13

Question:

Write a program to print whether the given number is positive or negative.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int a;
a = -35;
if(a>0)
{
System.out.println("Number is positive");
}
else
{
System.out.println("Number entered is negative");
}
}
}
```

Question 14

Question:

Write a program to check the equivalence of two numbers.

Solution:

```
import java.util.Scanner;
public class MyClass {
public static void main(String [] args) {
int x, y;
Scanner scan = new Scanner(System.in);
System.out.println("Enter a number: ");
x = scan.nextInt();
System.out.println("Enter a number: ");
y = scan.nextInt();
if(x-y==0)
{
System.out.println("The two numbers are equivalent");
}
else
{
```

```
System.out.println("The two numbers are not equivalent");
}
}
}
```

Question 15

Question:

Write a program to print the remainder of two numbers.

Solution:

```
import java.util.Scanner;
public class MyClass {
public static void main(String [] args) {
int a, b, c;
Scanner scan = new Scanner(System.in);
System.out.println("Enter a number: ");
a = scan.nextInt();
System.out.println("Enter a number: ");
b = scan.nextInt();
c = a%b;
System.out.println("The remainder of a and b = " + c);
}
}
```

Question 16

Question:

Write a program to print the given number is even or odd.

Solution:

```
import java.util.Scanner;
public class MyClass {
public static void main(String [] args) {
int a;
Scanner scan = new Scanner(System.in);
System.out.println("Enter a number: ");
a = scan.nextInt();
if(a%2 == 0)
{
System.out.println("The number is even");
}
else
{
System.out.println("The number is odd");
}
}
}
```

Question 17

Question:

Write a program to print the characters from A to Z.

Solution:

```
public class MyClass {
```

```
public static void main(String [] args) {
char a;
for( a='A'; a<='Z'; a++)
System.out.println("\n " + a);
}
}
```

Question 18

Question:

Write a program to find the incremented and decremented values of two numbers.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int a, b, c, d, e, f;
a = 10;
b=12;
c=a+1;
d=b+1;
e=a-1;
f=b-1;
System.out.print("The incremented value of a = "+ c);
System.out.print("The incremented value of b = "+ d);
System.out.print("The decremented value of a = "+ e);
System.out.print("The decremented value of b = "+ f);
}
}
```

Question 19

Question:

Write a program to calculate the simple interest.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int P,T, R, SI;
P = 1000;
T = 2;
R = 3;
SI = P*T*R/100;
System.out.println("The simple interest = " + SI);
}
}
```

Question 20

Question:

Write a program to Find the largest of three numbers.

Solution:

```
import java.util.Scanner;
public class MyClass {
```

```
public static void main(String [] args) {
int a, b, c;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any number:");
a = scan.nextInt();
System.out.println("Enter any number:");
b = scan.nextInt();
System.out.println("Enter any number:");
c = scan.nextInt();
if(a>b&&a>c)
{
System.out.println("a is greater than b and c");
}
else if(b>a&&b>c)
{
System.out.println("b is greater than a and c");
}
else
{
System.out.println("c is greater than b and a");
}
}
}
```

Question 21

Question:

Write a program to print the factorial of the entered number.

Solution:

```
import java.util.Scanner;
public class MyClass {
```

```
public static void main(String []args){
int i, n, fact=1 ;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any number:");
n = scan.nextInt();
for(i = 1; i <= n; i++)
        {
            fact = fact * i;
        }
        System.out.println("Factorial of " + n + " is: " + fact);
     }
}
```

Question 22

Question:

Write a program to print the length of the entered string.

Solution:

```
import java.util.Scanner;
public class MyClass {
public static void main(String[] args) {
String a;
Scanner scan = new Scanner(System.in);
System.out.print("Enter Your Name : ");
      a = scan.nextLine();
System.out.println("The length of the String is: " + a.length());
}
}
```

Question 23

Question:

Write a program to print the output:

Einstein [0] = E

Einstein [1] = I

Einstein [2] = N

Einstein [3] = S

Einstein [4] = T

Einstein [5] = E

Einstein [6] = I

Einstein [7] = N

Solution:

```
public class MyClass {
public static void main(String[] args) throws Exception{
int i;
char [] num = {'E' , 'I', 'N', 'S', 'T', 'E', 'I', 'N'};
for(i=0; i<8; i++)
System.out.println("Einstein [" + i + " ] = " + num[i]);
}
}
```

Question 24

Question:

Write a program to find square of a number using method.

Solution:

```
import java.util.Scanner;
public class MyClass {
public static void main(String[] args) {
int x;
Scanner scan = new Scanner(System.in);
System.out.println("Enter any number: ");
x = scan.nextInt();
System.out.println("Square of the number = " + square (x));
}
public static int square (int x){
return x*x;
}
}
```

Question 25

Question:

Write a program To print "hello world" 10 times.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int i;
for (i =1; i<=10; i ++)
System.out.println("\n hello world");
}
}
```

Question 26

Question:

Write a program to print first 5 numbers using do while loop statement.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int i =1;
do
{
System.out.println(" \n " + i++);
} while (i<=5);
}
}
```

Question 27

Question:

Write a program to print the output:

body [b] = b

body [o] = o

body [d] = d

body [y] = y

Solution:

```
public class MyClass {
public static void main(String[] args) throws Exception{
int i;
char [] body = {'b', 'o', 'd', 'y'};
for(i=0; i<4; i++)
System.out.println("body [" + body [i] + " ] = " + body [i]);
}
}
```

Question 28

Question:

Write a program to print the first ten natural numbers using while loop statement.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int i = 1;
while (i<=10)
{
System.out.println("\n " + i++);
}
}
}
```

Question 29

Question:

What will be the output of the below program:

```
public class MyClass {
public static void main(String []args) {
int i;
for (i=1; i<=5; i++) {
if (i==3) {
continue;
}
System.out.println("" + i);
}
}
}
```

Solution:

```
1
2
4
5
```

Question 30

Question:

Write a program to find the size of an array.

Solution:

```
public class MyClass {
public static void main(String[] args) {
   int num [] = {11, 22, 33, 44, 55, 66};
         System.out.println("Size of the array is: " + num.length);
}
}
```

Question 31

Question:

What would be the output of the following programs:

```
public class MyClass {
public static void main(String []args) {
int i;
for (i=1; i<=5; i++) {
if (i==3) {
break;
}
System.out.println("" + i);
}
}
}
```

Solution:

```
1
2
```

```
public class MyClass {
public static void main(String [] args) {
int x = 2;
System.out.println(" Square of a number = " + Math.pow((x), 2));
}
}
```

Solution:

```
Square of a number = 4.0
```

```
public class MyClass {
public static void main(String [] args) {
int i = 54;
int y = i<<1;
System.out.println("The value of y = " + y);
}
}
```

Solution:

```
The value of y = 108
```

```
public class MyClass {
```

```
public static void main(String [] args) {
int i = 54;
int y = i>>1;
System.out.println("The value of y = " + y);
}
}
```

Solution:

```
The value of y = 27
```

```
import java.util.Scanner;
public class MyClass {
public static void main(String [] args) {
String m;
Scanner in = new Scanner(System.in);
System.out.print("Enter the name: ");
m = in.nextLine();
System.out.println("The name you entered = " + m);
}
}
```

Solution:

```
Enter the name:
Dennis
The name you entered = Dennis
```

```
public class MyClass {
public static void main(String[] args) {
for( ; ; )
{
System.out.println("This loop will run forever.\n");
}
}
}
```

Solution:

```
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever. .........
```

```
public class MyClass {
public static void main(String [] args) {
System.out.println("Hello, World!");
System.exit(0);
System.out.println("Hello, World!");
}
}
```

Solution:

```
Hello,world!
```

Question 32

Question:

Write a program to check whether the person is a senior citizen or not.

Solution:

```
public class MyClass {
public static void main(String [] args) {
int age;
age=20;
if(age>= 60)
{
System.out.println("senior citizen");
}
else
{
System.out.println("not a senior citizen");
}
}
}
```

"Sometimes abstraction and encapsulation are at odds with performance — although not nearly as often as many developers believe — but it is always a good practice first to make your code right, and then make it fast."

— Brian Goetz

To search an element in a given array, it can be done in two ways:

- Linear search
- Binary search

Linear Search:

Linear search is a very basic and simple search algorithm. In this type of search, a sequential search is made over all elements one by one. Every element is checked and if a match is found then that particular element is returned, otherwise the search continues till the end of the data collection.

For Example:

To search the element 17 it will go step by step in a sequence order:

8	10	12	15	17	20	25
17						

Match not found

8	10	12	15	17	20	25
	17					

Match not found

8	10	12	15	17	20	25
		17				

Match not found

8	10	12	15	17	20	25
			17			

Match not found

8	10	12	15	17	20	25
				17		

Match found

Element 17 is returned.

Linear search (whose running time increases linearly with the number of elements in the array. For example if number of elements is doubled then, on average, the search would take twice as long) is rarely used practically because other search algorithms such as the binary search algorithm and hash tables allow significantly faster searching comparison to linear search.

Binary Search:

Binary Search is applied on the sorted array or list. In binary search, we first compare the value with the elements in the middle position of the array. If the value is matched, then we return the value. If the value is less than the middle element, then it must lie in the lower half of the array and if it's greater than the element then it must lie in the upper half of the array. We repeat this procedure on the lower (or upper) half of the array. Binary Search is useful when there are large numbers of elements in an array.

We shall learn the process of binary search with a pictorial example. The following is our sorted array and let us assume that we need to search the location of value 31 using binary search.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

First, we shall determine half of the array by using this formula −

$$\text{mid} = \text{low} + \frac{(\text{high} - \text{low})}{2}$$

Here it is, $0 + \frac{(9-0)}{2} = 4$ (integer value of 4.5). So, 4 is the mid of the array.

				↓					
10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

Now we compare the value stored at location 4, with the value being searched, i.e. 31. We find that the value at location 4 is 27, which is not a match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

We change our low to mid + 1 and find the new mid value again.

```
low = mid + 1
```

$$\text{mid} = \text{low} + \frac{(\text{high} - \text{low})}{2}$$

Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.

							↓		
10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

The value stored at location 7 is not a match; rather it is more than what we are looking for. So, the value must be in the lower part from this location.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

Hence, we calculate the mid again. This time it is 5.

					↓				
10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

We compare the value stored at location 5 with our target value. We find that it is a match.

10	14	19	26	27	31	33	35	42	44
0	1	2	3	4	5	6	7	8	9

We conclude that the target value 31 is stored at location 5.

Note that it is a good programming practice to define constants in CAPITALS.

```
#include<stdio.h>

int main(){

constint  LENGTH =10;

constint  WIDTH =5;

constchar NEWLINE ='\n';

int area;

area = LENGTH * WIDTH;

printf("value of area : %d", area);

printf("%c", NEWLINE);

return0;

}
```

C Exercise: Binary search

```
#include<stdio.h>
 void main()
 {
 int  arra[100],i,n,x,f,l,m,flag=0;
 printf("Input no. of elements in  an array\n");
 scanf("%d", &n);
 printf("Input  %d value in ascending order\n",n);
 for(i=0; i<n; i++)
 scanf("%d", &arra[i]);
 printf("Input  the value to be search : ");
 scanf("%d", &x);

 /* Binary Search logic */

 f=0;l=n-1;
 while(f<=l)
 {
```

```
m=(f+l)/2;
if(x==arra[m])
{
flag=1;
break;
}
else if(x<arra[m])
l=m-1;
else
f=m+1;
}
if(flag==0)
printf("%d  value not found\n", x);
else
printf("%d value  found at %d position\n", x, m);
}
```

Output on the screen:

```
Input no. of elements in an array
3
Input 3 value in ascending order
15
18
20
Input the value to be search: 15
15 value found at 0 position
```

Generate Alert Box if user tries to change text in text input field

Code:

```
<FORM>
<INPUT TYPE="text" VALUE= "dont change" NAME = "leavebutton"
onChange= "alert('Please dont change this')">
</FORM>
```

Animate the background color of a document

Code:

```
<SCRIPT LANGUAGE= "javascript">

setTimeout("document.bgColor='white'", 1000)
setTimeout("document.bgColor='lightpink'", 1500)
setTimeout("document.bgColor = 'pink'", 2000)
setTimeout("document.bgColor =  'deeppink'", 2500)
setTimeout("document.bgColor = 'red'", 3000)
setTimeout("document.bgColor = 'tomato'", 3500)
setTimeout("document.bgColor = 'darkred'", 4000)

</SCRIPT>
```

User assignment of a property and dynamic generation of a Web page

Code:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE = "Javascript">
document.firstline = "Welcome to this page"
</SCRIPT>
<TITLE>load demo</TITLE>
</HEAD>

<BODY>
<SCRIPT>
document.open()
document.write(document.firstline)
document.open()
</SCRIPT>

</BODY>
</HTML>
```

Sample Script to Animate Text in Text Field

Code:

```
<HTML>
<HEAD>
<TITLE> Animated Text</TITLE>
</HEAD>
<BODY>

<FORM NAME="f1">
<TABLE>
<TR> <TD> <INPUT NAME="ta1" TYPE="text" SIZE="20">
<TD> <INPUT NAME="ta2" TYPE="text" SIZE="20">
<TD> <INPUT NAME="ta3" TYPE="text" SIZE="20">
</TABLE></FORM>

<HR>

<FORM NAME="f2" ACTION="http://netadd.com/nam.cgi" METHOD="POST">
<CENTER>
Name <INPUT NAME="pername" TYPE="text" SIZE="20"> Name<P>
Age <INPUT NAME="perage" TYPE="text" SIZE="5"> Age<P>
Occupation <INPUT NAME="perocc" TYPE="text" SIZE="20">Occupation <P>
<INPUT TYPE="Submit" VALUE="Submit">
<INPUT TYPE="Reset" VALUE="Reset">
</CENTER>
</FORM>

<HR>
<FORM NAME="f3">
<TABLE>
<TR> <TD> <INPUT NAME="ta4" TYPE="text" SIZE="20">
<TD> <INPUT NAME="ta5" TYPE="text" SIZE="20">
<TD> <INPUT NAME="ta6" TYPE="text" SIZE="20">
</TABLE></FORM>

<SCRIPT LANGUAGE= "javascript">
setTimeout("document.f1.ta1.value = 'Answer Soon'", 1000)
setTimeout("document.f1.ta1.value = ''", 1300)
setTimeout("document.f1.ta2.value = 'Answer Soon'", 1600)
setTimeout("document.f1.ta2.value = ''", 1900)
setTimeout("document.f1.ta3.value = 'Answer Soon'", 2200)
setTimeout("document.f1.ta3.value = ''", 2500)
setTimeout("document.f3.ta4.value = 'Answer Soon'", 2800)
setTimeout("document.f3.ta4.value = ''", 3100)
setTimeout("document.f3.ta5.value = 'Answer Soon'", 3400)
setTimeout("document.f3.ta5.value = ''", 3700)
setTimeout("document.f3.ta6.value = 'Answer Soon'", 4000)
setTimeout("document.f3.ta6.value = ''", 4300)
</SCRIPT>
</BODY>
</HTML>
```

Using string properties to set characteristics of text on a page

Code:

```
<SCRIPT>
//assigns value to variable
test ="What is all this?"

// opens document and uses methods to modify text characteristics
document.open()
document.write(test.bold()+"<P>")
document.write(test.fontsize(7)+"<P>")
document.write(test.fontcolor("red")+"<P>")
document.write(test.toUpperCase()+"<P>")

//assigns multiple characteristics to text
document.write(test.italics().fontsize(6).fontcolor("green")+"<P>")
document.open()
</SCRIPT>
```

Using Substrings to Generate Scrolling Banners

Code:

```
<HTML>
<HEAD><TITLE> Banner</TITLE>

<SCRIPT LANGUAGE= "javascript">

// Puts the text to scroll into variable called sent
// uses length propert to assess its length and put into variable slen
// initalizes a,b,n, and subsent variables

var sent = "This is a demonstration of a banner moving from the left to right. It makes use of the substring property of Javascript to make an interesting display"
var slen = sent.length
var siz = 25
var a = -3, b = 0
var subsent = "x"

// Creates a function to capture substrings of sent

function makeSub(a,b) {
subsent = sent.substring(a,b) ;
return subsent;
}

//Creates a function that increments the indexes of the substring
//each time and calls the makeSub() function to generate strings
//a indicates start of substring and siz indicates size of string required

function newMake() {
a = a + 3;
b = a + siz
makeSub(a,b);
return subsent
}

//function uses loop to get changing substrings of target
//repeatedly calls newMake to get next substring
//uses setTimeout() command to arrange for substrings to display
// at specified times
```

```
function doIt() {
for (var i = 1; i <= slen ; i++) {
setTimeout("document.z.textdisplay.value = newMake()", i*300);
setTimeout("window.status = newMake()", i*300);
}
}

</SCRIPT>
</HEAD>

<BODY >
<HR> <CENTER>
<FORM NAME="z">
<INPUT NAME="textdisplay" TYPE="text" SIZE=25> <P>
<INPUT NAME="doit" Type="button" value = "Run Banner" onClick = "doIt()">
</FORM></CENTER>

<HR>

</BODY>
</HTML>
```

What is HTML?

Before you continue, you should have a basic understanding of how to use a browser to view pages on the Web. If you want to study these subjects first, please read **The Internet For Dummies**, 12th Edition, from Wiley Publishing.

- HTML is a language for describing Web pages
- HTML stands for HyperText Markup Language

HTML Fundamentals

HTML Headings

Code:

```
<html>
<body>
<h1>This is Heading 1</h1>
<h2>Heading 2 is Smaller</h2>
<h3>Heading 3 is Smaller Still</h3>
</body>
</html>
```

HTML Paragraphs

Code:

```
<html>
<body>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
</body>
</html>
```

HTML Links

Code:

```
<html>
<body>
<a href="http://www.google.com">This is a link to the google Web site.</a>
</body>
</html>
```

Code:

```
<html>
```

```
<body>
<a href="http://www.google.com" style="text-decoration: none">This is a link to the
google Web site.</a>
</body>
</html>
```

Code:

```
<html>
<head>
<style type="text/css">
  a.nounderline {text-decoration: none;}
</style>
</head>
<body>
<a href="http://www.google.com" class="nounderline">This is a link to the google Web
site.</a>
</body>
</html>
```

HTML Images

Code:

```
<html>
<body>
<img src="images/img1.jpg" width="295" height="175" />
</body>
</html>
```

HTML Rules (Lines)

Code:

```
<html>
<body>
<p>The hr tag defines a horizontal rule:</p> <hr/>
<p>This is a paragraph</p>
<hr/>
<p>This is a paragraph</p>
<hr/>
<p>This is a paragraph</p>
</body>
</html>
```

HTML Line Breaks

Code:

```
<html>
<body>
<p>This is<br/>a para-<br/>graph with line breaks</p>
</body>
</html>
```

Text Formatting

Code:

```
<html>
<body>
<p><b>This text is bold</b></p>
<p><strong>This text is strong</strong></p>
<p><big>This text is big</big></p>
<p><em>This text is emphasized</em></p>
<p><i>This text is italic</i></p>
<p><small>This text is small</small></p>
<p>This is<sub> subscript</sub> and <sup>superscript</sup></ p>
</body>
</html>
```

Preformatted Text

Code:

```
<html>
<body>
<pre>
This is
preformatted text.
It preserves        both spaces
and line breaks and shows the text in a monospace font. </pre>
<p>The pre tag is good for displaying computer code:</p>
<pre>
for i = 1 to 10
print i
next i
</pre>
</body>
</html>
```

Computer Output Tags

Code:

```
<html>
<body>
<code>Computer code</code>
<br/>
<kbd>Keyboard input</kbd>
<br/>
<tt>Teletype text</tt>
<br/>
<samp>Sample text</samp>
<br/>
<var>Computer variable</var>
<br/>
<p>
<b>Note:</b> These tags are often used to display computer/ programming code on the
page.
```

```
</p>
</body>
</html>
```

Address

Code:

```
<html>
<body>
<address>
Donald Duck<br>
BOX 555<br>
Disneyland<br>
USA
</address>
</body>
</html>
```

Special HTML codes

Char	Numeric code	Named code	Description
	&#09;		horizontal tab
	&#10;		line feed
	&#13;		carriage return / enter
	&#160;	&nbsp;	non-breaking space

Regular HTML character codes

Char	Numeric code	Named code	Description
	&#32;		space
!	&#33;		exclamation mark
"	&#34;	&quot;	double quote
#	&#35;		number
$	&#36;		dollar
%	&#37;		percent
&	&#38;	&amp;	ampersand
'	&#39;	&apos;	single quote
(	&#40;		left parenthesis
)	&#41;		right parenthesis
*	&#42;		asterisk
+	&#43;		plus
,	&#44;		comma
-	&#45;		minus
.	&#46;		period
/	&#47;		slash
0	&#48;		zero
1	&#49;		one
2	&#50;		two
3	&#51;		three
4	&#52;		four
5	&#53;		five
6	&#54;		six
7	&#55;		seven

Char	Numeric code	Named code	Description
8	&#56;		eight
9	&#57;		nine
:	&#58;		colon
;	&#59;		semicolon
<	&#60;	&lt;	less than
=	&#61;		equality sign
>	&#62;	&gt;	greater than
?	&#63;		question mark
@	&#64;		at sign
A	&#65;		
B	&#66;		
C	&#67;		
D	&#68;		
E	&#69;		
F	&#70;		
G	&#71;		
H	&#72;		
I	&#73;		
J	&#74;		
K	&#75;		
L	&#76;		
M	&#77;		
N	&#78;		
O	&#79;		

Char	Numeric code	Named code	Description
P	&#80;		
Q	&#81;		
R	&#82;		
S	&#83;		
T	&#84;		
U	&#85;		
V	&#86;		
W	&#87;		
X	&#88;		
Y	&#89;		
Z	&#90;		
[	&#91;		left square bracket
\	&#92;		backslash
]	&#93;		right square bracket
^	&#94;		caret / circumflex
_	&#95;		underscore
`	&#96;		grave / accent
a	&#97;		
b	&#98;		
c	&#99;		
d	&#100;		
e	&#101;		
f	&#102;		
g	&#103;		

Char	Numeric code	Named code	Description
h	&#104;		
i	&#105;		
j	&#106;		
k	&#107;		
l	&#108;		
m	&#109;		
n	&#110;		
o	&#111;		
p	&#112;		
q	&#113;		
r	&#114;		
s	&#115;		
t	&#116;		
u	&#117		
v	&#118;		
w	&#119;		
x	&#120;		
y	&#121;		
z	&#122;		
{	&#123;		left curly bracket
|	&#124;		vertical bar
}	&#125;		right curly bracket
~	&#126;		tilde

Extra codes

Char	Numeric code	Named code	Description
	&#160;	&nbsp;	non-breaking space
¡	&#161;	&iexcl;	inverted exclamation mark
¢	&#162;	&cent;	cent sign
£	&#163;	&pound;	pound sign
¤	&#164;	&curren;	currency sign
¥	&#165;	&yen;	yen sign
¦	&#166;	&brvbar;	broken bar
§	&#167;	&sect;	section sign
¨	&#168;	&uml;	diaeresis
©	&#169;	&copy;	copyright sign
ª	&#170;	&ordf;	feminine ordinal indicator
«	&#171;	&laquo;	left pointing guillemet
¬	&#172;	&not;	not sign
	&#173;	&shy;	soft hyphen
®	&#174;	&reg;	registered sign
¯	&#175;	&macr;	macron
°	&#176;	&deg;	degree sign
±	&#177;	&plusmn;	plus-minus sign
²	&#178;	&sup2;	superscript two
³	&#179;	&sup3;	superscript three
´	&#180;	&acute;	acute accent
µ	&#181;	&micro;	micro sign

Char	Numeric code	Named code	Description
¶	&#182;	&para;	paragraph sign
·	&#183;	&middot;	middle dot
¸	&#184;	&cedil;	spacing cedilla
¹	&#185;	&sup1;	superscript one
º	&#186;	&ordm;	masculine ordinal indicator
»	&#187;	&raquo;	right pointing guillemet
¼	&#188;	&frac14;	fraction one quarter
½	&#189;	&frac12;	fraction one half
¾	&#190;	&frac34;	fraction three quarters
¿	&#191;	&iquest;	inverted question mark
À	&#192;	&Agrave;	capital A with grave
Á	&#193;	&Aacute;	capital A with acute
Â	&#194;	&Acirc;	capital A with circumflex
Ã	&#195;	&Atilde;	capital A with tilde
Ä	&#196;	&Auml;	capital A with diaeresis
Å	&#197;	&Aring;	capital A with ring
Æ	&#198;	&AElig;	capital AE
Ç	&#199;	&Ccedil;	capital C with cedilla
È	&#200;	&Egrave;	capital E with grave
É	&#201;	&Eacute;	capital E with acute
Ê	&#202;	&Ecirc;	capital E with circumflex
Ë	&#203;	&Euml;	capital E with diaeresis
Ì	&#204;	&Igrave;	capital I with grave
Í	&#205;	&Iacute;	capital I with acute

Char	Numeric code	Named code	Description
Î	&#206;	&Icirc;	capital I with circumflex
Ï	&#207;	&Iuml;	capital I with diaeresis
Ð	&#208;	&ETH;	capital ETH
Ñ	&#209;	&Ntilde;	capital N with tilde
Ò	&#210;	&Ograve;	capital O with grave
Ó	&#211;	&Oacute;	capital O with acute
Ô	&#212;	&Ocirc;	capital O with circumflex
Õ	&#213;	&Otilde;	capital O with tilde
Ö	&#214;	&Ouml;	capital O with diaeresis
×	&#215;	&times;	multiplication sign
Ø	&#216;	&Oslash;	capital O with stroke
Ù	&#217;	&Ugrave;	capital U with grave
Ú	&#218;	&Uacute;	capital U with acute
Û	&#219;	&Ucirc;	capital U with circumflex
Ü	&#220;	&Uuml;	capital U with diaeresis
Ý	&#221;	&Yacute;	capital Y with acute
Þ	&#222;	&THORN;	capital THORN
ß	&#223;	&szlig;	small sharp s
à	&#224;	&agrave;	small a with grave
á	&#225;	&aacute;	small a with acute
â	&#226;	&;	small a with circumflex
ã	&#227;	&atilde;	small a with tilde
ä	&#228;	&auml;	small a with diaeresis
å	&#229;	&aring;	small a with ring above

Char	Numeric code	Named code	Description
æ	&#230;	&aelig;	small ae
ç	&#231;	&ccedil;	small c with cedilla
è	&#232;	&egrave;	small e with grave
é	&#233;	&eacute;	small e with acute
ê	&#234;	&ecirc;	small e with circumflex
ë	&#235;	&euml;	small e with diaeresis
ì	&#236;	&igrave;	small i with grave
í	&#237;	&iacute;	small i with acute
î	&#238;	&icirc;	small i with circumflex
ï	&#239;	&iuml;	small i with diaeresis
ð	&#240;	&eth;	small eth
ñ	&#241;	&ntilde;	small n with tilde
ò	&#242;	&ograve;	small o with grave
ó	&#243;	&oacute;	small o with acute
ô	&#244;	&ocirc;	small o with circumflex
õ	&#245;	&otilde;	small o with tilde
ö	&#246;	&ouml;	small o with diaeresis
÷	&#247;	&divide;	division sign
ø	&#248;	&oslash;	small o with stroke
ù	&#249;	&ugrave;	small u with grave
ú	&#250;	&uacute;	small u with acute
û	&#251;	&ucirc;	small u with circumflex
ü	&#252;	&uuml;	small u with diaeresis
ý	&#253;	&yacute;	small y with acute

Char	Numeric code	Named code	Description
þ	&#254;	&thorn;	small thorn
ÿ	&#255;	&yuml;	small y with diaeresis

Symbols codes

Char	Numeric code	Named code	Description
&	&#38;	&amp;	ampersand
•	&#8226;	&bull;	bullet
◦	&#9702;		white bullet
∙	&#8729;		bullet operator
‣	&#8227;		triangular bullet
⁃	&#8259;		hyphen bullet
°	&#176;	&deg;	degree
∞	&#8734;	&infin;	infinity
‰	&#8240;	&permil;	per-mille
⋅	&#8901;	&sdot;	multiplication dot
±	&#177;	&plusmn;	plus-minus
†	&#8224;	&dagger;	hermitian
—	&#8212;	&mdash;	
¬	&#172;	&not;	
µ		&micro;	

Currency codes

Char	Numeric code	Named code	Description
$	&#36;		dollar
€	&#8364;	&euro;	euro
£	&#163;	&pound;	pound
¥	&#165;	&yen;	yen / yuan
¢	&#162;	&cent;	cent
₹	&#8377;		indean Rupee
₨	&#8360;		rupee
₱	&#8369;		peso
₩	&#8361;		korean won
฿	&#3647;		thai baht
₫	&#8363;		dong
₪	&#8362;		shekel

Intellectual property codes

Char	Numeric code	Named code	Description
©	&#169;	&copy;	copyright
®	&#174;	&reg;	registered trademark
℗	&#8471;		sound recording copyright
™	&#8482;	&trade;	trademark
℠	&#8480;		service mark

Greek alphabet codes

Char	Numeric code	Named code	Description
α	&#945;	&alpha;	small alpha
β	&#946;	&beta;	small beta
γ	&#947;	&gamma;	small gamma
δ	&#948;	&delta;	small delta
ε	&#949;	&epsilon;	small epsilon
ζ	&#950;	&zeta;	small zeta
η	&#951;	&eta;	small eta
θ	&#952;	&theta;	small theta
ι	&#953;	&iota;	small iota
κ	&#954;	&kappa;	small kappa
λ	&#955;	&lambda;	small lambda
μ	&#956;	&mu;	small mu
ν	&#957;	&nu;	small nu
ξ	&#958;	&xi;	small xi
ο	&#959;	&omicron;	small omicron
π	&#960;	&pi;	small pi
ρ	&#961;	&rho;	small rho
σ	&#963;	&sigma;	small sigma
τ	&#964;	&tau;	small tau
υ	&#965;	&upsilon;	small upsilon
φ	&#966;	&phi;	small phi
χ	&#967;	&chi;	small chi
ψ	&#968;	&psi;	small psi

Char	Numeric code	Named code	Description
ω	&#969;	&omega;	small omega
Α	&#913;	&Alpha;	capital alpha
Β	&#914;	&Beta;	capital beta
Γ	&#915;	&Gamma;	capital gamma
Δ	&#916;	&Delta;	capital delta
Ε	&#917;	&Epsilon;	capital epsilon
Ζ	&#918;	&Zeta;	capital zeta
Η	&#919;	&Eta;	capital eta
Θ	&#920;	&Theta;	capital theta
Ι	&#921;	&Iota;	capital iota
Κ	&#922;	&Kappa;	capital kappa
Λ	&#923;	&Lambda;	capital lambda
Μ	&#924;	&Mu;	capital mu
Ν	&#925;	&Nu;	capital nu
Ξ	&#926;	&Xi;	capital xi
Ο	&#927;	&Omicron;	capital omicron
Π	&#928;	&Pi;	capital pi
Ρ	&#929;	&Rho;	capital rho
Σ	&#931;	&Sigma;	capital sigma
Τ	&#932;	&Tau;	capital tau
Υ	&#933;	&Upsilon;	capital upsilon
Φ	&#934;	&Phi;	capital phi
Χ	&#935;	&Chi;	capital chi
Ψ	&#936;	&Psi;	capital psi

Char	Numeric code	Named code	Description
Ω	&#937;	&Omega;	capital omega

Abbreviations and Acronyms

Code:

```
<html>
<body>
<abbr title="United Nations">UN</abbr>
<br/>
<acronym title="World Wide Web">WWW</acronym>
<p>The title attribute is used to show the spelled-out version when holding the mouse
pointer over the acronym or abbreviation.</p>
</body>
</html>
```

Text Direction

Code:

```
<html>
<body>
<p>
If your browser supports bidirectional override (bdo), the next line will be written
from the right to the left (rtl):
</p>
<bdo dir="rtl">
Here is some backward text
</bdo>
</body>
</html>
```

Quotations

Code:

```
<html>
<body>
A blockquote quotation:
<blockquote>
This is a long quotation. This is a long quotation. This is a long quotation. This is a
long quotation. This is a long quotation.
</blockquote>
<p><b>The browser inserts line breaks and margins for a blockquote element.</b></p>
A short quotation:
<q>This is a short quotation</q>
<p><b>The q element does not render as anything special.</ b></p>
</body>
</html>
```

Deleted and Inserted Text

Code:

```
<html>
<body>
<p>
a dozen is
<del>twenty</del>
<ins>twelve</ins>
pieces
</p>
<p>
Most browsers will <del>overstrike</del> deleted text and <ins>underscore</ins>
inserted text.
</p>
<p>
Some older browsers will display deleted or inserted text as plain text.
</p>
</body>
</html>
```

The HTML Style Attribute

Code:

```
<html>
<body style="background-color:Gray;">
<h1>Look! Styles and colors</h1>
<p style="font-family:verdana;color:red"> This text is in Verdana and red</p>
<p style="font-family:times;color:green">
This text is in Times and green</p>
<p style="font-size:30px">This text is 30 pixels high</p>
</body>
</html>
```

Background Color

Code:

```
<html>
<body style="background-color:gray"> <h2>Look: Colored Background!</h2>
</body>
</html>
```

Code:

```
<html>
<body bgcolor="gray">
<h2>Look: Colored Background!</h2>
<p>For future-proof HTML, use HTML styles instead:</p> <p>style="background-
color:gray"</p>
</body>
</html>
```

Font Family, Color, and Size

Code:

```
<html>
<body>
<h1 style="font-family:verdana">A heading</h1>
<p style="font-family:courier new; color:red; font-size:20px;">A paragraph</p>
</body>
</html>
```

Code:

```
<html>
<body>
<p><font size="2" face="Verdana">
This is a paragraph.
</font></p>
<p><font size="5" face="Times" color="red"> This is another paragraph.</font></p>
</body>
</html>
```

Text Alignment

Code:

```
<html>
<body>
<h1 style="text-align:center">This is heading 1</h1>
<p>The heading above is aligned to the center of this page. The heading above is
aligned to the center of this page. The heading above is aligned to the center of this
page. </p>
</body>
</html>
```

Code:

```
<html>
<body>
<h1 align="center">This is heading 1</h1>
<p>The heading above is aligned to the center of this page. The heading above is
aligned to the center of this page. The heading above is aligned to the center of this
page.</ p>
</body>
</html>
```

HTML Colors

Code:

```
<!DOCTYPE html>
<html>
<body>
<h2 style="background-color:#FF0000"> Background-color set by using #FF0000 </h2>
<h2 style="background-color:#00FF00"> Background-color set by using #00FF00 </h2>
<h2 style="background-color:#0000FF"> Background-color set by using #0000FF </h2>
<h2 style="background-color:#FFFF00"> Background-color set by using #FFFF00 </h2>
<h2 style="background-color:#FF00FF"> Background-color set by using #FF00FF </h2>
<h2 style="background-color:#00FFFF"> Background-color set by using #00FFFF </h2>
</body>
</html>
```

Code:

```
<!DOCTYPE html>
<html>
<body>
<h2 style="background-color:rgb(0,0,0);color:white"> Background-color set by using
rgb(0,0,0) </h2>
```

```
<h2 style="background-color:rgb(90,90,90);color:white"> Background-color set by using rgb(90,90,90) </h2>
<h2 style="background-color:rgb(128,128,128);color:white"> Background-color set by using rgb(128,128,128) </h2>
<h2 style="background-color:rgb(200,200,200);color:white"> Background-color set by using rgb(200,200,200) </h2>
<h2 style="background-color:rgb(255,255,255);"> Background-color set by using rgb(255,255,255) </h2>
</body>
</html>
```

Code:

```
<!DOCTYPE html>
<html>
<body>
<h2 style="background-color:#000000;color:white"> Background-color set by using #000000 </h2>
<h2 style="background-color:#808080;color:white"> Background-color set by using #808080 </h2>
<h2 style="background-color:#FFFFFF;"> Background-color set by using #FFFFFF </h2>
</body>
</html>
```

Color	HTML Code
Alice Blue	#F0F8FF
Antique White	#FAEBD7
Aqua	#00FFFF
Aquamarine	#7FFFD4
Azure	#F0FFFF
Beige	#F5F5DC
Bisque	#FFE4C4
Black	#000000

Blanched Almond	#FFEBCD
Blue	#0000FF
Blue Violet	#8A2BE2
Brown	#A52A2A
Burly Wood	#DEB887
Cadet Blue	#5F9EA0
Chartreuse	#7FFF00
Chocolate	#D2691E
Coral	#FF7F50
Cornflower Blue	#6495ED
Corn silk	#FFF8DC
Crimson	#DC143C
Cyan	#00FFFF
Dark Blue	#00008B
Dark Cyan	#008B8B
Dark Goldenrod	#B8860B
Dark Gray	#A9A9A9
Dark Grey	#A9A9A9
Dark Green	#006400
Dark Khaki	#BDB76B
Dark Magenta	#8B008B
Dark Olive Green	#556B2F
Dark orange	#FF8C00
Dark Orchid	#9932CC
Dark Red	#8B0000
Dark Salmon	#E9967A
Dark Sea Green	#8FBC8F
Dark Slate Blue	#483D8B
Dark Slate Gray	#2F4F4F
Beige	#F5F5DC
Dark Slate Grey	#2F4F4F
Dark Turquoise	#00CED1
Dark Violet	#9400D3
Deep Pink	#FF1493
Deep Sky Blue	#00BFFF
Dim Gray	#696969
Dim Grey	#696969

Dodger Blue	#1E90FF
Fire Brick	#B22222
Floral White	#FFFAF0
Forest Green	#228B22
Fuchsia	#FF00FF
Gainsboro	#DCDCDC
Ghost White	#F8F8FF
Gold	#FFD700
Goldenrod	#DAA520
Gray	#808080
Grey	#808080
Green	#008000
Green Yellow	#ADFF2F
Honeydew	#F0FFF0
Hot Pink	#FF69B4
Indian Red	#CD5C5C
Indigo	#4B0082
Ivory	#FFFFF0
Khaki	#F0E68C
Lavender	#E6E6FA
Lavender Blush	#FFF0F5
Lawn Green	#7CFC00
Lemon Chiffon	#FFFACD
Light Blue	#ADD8E6
Light Coral	#F08080
Light Cyan	#E0FFFF
Light Golden Rod Yellow	#FAFAD2
Light Gray	#D3D3D3
Light Grey	#D3D3D3
Light Green	#90EE90
Light Pink	#FFB6C1
Light Salmon	#FFA07A
Light Sea Green	#20B2AA
Light Sky Blue	#87CEFA
Yellow Green	#9ACD32
Yellow	#FFFF00
White Smoke	#F5F5F5

White	#FFFFFF
Wheat	#F5DEB3
Violet	#EE82EE
Light Slate Gray	#778899
Light Slate Grey	#778899
Light Steel Blue	#B0C4DE
Light yellow	#FFFFE0
Lime	#00FF00
Lime Green	#32CD32
Linen	#FAF0E6
Magenta	#FF00FF
Maroon	#800000
Turquoise	#40E0D0
Tomato	#FF6347
Thistle	#D8BFD8
Teal	#008080
Tan	#D2B48C
Medium Aqua Marine	#66CDAA
Steel Blue	#4682B4
Medium Blue	#0000CD
Medium Orchid	#BA55D3
Spring Green	#00FF7F
Snow	#FFFAFA
Medium Purple	#9370D8
Medium Sea Green	#3CB371
Medium Slate Blue	#7B68EE
Medium Spring Green	#00FA9A
Slate Gray	#708090
Slate Grey	#708090
Silver	#C0C0C0
Sienna	#A0522D
Medium Turquoise	#48D1CC
Medium Violet Red	#C71585
Midnight Blue	#191970
Olive	#808000
Mint Cream	#F5FFFA
Misty Rose	#ffe4e1

Moccasin	#FFE4B5
Peru	#CD853F
Plum	#DDA0DD
Navajo White	#FFDEAD
Navy	#000080
Old Lace	#FDF5E6
Olive Drab	#6B8E23
Orange	#FFA500
Orange Red	#FF4500
Orchid	#DA70D6
Pink	#FFC0CB
Salmon	#FA8072
Pale Golden Rod	#EEE8AA
Slate Blue	#6A5ACD
Sky Blue	#87CEEB
Sea Shell	#FFF5EE
Pale Green	#98FB98
Pale Turquoise	#AFEEEE
Pale Violet Red	#D87093
Sea Green	#2E8B57
Sandy Brown	#F4A460
Papaya Whip	#FFEFD5
Peach Puff	#FFDAB9
Powder Blue	#B0E0E6
Purple	#800080
Red	#FF0000
Rosy Brown	#BC8F8F
Royal Blue	#4169E1
Saddle Brown	#8B4513

CSS Properties:

align-content Specifies the alignment between the lines inside a flexible container when the items do not use all available space

align-items	Specifies the alignment for items inside a flexible container
align-self	Specifies the alignment for selected items inside a flexible container
all	Resets all properties (except unicode-bidi and direction)
animation	A shorthand property for all the *animation-** properties
animation-delay	Specifies a delay for the start of an animation
animation-direction	Specifies whether an animation should be played forwards, backwards or in alternate cycles
animation-duration	Specifies how long an animation should take to complete one cycle
animation-fill-mode	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
animation-iteration-count	Specifies the number of times an animation should be played
animation-name	Specifies a name for the @keyframes animation
animation-play-state	Specifies whether the animation is running or paused
animation-timing-function	Specifies the speed curve of an animation

backface-visibility	Defines whether or not the back face of an element should be visible when facing the user
background	A shorthand property for all the *background-** properties
background-attachment	Sets whether a background image scrolls with the rest of the page, or is fixed
background-blend-mode	Specifies the blending mode of each background layer (color/image)
background-clip	Defines how far the background (color or image) should extend within an element
background-color	Specifies the background color of an element
background-image	Specifies one or more background images for an element
background-origin	Specifies the origin position of a background image
background-position	Specifies the position of a background image
background-repeat	Sets if/how a background image will be repeated
background-size	Specifies the size of the background images
border	A shorthand property for *border-width, border-style* and *border-color*
border-bottom	A shorthand property for *border-bottom-width, border-bottom-style* and *border-bottom-color*
border-bottom-color	Sets the color of the bottom border
border-bottom-left-radius	Defines the radius of the border of the bottom-left corner

border-bottom-right-radius	Defines the radius of the border of the bottom-right corner
border-bottom-style	Sets the style of the bottom border
border-bottom-width	Sets the width of the bottom border
border-collapse	Sets whether table borders should collapse into a single border or be separated
border-color	Sets the color of the four borders
border-image	A shorthand property for all the *border-image-** properties
border-image-outset	Specifies the amount by which the border image area extends beyond the border box
border-image-repeat	Specifies whether the border image should be repeated, rounded or stretched
border-image-slice	Specifies how to slice the border image
border-image-source	Specifies the path to the image to be used as a border
border-image-width	Specifies the width of the border image
border-left	A shorthand property for all the *border-left-** properties
border-left-color	Sets the color of the left border
border-left-style	Sets the style of the left border
border-left-width	Sets the width of the left border
border-radius	A shorthand property for the four *border-*-radius* properties
border-right	A shorthand property for all the *border-right-** properties
border-right-color	Sets the color of the right border
border-right-style	Sets the style of the right border
border-right-width	Sets the width of the right border
border-spacing	Sets the distance between the borders of adjacent cells
border-style	Sets the style of the four borders
border-top	A shorthand property for *border-top-width, border-top-style* and *border-top-color*
border-top-color	Sets the color of the top border
border-top-left-radius	Defines the radius of the border of the top-left corner
border-top-right-radius	Defines the radius of the border of the top-right corner
border-top-style	Sets the style of the top border
border-top-width	Sets the width of the top border
border-width	Sets the width of the four borders
bottom	Sets the elements position, from the bottom of its parent element
box-decoration-break	Sets the behavior of the background and border of an element at page-break, or, for in-line elements, at line-break.

box-shadow	Attaches one or more shadows to an element
box-sizing	Defines how the width and height of an element are calculated: should they include padding and borders, or not
break-after	Specifies whether or not a page-, column-, or region-break should occur after the specified element
break-before	Specifies whether or not a page-, column-, or region-break should occur before the specified element
break-inside	Specifies whether or not a page-, column-, or region-break should occur inside the specified element

caption-side	Specifies the placement of a table caption
caret-color	Specifies the color of the cursor (caret) in inputs, textareas, or any element that is editable
@charset	Specifies the character encoding used in the style sheet
clear	Specifies on which sides of an element floating elements are not allowed to float
clip	Clips an absolutely positioned element
color	Sets the color of text
column-count	Specifies the number of columns an element should be divided into
column-fill	Specifies how to fill columns, balanced or not
column-gap	Specifies the gap between the columns
column-rule	A shorthand property for all the *column-rule-** properties
column-rule-color	Specifies the color of the rule between columns
column-rule-style	Specifies the style of the rule between columns
column-rule-width	Specifies the width of the rule between columns
column-span	Specifies how many columns an element should span across
column-width	Specifies the column width
columns	A shorthand property for *column-width* and *column-count*
content	Used with the :before and :after pseudo-elements, to insert generated content
counter-increment	Increases or decreases the value of one or more CSS counters
counter-reset	Creates or resets one or more CSS counters
cursor	Specifies the mouse cursor to be displayed when pointing over an element

direction	Specifies the text direction/writing direction

display	Specifies how a certain HTML element should be displayed

empty-cells	Specifies whether or not to display borders and background on empty cells in a table

filter	Defines effects (e.g. blurring or color shifting) on an element before the element is displayed
flex	A shorthand property for the *flex-grow, flex-shrink*, and the *flex-basis* properties
flex-basis	Specifies the initial length of a flexible item
flex-direction	Specifies the direction of the flexible items
flex-flow	A shorthand property for the *flex-direction* and the *flex-wrap* properties
flex-grow	Specifies how much the item will grow relative to the rest
flex-shrink	Specifies how the item will shrink relative to the rest
flex-wrap	Specifies whether the flexible items should wrap or not
float	Specifies whether or not a box should float
font	A shorthand property for the *font-style, font-variant, font-weight, font-size/line-height*, and the *font-family* properties
@font-face	A rule that allows websites to download and use fonts other than the "web-safe" fonts
font-family	Specifies the font family for text
font-feature-settings	Allows control over advanced typographic features in OpenType fonts
@font-feature-values	Allows authors to use a common name in font-variant-alternate for feature activated differently in OpenType
font-kerning	Controls the usage of the kerning information (how letters are spaced)
font-language-override	Controls the usage of language-specific glyphs in a typeface
font-size	Specifies the font size of text
font-size-adjust	Preserves the readability of text when font fallback occurs
font-stretch	Selects a normal, condensed, or expanded face from a font family
font-style	Specifies the font style for text
font-synthesis	Controls which missing typefaces (bold or italic) may be synthesized by the browser
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-variant-alternates	Controls the usage of alternate glyphs associated to alternative names defined in @font-feature-values
font-variant-caps	Controls the usage of alternate glyphs for capital letters

font-variant-east-asian	Controls the usage of alternate glyphs for East Asian scripts (e.g Japanese and Chinese)
font-variant-ligatures	Controls which ligatures and contextual forms are used in textual content of the elements it applies to
font-variant-numeric	Controls the usage of alternate glyphs for numbers, fractions, and ordinal markers
font-variant-position	Controls the usage of alternate glyphs of smaller size positioned as superscript or subscript regarding the baseline of the font
font-weight	Specifies the weight of a font
grid	A shorthand property for the *grid-template-rows, grid-template-columns, grid-template-areas, grid-auto-rows, grid-auto-columns*, and the *grid-auto-flow* properties
grid-area	Either specifies a name for the grid item, or this property is a shorthand property for the *grid-row-start*, *grid-column-start*, *grid-row-end*, and *grid-column-end* properties
grid-auto-columns	Specifies a default column size
grid-auto-flow	Specifies how auto-placed items are inserted in the grid
grid-auto-rows	Specifies a default row size
grid-column	A shorthand property for the *grid-column-start* and the *grid-column-end* properties
grid-column-end	Specifies where to end the grid item
grid-column-gap	Specifies the size of the gap between columns
grid-column-start	Specifies where to start the grid item
grid-gap	A shorthand property for the *grid-row-gap* and *grid-column-gap* properties
grid-row	A shorthand property for the *grid-row-start* and the *grid-row-end* properties
grid-row-end	Specifies where to end the grid item
grid-row-gap	Specifies the size of the gap between rows
grid-row-start	Specifies where to start the grid item
grid-template	A shorthand property for the *grid-template-rows*, *grid-template-columns* and *grid-areas* properties
grid-template-areas	Specifies how to display columns and rows, using named grid items
grid-template-columns	Specifies the size of the columns, and how many columns in a grid layout
grid-template-rows	Specifies the size of the rows in a grid layout
hanging-punctuation	Specifies whether a punctuation character may be placed outside the line box

height	Sets the height of an element
hyphens	Sets how to split words to improve the layout of paragraphs

image-rendering	Gives a hint to the browser about what aspects of an image are most important to preserve when the image is scaled
@import	Allows you to import a style sheet into another style sheet
isolation	Defines whether an element must create a new stacking content

justify-content	Specifies the alignment between the items inside a flexible container when the items do not use all available space

@keyframes	Specifies the animation code

left	Specifies the left position of a positioned element
letter-spacing	Increases or decreases the space between characters in a text
line-break	Specifies how/if to break lines
line-height	Sets the line height
list-style	Sets all the properties for a list in one declaration
list-style-image	Specifies an image as the list-item marker
list-style-position	Specifies the position of the list-item markers (bullet points)
list-style-type	Specifies the type of list-item marker

margin	Sets all the margin properties in one declaration
margin-bottom	Sets the bottom margin of an element
margin-left	Sets the left margin of an element
margin-right	Sets the right margin of an element
margin-top	Sets the top margin of an element
max-height	Sets the maximum height of an element
max-width	Sets the maximum width of an element

@media Sets the style rules for different media types/devices/sizes

min-height Sets the minimum height of an element

min-width Sets the minimum width of an element

mix-blend-mode Specifies how an element's content should blend with its direct parent background

object-fit Specifies how the contents of a replaced element should be fitted to the box established by its used height and width

object-position Specifies the alignment of the replaced element inside its box

opacity Sets the opacity level for an element

order Sets the order of the flexible item, relative to the rest

orphans Sets the minimum number of lines that must be left at the bottom of a page when a page break occurs inside an element

outline A shorthand property for the *outline-width, outline-style*, and the *outline-color* properties

outline-color Sets the color of an outline

outline-offset Offsets an outline, and draws it beyond the border edge

outline-style Sets the style of an outline

outline-width Sets the width of an outline

overflow Specifies what happens if content overflows an element's box

overflow-wrap Specifies whether or not the browser may break lines within words in order to prevent overflow (when a string is too long to fit its containing box)

overflow-x Specifies whether or not to clip the left/right edges of the content, if it overflows the element's content area

overflow-y Specifies whether or not to clip the top/bottom edges of the content, if it overflows the element's content area

padding A shorthand property for all the *padding-** properties

padding-bottom Sets the bottom padding of an element

padding-left Sets the left padding of an element

padding-right Sets the right padding of an element

padding-top	Sets the top padding of an element
page-break-after	Sets the page-break behavior after an element
page-break-before	Sets the page-break behavior before an element
page-break-inside	Sets the page-break behavior inside an element
perspective	Gives a 3D-positioned element some perspective
perspective-origin	Defines at which position the user is looking at the 3D-positioned element
pointer-events	Defines whether or not an element reacts to pointer events
position	Specifies the type of positioning method used for an element (static, relative, absolute or fixed)

quotes	Sets the type of quotation marks for embedded quotations

resize	Defines if (and how) an element is resizable by the user
right	Specifies the right position of a positioned element

scroll-behavior	Specifies whether to smoothly animate the scroll position in a scrollable box, instead of a straight jump

tab-size	Specifies the width of a tab character
table-layout	Defines the algorithm used to lay out table cells, rows, and columns
text-align	Specifies the horizontal alignment of text
text-align-last	Describes how the last line of a block or a line right before a forced line break is aligned when text-align is "justify"
text-combine-upright	Specifies the combination of multiple characters into the space of a single character
text-decoration	Specifies the decoration added to text
text-decoration-color	Specifies the color of the text-decoration
text-decoration-line	Specifies the type of line in a text-decoration
text-decoration-style	Specifies the style of the line in a text decoration

text-indent	Specifies the indentation of the first line in a text-block
text-justify	Specifies the justification method used when text-align is "justify"
text-orientation	Defines the orientation of the text in a line
text-overflow	Specifies what should happen when text overflows the containing element
text-shadow	Adds shadow to text
text-transform	Controls the capitalization of text
text-underline-position	Specifies the position of the underline which is set using the text-decoration property
top	Specifies the top position of a positioned element
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space
transition	A shorthand property for all the *transition-** properties
transition-delay	Specifies when the transition effect will start
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
transition-property	Specifies the name of the CSS property the transition effect is for
transition-timing-function	Specifies the speed curve of the transition effect

unicode-bidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
user-select	Specifies whether the text of an element can be selected

vertical-align	Sets the vertical alignment of an element
visibility	Specifies whether or not an element is visible

white-space	Specifies how white-space inside an element is handled
widows	Sets the minimum number of lines that must be left at the top of a page when a page break occurs inside an element

width	Sets the width of an element
word-break	Specifies how words should break when reaching the end of a line
word-spacing	Increases or decreases the space between words in a text
word-wrap	Allows long, unbreakable words to be broken and wrap to the next line
writing-mode	Specifies whether lines of text are laid out horizontally or vertically

z-index	Sets the stack order of a positioned element

HTML Tables

Code:

```
<table border="4" cellpadding="2" cellspacing="2" width="50%">
<tr>
<th> Language </th>
<th> Inventor </th>
<th> Year </th>
</tr>
<tr>
<td> C Language </td>
<td> Dennis Ritchie </td>
<td> 1967-1973 </td>
</tr>
<tr>
<td> Java </td>
<td> James Gosling </td>
<td> 2010 </td>
</tr>
<tr>
<td> Pascal </td>
<td> Niklaus Wirth </td>
<td> 1970 </td>
</tr>
```

```
</table>
```

Code:

```
<table border="4" cellpadding="2" cellspacing="2" bordercolor="#008000" width="50%">
<tr>
<th> Language </th>
<th> Inventor </th>
<th> Year </th>
</tr>
<tr>
<td> C Language </td>
<td> Dennis Ritchie </td>
<td> 1967-1973 </td>
</tr>
<tr>
<td> Java </td>
<td> James Gosling </td>
<td> 2010 </td>
</tr>
<tr>
<td> Pascal </td>
<td> Niklaus Wirth </td>
<td> 1970 </td>
</tr>
</table>
```

Code:

```
<table border="4" cellpadding="2" cellspacing="2" width="50%">
<tr>
<th bgcolor="#cccccc"> Language </th>
<th bgcolor="#cccccc"> Inventor </th>
<th bgcolor="#cccccc"> Year </th>
</tr>
<tr>
<td> C Language </td>
```

```
<td> Dennis Ritchie </td>
<td> 1967-1973 </td>
</tr>
<tr>
<td> Java </td>
<td> James Gosling </td>
<td> 2010 </td>
</tr>
<tr>
<td> Pascal </td>
<td> Niklaus Wirth </td>
<td> 1970 </td>
</tr>
</table>
```

Code:

```
<table border="4" cellpadding="2" cellspacing="2" width="50%" align="right"
bgcolor="#CCCCCC">
<tr>
<th> Language </th>
<th> Inventor </th>
<th> Year </th>
</tr>
<tr>
<td> C Language </td>
<td> Dennis Ritchie </td>
<td> 1967-1973 </td>
</tr>
<tr>
<td> Java </td>
<td> James Gosling </td>
<td> 2010 </td>
</tr>
<tr>
<td> Pascal </td>
<td> Niklaus Wirth </td>
<td> 1970 </td>
</tr>
</table>
```

Code:

```
<!DOCTYPE html>
<html>
<head>

<style>
thead {color: orange;}
tbody {color:blue;}
tfoot {color:red;}
table, th, td {
border: 1px solid black;
}
</style>
</head>

<body>
<table>
<thead>
<tr>
<th>Language </th>
<th>Inventor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>James Gosling</td>
</tr>
<tr>
<td>Pascal</td>
<td>Niklaus Wirth</td>
</tr>
</tbody>
<tfoot>
<tr>
<td>Python</td>
<td>Guido van Rossum</td>
</tr>
</tfoot>
```

```
</table>
</body>
</html>
```

HTML Marquees

Code:

```
<!DOCTYPE html>
<html>
<body>
<marquee behavior="scroll" direction="right"> HTML </marquee>
</body>
</html>
```

Code:

```
<!DOCTYPE html>
<html>
<body>
<marquee behavior="scroll" direction="left"> HTML </marquee>
</body>
</html>
```

Code:

```
<!DOCTYPE html>
<html>
<body>
<marquee behavior="scroll" direction="up"> HTML </marquee>
</body>
</html>
```

Code:

```
<!DOCTYPE html>
<html>
<body>
<marquee behavior="scroll" direction="down"> HTML </marquee>
```

```
<marquee behavior="alternate"> HTML </marquee>
<marquee bgcolor="#cccccc" loop="-1" scrollamount="2" width="100%"> HTML </marquee>
<marquee behavior="scroll" direction="left" scrollamount="30"> HTML </marquee>
</body>
</html>
```

Code:

```
<!DOCTYPE html>
<html>
<body>
<marquee behavior="scroll" direction="up" style="height: 200px;"> HTML </marquee>
<marquee behavior="scroll" direction="right" scrollamount="1"> HTML </marquee>
<marquee behavior="scroll" direction="right" scrollamount="10"> HTML </marquee>
<marquee behavior="scroll" direction="right" scrollamount="20"> HTML </marquee>
</body>
</html>
```

HTML Images, Video & Audio

Code:

```
<img src="images/pic.jpg" style="width:323x;height:100px; margin-top: 4px;">
```

```
<video width="300" height="300" controls>
<source src="images/video.mp4" type="video/mp4">
<source src="images/video.ogg" type="video/ogg">
<source src="images/video.webm" type="video/webm">
</video>
```

```
<video width="300" height="300" controls muted>
```

<source src="images/video.mp4" type="video/mp4"> <source src="images/video.ogg" type="video/ogg"> <source src="images/video.webm" type="video/webm"> **</video>**
<marquee> <img src="images/pic.jpg" style="width:323x;height:100px; margin-top: 4px;"> **</marquee>**
<body style="background-color:powderblue;" > **<center>** **<h1>** HTML **</h1>** **<p>** Hypertext Markup Language **</p>** **</center>** </body>
<body background="images/pic3.jpg" > **<center>** **<h1>** HTML **</h1>** **<p>** Hypertext Markup Language **</p>** **</center>** </body>
<audio src="images/music.ogg" controls >
<embed src="images/music.ogg" autostart="true" loop="true" hidden="true" >
<a href="images/video.ogg"> Play the Video **</a>**
<a href="images/music.ogg"> Play the Audio **</a>**
<style>

```
body {

background-image: url("images/pic3.jpg");

}

</style>

<center>

<h1> HTML </h1>

<p> Hypertext Markup Language </p>

</center>
```

```
<style>

body {

background-image: url("images/pic3.jpg");

background-position: 50% 50%;

background-repeat: repeat;

}

</style>

<center>

<h1> HTML </h1>

<p> Hypertext Markup Language </p>

</center>
```

```
<style>

body {

background-image: url("images/pic3.jpg");

padding:5px;

width:150px;

height:200px;

border:1px solid black;

background-repeat:no-repeat;

}

</style>
```

```
<center>
<h1> HTML </h1>
<p> Hypertext Markup Language </p>
</center>
```

```
<img src="images/pic.jpg" style="width:323x;height:100px; margin-top:
4px;"align="right">
```

```
<img src="images/pic.jpg" style="width:323x;height:100px; margin-top: 4px; border:5px
solid #337AB7;">
```

```
<marquee width="50%"> <img src="images/pic.jpg" style="width:323x;height:100px;
margin-top: 4px;"> </marquee>
```

```
<body style="background-color:yellowgreen;color:white;">
...HTML...
</body>
```

```
<div style="background-image:url('images/bg1.jpg');
background-repeat:repeat; background-attachment:fixed; overflow:scroll; width:200px;
height:200px;">
<p>HyperText Markup Language (HTML) is the standard markup language for creating web
pages and web applications.</p>
</div>
```

```
<div style="background-image:url('images/bg1.jpg'); background-repeat:repeat;
height:500px;width:500px;"> <p>HTML</p> </div>
```

```
<div style="background-color:white;background-image:url(images/bg1.jpg);background-
repeat:no-repeat;background-position:bottom right;border:1px solid
black;width:300px;height:300px;font-size:18px;">HTML </div>
```

```
<div style="background-color:white;background-image:url(images/bg1.jpg);background-
repeat:no-repeat;background-position:center;border:1px solid
black;width:300px;height:300px;font-size:18px;">HTML </div>
```

```
<a href="http://www.youtube.com/"> <img src="images/pic1.jpg"
style="width:50x;height:50px; margin-top: 4px; "> </a>
```

```
<div style="background:url(images/bg1.jpg) white center repeat-y scroll;border:1px
solid black;width:300px;height:300px;font-size:18px;">HTML</div>
```

```
<body background="images/transparent_image.png" bgcolor="#ADD8E6">
<center>
<h1> HTML </h1>
<p> Hypertext Markup Language </p>
</center>
</body>
```

```
<audio src="images/song.ogg" controls="controls" preload="auto"> </audio>
```

```
<video width="300" height="300" autoplay loop muted="muted">
<source src="images/video.mp4" type="video/mp4">
<source src="images/video.ogg" type="video/ogg">
<source src="images/video.webm" type="video/webm">
</video>
```

HTML Links & iFrames

Code:

```
<a href="http://www.google.com/"> Web Browser </a>
```

```
<a href="http://www.youtube.com/"> <img src="images/pic1.jpg"
style="width:50x;height:50px; margin-top: 4px; "> </a>
```

```
<button type="button" style="font-size:12pt;color:white;background-
color:orange;border:2px solid orange;padding:3px"
onclick="alert('HTML')">Click Me</button>
```

```
<a href="http://www.youtube.com/"> <button type="button" style="font-
size:12pt;color:white;background-color:green;border:2px solid
orange;padding:3px" > youtube</button></a>
```

```
<a href="http://www.youtube.com/"> <video width="200" controls>
<source src="images/video.mp4" type="video/mp4">
<source src="images/video.ogg" type="video/ogg">
<source src="images/video.webm" type="video/webm">
</video></a>
```

```
<form action="http://google.com">
<input type="submit" value="Go to Google"/>
</form>
```

```
<a href="http://www.google.com/">Google </a> | <a
href="http://www.wikipedia.org/">Wikipedia </a> | <a
```

href="http://www.youtube.com/">Youtube </a>
<iframe src="page1.html" width="200" height="200"> **</iframe>**
<iframe src="page1.html" style="border:none;" width="200" height="200">**</iframe>**
<select onchange="this.options[this.selectedIndex].value && (window.location = this.options[this.selectedIndex].value);" > <option value="">Select...</option> <option value="http://www.google.com">Google</option> <option value="http://www.yahoo.com">Yahoo</option> **</select>**
<iframe src="page1.html" style="border:2px solid grey;" width="200" height="200">**</iframe>**
<div style="width:150px;height:150px;line-height:3em;overflow:scroll;padding:5px;background-color:#FCFADD;color:#714D03;border:4px double #DEBB07;"> HTML </div>
<div style="width:150px;height:150px;line-height:3em;overflow:scroll;padding:5px;background-color:#FCFADD;color:#714D03;scrollbar-base-color:#DEBB07;"> HTML </div>
<div style="width:150px;height:150px;line-height:3em;overflow:auto;padding:5px;"> HyperText Markup Language (HTML) is the standard markup language for creating web pages and web applications.

```
With Cascading Style Sheets (CSS), and JavaScript, it forms a triad of
cornerstone technologies for the World Wide Web.[1] Web browsers receive HTML
documents from a webserver or from local storage and render them into
multimedia web pages. HTML describes the structure of a web page semantically
and originally included cues for the appearance of the document. </div>
```

```
<form name="blah_blah">
<select name="ddmenu_name" id="ddmenu_name" style="width: 80% !important;">
<option value="" selected>Select Site</option>
<option value="http://www.yahoo.com">Yahoo!!!</option>
<option value="http://www.gmail.com">Gmail</option>
<option value="http://www.google.co.in">Google</option>
<option value="http://www.facebook.com">Facebook</option>
</select>
<input type="button" name="Submit" value="Go!"
onClick="window.open(ddmenu_name.value,'newtab'+ddmenu_name.value)">
</form>
```

```
<center> <iframe name="cwindow" style="width:100x;height:100px; border:16px
double purple; margin-top: 5px; " src="page1.html"></iframe> </center>
```

```
<a style="color:blue;" href="http://www.google.com/"> Web Browser </a>
```

```
<a style="color:blue;font-size:12pt;font-style:italic;"
href="http://www.google.com/"> Web Browser </a>
```

```
<iframe src="page1.html" width="100%" height="100%" frameborder="0"
scrolling="no" allowtransparency="true"></iframe>
```

HTML Fonts & Styles

Code:

```
<div style="width:200px;height:100px;padding:10px;border:10px solid yellowgreen;"> Hyper TEXT MARKUP language</div>

<div style="width:200px;height:100px;padding:10px;border:1px dotted black;"> Hyper TEXT MARKUP language</div>

<div style="width:200px;height:100px;padding:10px;border:1px dashed black;"> Hyper TEXT MARKUP language</div>

<div style="width:200px;height:100px;padding:10px;border:10px double yellowgreen;"> Hyper TEXT MARKUP language</div>

<div style="width:200px;height:100px;padding:10px;border-width:6px;border-color:yellowgreen;border-style:dotted dashed solid double;"> Hyper TEXT MARKUP language</div>

<div style="width:200px;height:100px;padding:10px;border:10px groove yellowgreen;"> Hyper TEXT MARKUP language</div>

<p style="font-size: 12pt; color: #008000"> Hyper TEXT MARKUP language</p>

<span style="font-size: 10pt; color: #228B22"> Hyper TEXT MARKUP language</span>

<div style="background-color:black;color:white;padding:20px;">
```

```
<h1>HTML</h1>

<p> HyperText Markup Language (HTML) is the standard markup language for
creating web pages and web applications. With Cascading Style Sheets (CSS),
and JavaScript, it forms a triad of cornerstone technologies for the World
Wide Web. </p>

</div>
```

```
<p style="color:##B22222">Color text and <span
style="color:limegreen;">another color</span>, and now back to the same. Oh,
and here's a <span style="background-color:PaleGreen;">different background
color</span> just in case you need it!</p>
```

```
<pre style="color:black">Preformatted text displays just
as you
type it...
...line breaks,
spaces...
...and all!
</pre>
```

```
<code style="color:green">Code text.</code>
```

```
<style>
.different-font-color { color: orange; }
.different-background-color { background-color: limegreen }
</style>
<p>Normal font color <span class="different-font-color">different font
color</span> normal font color
```

<span class="different-background-color">different background color</span></p>
<div style="background-color:GreenYellow;width:200px;border:1px solid black;padding:15px;"> <p>HTML background code is actually CSS!</p> </div>
<h3 style="color:orange;">HTML </h3>
<h3 style="background-color:yellow;">HTML </h3>
<h3 style="border:1px dashed orange;">HTML</h3>
<div style="width:300px;background-color:yellow;color:blue;border:1px solid black;"> <p>HyperText Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS), and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web.</p> <p style="color:white;background-color:orange;">Web browsers receive HTML documents from a webserver or from local storage and render them into multimedia web pages. HTML describes <span style="color:black;">the</span> structure of a web page semantically and originally included cues for the appearance of the <span style="background-color:red;color:lime;font-weight:bold;">document.</span>.</p> **</div>**
<div style="background-color:black;padding:5%;"> <div style="background-color:orange;border:3px solid white;color:white;width:60%;padding:2%;"> HTML </div> </div>
<div align="center"><p> <font face="arial, helvetica" size="-2">Vist <a href="http://www.google.com/"> Google</a></font></p> </div>

Timeline of quantum computing

1960s

- 1960
  - Stephen Wiesner invents conjugate coding.

1970s

- 1970
  - James Park articulates the no-cloning theorem

- 1973
  - Alexander Holevo publishes a paper showing that *n* qubits can carry more than *n* classical bits of information, but at most *n* classical bits are accessible (a result known as "Holevo's theorem" or "Holevo's bound").
  - Charles H. Bennett shows that computation can be done reversibly.
- 1975
  - R. P. Poplavskii publishes "Thermodynamical models of information processing" (in Russian) which showed the computational infeasibility of simulating quantum systems on classical computers, due to the superposition principle.
- 1976
  - Polish mathematical physicist Roman Stanisław Ingarden publishes a seminal paper entitled "Quantum Information Theory" in Reports on Mathematical Physics, vol. 10, 43–72, 1976. (The paper was submitted in 1975.) It is one of the first attempts at creating a quantum information theory, showing that Shannon information theory cannot directly be generalized to the quantum case, but rather that it is possible to construct a quantum information theory, which is a generalization of Shannon's theory, within the formalism of a generalized quantum mechanics of open systems and a generalized concept of observables (the so-called semi-observables).

1980s

- 1980
  - Paul Benioff describes the first quantum mechanical model of a computer. In this work, Benioff showed that a computer could operate under the laws of quantum mechanics by describing a Schrödinger equation description of Turing machines, laying a foundation for further work in quantum computing. The paper was submitted in June 1979 and published in April 1980.
  - Yuri Manin briefly motivates the idea of quantum computing

    - Tommaso Toffoli introduces the reversible Toffoli gate, which, together with the NOT and XOR gates provides a universal set for reversible classical computation.
- 1981
    - At the First Conference on the Physics of Computation, held at MIT in May, Paul Benioff and Richard Feynman give talks on quantum computing. Benioff's built on his earlier 1980 work showing that a computer can operate under the laws of quantum mechanics. The talk was titled "Quantum mechanical Hamiltonian models of discrete processes that erase their own histories: application to Turing machines". In Feynman's talk, he observed that it appeared to be impossible to efficiently simulate an evolution of a quantum system on a classical computer, and he proposed a basic model for a quantum computer.
- 1982
    - Paul Benioff further develops his original model of a quantum mechanical Turing machine.
    - William Wootters and Wojciech Zurek, and independently Dennis Dieks rediscover the no-cloning theorem.
- 1984
    - Charles Bennett and Gilles Brassard employ Wiesner's conjugate coding for distribution of cryptographic keys.
- 1985
    - David Deutsch, at the University of Oxford, describes the first universal quantum computer. Just as a Universal Turing machine can simulate any other Turing machine efficiently (Church-Turing thesis), so the universal quantum computer is able to simulate any other quantum computer with at most a polynomial slowdown.
- 1988
    - Yoshihisa Yamamoto (scientist) and K. Igeta propose the first physical realization of a quantum computer, including Feynman's CNOT gate. Their approach uses atoms and photons and is the progenitor of modern quantum computing and networking protocols using photons to transmit qubits and atoms to perform two-qubit operations.
    - Gerard J. Milburn proposes a quantum-optical realization of a Fredkin gate.
- 1989
    - Bikas K. Chakrabarti & collaborators from Saha Institute of Nuclear Physics, Kolkata, propose the idea that quantum fluctuations could help explore rough energy landscapes by escaping from local minima of glassy systems having tall but thin barriers by tunneling (instead of climbing over using thermal excitations), suggesting the effectiveness of quantum annealing over classical simulated annealing.

1990s

- 1991
  - Artur Ekert at the University of Oxford, expands on the original proposal by David Deutsch , for entanglement-based secure communication.
- 1992
  - David Deutsch and Richard Jozsa propose a computational problem that can be solved efficiently with the determinist Deutsch–Jozsa algorithm on a quantum computer, but for which no deterministic classical algorithm is possible. This was perhaps the earliest result in the computational complexity of quantum computers, proving that they were capable of performing *some* well-defined computational task more efficiently than any classical computer.
- 1993
  - Dan Simon, at Université de Montréal, invents an oracle problem for which a quantum computer would be exponentially faster than a conventional computer. This algorithm introduces the main ideas which were then developed in Peter Shor's factorization algorithm.
- 1994
  - Peter Shor, at AT&T's Bell Labs in New Jersey, discovers an important algorithm. It allows a quantum computer to factor large integers quickly. It solves both the factoring problem and the discrete log problem. Shor's algorithm can theoretically break many of the cryptosystems in use today. Its invention sparked a tremendous interest in quantum computers.
  - First United States Government workshop on quantum computing is organized by NIST in Gaithersburg, Maryland, in autumn
  - Isaac Chuang and Yoshihisa Yamamoto (scientist) propose a quantum-optical realization of a quantum computer to implement Deutsch's algorithm. Their work introduces dual-rail encoding for photonic qubits.
  - In December, Ignacio Cirac, at University of Castilla-La Mancha at Ciudad Real, and Peter Zoller at the University of Innsbruck propose an experimental realization of the controlled-NOT gate with cold trapped ions.
- 1995
  - The first United States Department of Defense workshop on quantum computing and quantum cryptography is organized by United States Army physicists Charles M. Bowden, Jonathan P. Dowling, and Henry O. Everitt; it takes place in February at the University of Arizona in Tucson.
  - Peter Shor proposes the first schemes for quantum error correction.
  - Christopher Monroe and David Wineland at NIST (Boulder, Colorado) experimentally realize the first quantum logic gate – the controlled-NOT gate – with trapped ions, following the Cirac-Zoller proposal.
- 1996

  - Lov Grover, at Bell Labs, invents the quantum database search algorithm. The quadratic speedup is not as dramatic as the speedup for factoring, discrete logs, or physics simulations. However, the algorithm can be applied to a much wider variety of problems. Any problem that has to be solved by random, brute-force search, can take advantage of this quadratic speedup (in the number of search queries).
  - The United States Government, particularly in a joint partnership of the Army Research Office (now part of the Army Research Laboratory) and the National Security Agency, issues the first public call for research proposals in quantum information processing.
  - Andrew Steane designs Steane codes for error correction.
  - David P. DiVincenzo, from IBM, proposes a list of minimal requirements for creating a quantum computer.
- 1997
  - David Cory, Amr Fahmy and Timothy Havel, and at the same time Neil Gershenfeld and Isaac L. Chuang at MIT publish the first papers realizing gates for quantum computers based on bulk nuclear spin resonance, or thermal ensembles. The technology is based on a nuclear magnetic resonance (NMR) machine, which is similar to the medical magnetic resonance imaging machine.
  - Alexei Kitaev describes the principles of topological quantum computation as a method for combating decoherence.
  - Daniel Loss and David P. DiVincenzo propose the Loss-DiVincenzo quantum computer, using as qubits the intrinsic spin-1/2 degree of freedom of individual electrons confined to quantum dots.
- 1998
  - First experimental demonstration of a quantum algorithm. A working 2-qubit NMR quantum computer is used to solve Deutsch's problem by Jonathan A. Jones and Michele Mosca at Oxford University and shortly after by Isaac L. Chuang at IBM's Almaden Research Center and Mark Kubinec and the University of California, Berkeley together with coworkers at Stanford University and MIT.
  - First working 3-qubit NMR computer.
  - Bruce Kane proposes a silicon based nuclear spin quantum computer, using nuclear spins of individual phosphorus atoms in silicon as the qubits and donor electrons to mediate the coupling between qubits.
  - First execution of Grover's algorithm on an NMR computer.
  - Hidetoshi Nishimori & colleagues from Tokyo Institute of Technology showed that quantum annealing algorithm can perform better than classical simulated annealing.
  - Daniel Gottesman and Emanuel Knill independently prove that a certain subclass of quantum computations can be efficiently emulated with classical resources (Gottesman–Knill theorem).
- 1999

  - Samuel L. Braunstein and collaborators show that none of the bulk NMR experiments performed to date contained any entanglement, the quantum states being too strongly mixed. This is seen as evidence that NMR computers would likely not yield a benefit over classical computers. It remains an open question, however, whether entanglement is necessary for quantum computational speedup.
  - Gabriel Aeppli, Thomas Felix Rosenbaum and colleagues demonstrate experimentally the basic concepts of quantum annealing in a condensed matter system.
  - Yasunobu Nakamura and Jaw-Shen Tsai demonstrate that a superconducting circuit can be used as a qubit. This leads to a global effort to develop quantum computers using superconducting circuits, culminating in Google's demonstration of quantum supremacy using this technology in 2019.

2000s

- 2000
  - Arun K. Pati and Samuel L. Braunstein proved the quantum no-deleting theorem. This is dual to the no-cloning theorem which shows that one cannot delete a copy of an unknown qubit. Together with the stronger no-cloning theorem, the no-deleting theorem has important implication, i.e., quantum information can neither be created nor be destroyed.
  - First working 5-qubit NMR computer demonstrated at the Technical University of Munich.
  - First execution of order finding (part of Shor's algorithm) at IBM's Almaden Research Center and Stanford University.
  - First working 7-qubit NMR computer demonstrated at the Los Alamos National Laboratory.
  - The standard textbook, Quantum Computation and Quantum Information, by Michael Nielsen and Isaac Chuang is published.
- 2001
  - First execution of Shor's algorithm at IBM's Almaden Research Center and Stanford University. The number 15 was factored using 10^{18} identical molecules, each containing seven active nuclear spins.
  - Noah Linden and Sandu Popescu proved that the presence of entanglement is a necessary condition for a large class of quantum protocols. This, coupled with Braunstein's result (see 1999 above), called the validity of NMR quantum computation into question.
  - Emanuel Knill, Raymond Laflamme, and Gerard Milburn show that optical quantum computing is possible with single photon sources, linear optical elements, and single photon detectors, launching the field of linear optical quantum computing.
  - Robert Raussendorf and Hans Jürgen Briegel propose measurement-based quantum computation.
- 2002

    - The Quantum Information Science and Technology Roadmapping Project, involving some of the main participants in the field, laid out the Quantum computation roadmap.
    - The Institute for Quantum Computing was established at the University of Waterloo in Waterloo, Ontario by Mike Lazaridis, Raymond Laflamme and Michele Mosca.
- 2003
    - Implementation of the Deutsch–Jozsa algorithm on an ion-trap quantum computer at the University of Innsbruck
    - Todd D. Pittman and collaborators at Johns Hopkins University, Applied Physics Laboratory and independently Jeremy L. O'Brien and collaborators at the University of Queensland, demonstrate quantum controlled-not gates using only linear optical elements.
    - First implementation of a CNOT quantum gate according to the Cirac–Zoller proposal by a group at the University of Innsbruck led by Rainer Blatt.
    - DARPA Quantum Network becomes fully operational on October 23, 2003.
    - The Institute for Quantum Optics and Quantum Information (IQOQI) was established in Innsbruck and Vienna, Austria, by the founding directors Rainer Blatt, Hans Jürgen Briegel, Rudolf Grimm, Anton Zeilinger and Peter Zoller.
- 2004
    - First working pure state NMR quantum computer (based on parahydrogen) demonstrated at Oxford University and University of York.
    - Physicists at the University of Innsbruck show deterministic quantum-state teleportation between a pair of trapped calcium ions.
    - First five-photon entanglement demonstrated by Jian-Wei Pan's group at the University of Science and Technology of China, the minimal number of qubits required for universal quantum error correction.

2005

- University of Illinois at Urbana–Champaign scientists demonstrate quantum entanglement of multiple characteristics, potentially allowing multiple qubits per particle.
- Two teams of physicists measured the capacitance of a Josephson junction for the first time. The methods could be used to measure the state of quantum bits in a quantum computer without disturbing the state.
- In December, the first quantum byte, or *qubyte*, is announced to have been created by scientists at the Institute for Quantum Optics and Quantum Information and the University of Innsbruck in Austria.
- Harvard University and Georgia Institute of Technology researchers succeeded in transferring quantum information between "quantum memories" – from atoms to photons and back again.

2006

- Materials Science Department of Oxford University, cage a qubit in a "buckyball" (a molecule of buckminsterfullerene), and demonstrated quantum "bang-bang" error correction.

- Researchers from the University of Illinois at Urbana–Champaign use the Zeno Effect, repeatedly measuring the properties of a photon to gradually change it without actually allowing the photon to reach the program, to search a database without actually "running" the quantum computer.
- Vlatko Vedral of the University of Leeds and colleagues at the universities of Porto and Vienna found that the photons in ordinary laser light can be quantum mechanically entangled with the vibrations of a macroscopic mirror.
- Samuel L. Braunstein at the University of York along with the University of Tokyo and the Japan Science and Technology Agency gave the first experimental demonstration of quantum telecloning.
- Professors at the University of Sheffield develop a means to efficiently produce and manipulate individual photons at high efficiency at room temperature.
- New error checking method theorized for Josephson junction computers.
- First 12 qubit quantum computer benchmarked by researchers at the Institute for Quantum Computing and the Perimeter Institute for Theoretical Physics in Waterloo, as well as MIT, Cambridge.
- Two dimensional ion trap developed for quantum computing.
- Seven atoms placed in stable line, a step on the way to constructing a quantum gate, at the University of Bonn.
- A team at Delft University of Technology in the Netherlands created a device that can manipulate the "up" or "down" spin-states of electrons on quantum dots.
- University of Arkansas develops quantum dot molecules.
- Spinning new theory on particle spin brings science closer to quantum computing.
- University of Copenhagen develops quantum teleportation between photons and atoms.
- University of Camerino scientists develop theory of macroscopic object entanglement, which has implications for the development of quantum repeaters.
- Tai-Chang Chiang, at Illinois at Urbana–Champaign, finds that quantum coherence can be maintained in mixed-material systems.
- Cristophe Boehme, University of Utah, demonstrates the feasibility of reading spin-data on a silicon-phosphorus quantum computer.

2007

- Subwavelength waveguide developed for light.
- Single photon emitter for optical fibers developed.
- Six-photon one-way quantum computer is created in lab.
- New material proposed for quantum computing.

- Single atom single photon server devised.
- First use of Deutsch's Algorithm in a cluster state quantum computer.
- University of Cambridge develops electron quantum pump.
- Superior method of qubit coupling developed.
- Successful demonstration of controllably coupled qubits.
- Breakthrough in applying spin-based electronics to silicon.
- Scientists demonstrate quantum state exchange between light and matter.
- Diamond quantum register developed.
- Controlled-NOT quantum gates on a pair of superconducting quantum bits realized.
- Scientists contain, study hundreds of individual atoms in 3D array.
- Nitrogen in buckyball molecule used in quantum computing.
- Large number of electrons quantum coupled.
- Spin-orbit interaction of electrons measured.
- Atoms quantum manipulated in laser light.
- Light pulses used to control electron spins.
- Quantum effects demonstrated across tens of nanometers.
- Light pulses used to accelerate quantum computing development.
- Quantum RAM blueprint unveiled.
- Model of quantum transistor developed.
- Long distance entanglement demonstrated.
- Photonic quantum computing used to factor number by two independent labs.
- Quantum bus developed by two independent labs.
- Superconducting quantum cable developed.
- Transmission of qubits demonstrated.
- Superior qubit material devised.
- Single electron qubit memory.
- Bose-Einstein condensate quantum memory developed.
- D-Wave Systems demonstrates use of a 28-qubit quantum annealing computer.
- New cryonic method reduces decoherence and increases interaction distance, and thus quantum computing speed.
- Photonic quantum computer demonstrated.
- Graphene quantum dot spin qubits proposed.

2008

- Graphene quantum dot qubits

- Quantum bit stored
- 3D qubit-qutrit entanglement demonstrated
- Analog quantum computing devised
- Control of quantum tunneling
- Entangled memory developed
- Superior NOT gate developed
- Qutrits developed
- Quantum logic gate in optical fiber
- Superior quantum Hall Effect discovered
- Enduring spin states in quantum dots
- Molecular magnets proposed for quantum RAM
- Quasiparticles offer hope of stable quantum computer
- Image storage may have better storage of qubits
- Quantum entangled images
- Quantum state intentionally altered in molecule
- Electron position controlled in silicon circuit
- Superconducting electronic circuit pumps microwave photons
- Amplitude spectroscopy developed
- Superior quantum computer test developed
- Optical frequency comb devised
- Quantum Darwinism supported
- Hybrid qubit memory developed
- Qubit stored for over 1 second in atomic nucleus
- Faster electron spin qubit switching and reading developed
- Possible non-entanglement quantum computing
- D-Wave Systems claims to have produced a 128 qubit computer chip, though this claim has yet to be verified.

2009

- Carbon 12 purified for longer coherence times
- Lifetime of qubits extended to hundreds of milliseconds
- Quantum control of photons
- Quantum entanglement demonstrated over 240 micrometres
- Qubit lifetime extended by factor of 1000
- First electronic quantum processor created

- Six-photon graph state entanglement used to simulate the fractional statistics of anyons living in artificial spin-lattice models
- Single molecule optical transistor
- NIST reads, writes individual qubits
- NIST demonstrates multiple computing operations on qubits
- First large-scale topological cluster state quantum architecture developed for atom-optics
- A combination of all of the fundamental elements required to perform scalable quantum computing through the use of qubits stored in the internal states of trapped atomic ions shown
- Researchers at University of Bristol demonstrate Shor's algorithm on a silicon photonic chip
- Quantum Computing with an Electron Spin Ensemble
- Scalable flux qubit demonstrated
- Photon machine gun developed for quantum computing
- Quantum algorithm developed for differential equation systems
- First universal programmable quantum computer unveiled
- Scientists electrically control quantum states of electrons
- Google collaborates with D-Wave Systems on image search technology using quantum computing
- A method for synchronizing the properties of multiple coupled CJJ rf-SQUID flux qubits with a small spread of device parameters due to fabrication variations was demonstrated
- Realization of Universal Ion Trap Quantum Computation with Decoherence Free Qubits

2010s

2010

- Ion trapped in optical trap
- Optical quantum computer with three qubits calculated the energy spectrum of molecular hydrogen to high precision
- First germanium laser brings us closer to optical computers
- Single electron qubit developed
- Quantum state in macroscopic object
- New quantum computer cooling method developed
- Racetrack ion trap developed
- Evidence for a Moore-Read state in the u = $\frac{5}{2}$ quantum Hall plateau, which would be suitable for topological quantum computation
- Quantum interface between a single photon and a single atom demonstrated
- LED quantum entanglement demonstrated

- Multiplexed design speeds up transmission of quantum information through a quantum communications channel
- Two photon optical chip
- Microfabricated planar ion traps
- Qubits manipulated electrically, not magnetically

2011

- Entanglement in a solid-state spin ensemble
- NOON photons in superconducting quantum integrated circuit
- Quantum antenna
- Multimode quantum interference
- Magnetic Resonance applied to quantum computing
- Quantum pen
- Atomic "Racing Dual"
- 14 qubit register
- D-Wave claims to have developed quantum annealing and introduces their product called D-Wave One. The company claims this is the first commercially available quantum computer
- Repetitive error correction demonstrated in a quantum processor
- Diamond quantum computer memory demonstrated
- Qmodes developed
- Decoherence suppressed
- Simplification of controlled operations
- Ions entangled using microwaves
- Practical error rates achieved
- Quantum computer employing Von Neumann architecture
- Quantum spin Hall topological insulator
- Two Diamonds Linked by Quantum Entanglement could help develop photonic processors

2012

- D-Wave claims a quantum computation using 84 qubits.
- Physicists create a working transistor from a single atom
- A method for manipulating the charge of nitrogen vacancy-centres in diamond
- Reported creation of a 300 qubit/particle quantum simulator.
- Demonstration of topologically protected qubits with an eight-photon entanglement, a robust approach to practical quantum computing

- 1QB Information Technologies (1QBit) founded. World's first dedicated quantum computing software company.
- First design of a quantum repeater system without a need for quantum memories
- Decoherence suppressed for 2 seconds at room temperature by manipulating Carbon-13 atoms with lasers.
- Theory of Bell-based randomness expansion with reduced assumption of measurement independence.
- New low overhead method for fault-tolerant quantum logic developed, called lattice surgery

2013

- Coherence time of 39 minutes at room temperature (and 3 hours at cryogenic temperatures) demonstrated for an ensemble of impurity-spin qubits in isotopically purified silicon.
- Extension of time for qubit maintained in superimposed state for ten times longer than what has ever been achieved before
- First resource analysis of a large-scale quantum algorithm using explicit fault-tolerant, error-correction protocols was developed for factoring

2014

- Documents leaked by Edward Snowden confirm the Penetrating Hard Targets project, by which the National Security Agency seeks to develop a quantum computing capability for cryptography purposes.
- Researchers in Japan and Austria publish the first large-scale quantum computing architecture for a diamond based system
- Scientists at the University of Innsbruck do quantum computations on a topologically encoded qubit which is encoded in entangled states distributed over seven trapped-ion qubits
- Scientists transfer data by quantum teleportation over a distance of 10 feet (3.048 meters) with zero percent error rate, a vital step towards a quantum Internet.
- Nike Dattani & Nathan Bryans break the record for largest number factored on a quantum device: 56153 (previous record was 143).

2015

- Optically addressable nuclear spins in a solid with a six-hour coherence time.
- Quantum information encoded by simple electrical pulses.
- Quantum error detection code using a square lattice of four superconducting qubits.
- D-Wave Systems Inc. announced on June 22 that it had broken the 1000 qubit barrier.
- Two qubit silicon logic gate successfully developed.
- Quantum computer, along with quantum superposition and entanglement, emulated by a classical analog computer, with the result that the fully classical system behaves like a true quantum computer.

2016

- Physicists led by Rainer Blatt joined forces with scientists at MIT, led by Isaac Chuang, to efficiently implement Shor's algorithm in an ion-trap based quantum computer.
- IBM releases the Quantum Experience, an online interface to their superconducting systems. The system is immediately used to publish new protocols in quantum information processing
- Google, using an array of 9 superconducting qubits developed by the Martinis group and UCSB, simulates a hydrogen molecule.
- Scientists in Japan and Australia invent the quantum version of a Sneakernet communications system

2017

- D-Wave Systems Inc. announces general commercial availability of the D-Wave 2000Q quantum annealer, which it claims has 2000 qubits.
- Blueprint for a microwave trapped ion quantum computer published.
- IBM unveils 17-qubit quantum computer—and a better way of benchmarking it.
- Scientists build a microchip that generates two entangled qudits each with 10 states, for 100 dimensions total.
- Microsoft reveals Q Sharp, a quantum programming language integrated with Visual Studio. Programs can be executed locally on a 32-qubit simulator, or a 40-qubit simulator on Azure.
- Intel confirms development of a 17-qubit superconducting test chip.
- IBM reveals a working 50-qubit quantum computer that can maintain its quantum state for 90 microseconds.

2018

- MIT scientists report the discovery of a new triple-photon form of light.
- Oxford researchers successfully used a trapped-ion technique where they place two charged atoms in a state of quantum entanglement, to speed up logic gates by a factor of 20 to 60 times as compared with the previous best gates, translated to 1.6 microseconds long, with 99.8% precision.
- QuTech successfully tests silicon-based 2-spin-qubit processor.
- Google announces the creation of a 72-qubit quantum chip, called "Bristlecone", achieving a new record.
- Intel begins testing silicon-based spin-qubit processor, manufactured in the company's D1D Fab in Oregon.
- Intel confirms development of a 49-qubit superconducting test chip, called "Tangle Lake".
- Japanese researchers demonstrate universal holonomic quantum gates.
- Integrated photonic platform for quantum information with continuous variables.

- On December 17, 2018, the company IonQ introduced the first commercial trapped-ion quantum computer, with a program length of over 60 two-qubit gates, 11 fully connected qubits, 55 addressable pairs, one-qubit gate error <0.03% and two-qubit gate error <1.0%
- On December 21, 2018, the National Quantum Initiative Act was signed into law by President Donald Trump, establishing the goals and priorities for a 10-year plan to accelerate the development of quantum information science and technology applications in the United States.

2019

- IBM unveils its first commercial quantum computer, the IBM Q System One, designed by UK-based Map Project Office and Universal Design Studio and manufactured by Goppion.
- Nike Dattani and co-workers de-code D-Wave's Pegasus architecture and make its description open to the public.
- Austrian physicists demonstrate self-verifying, hybrid, variational quantum simulation of lattice models in condensed matter and high-energy physics using a feedback loop between a classical computer and a quantum co-processor.
- A paper by Google's quantum computer research team was briefly available in late September 2019, claiming the project has reached quantum supremacy.
- IBM reveals its biggest yet quantum computer, consisting of 53 qubits. The system goes online in October 2019.

2020

- UNSW Sydney develops a way of producing 'hot qubits' – quantum devices that operate at 1.5 Kelvin.

- Griffith university, UNSW and UTS in partnership with 7 USA universities develop Noise cancelling for quantum bits via machine learning, taking quantum noise in a quantum chip down to 0%.

- UNSW performs electric nuclear resonance to control single atoms in electronic devices.

- Bob Coecke (Oxford university) explains why NLP is quantum-native. A graphical representation of how the meanings of the words are combined to build the meaning of a sentence as a whole, was created.

- Tokyo university and Australian scientists create and successfully test a solution to the quantum wiring problem, creating a 2d structure for qubits. Such structure can be built using existing integrated circuit technology and has a considerably lower cross-talk.

Shell Keywords in Linux Programming:

echo	read	set	unset
readonly	shift	export	if
fi	else	while	do
done	for	until	case
esac	break	continue	exit
return	trap	wait	eval
exec	ulimit	umask	

Python Comments

Code:

```
#This is a comment

print("Hello, World!")
```

Output on the screen:

```
Hello, World!
```

Python Variables

Code:

```
x = 5
y = "John"
print(x)
print(y)
```

Output on the screen:

```
5

John
```

Code:

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

Output on the screen:

```
Orange

Banana

Cherry
```

Code:

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

Output on the screen:

```
Orange

Orange

Orange
```

Code:

```
x = "awesome"
print("Python is " + x)
```

Output on the screen:

```
Python is awesome
```

Code:

```
x = "Python is "
y = "awesome"
z = x + y
print(z)
```

Output on the screen:

```
Python is awesome
```

Code:

```
x = 5
y = 10
print(x + y)
```

Output on the screen:

```
15
```

Code:

```
x = 5
y = "John"
print(x + y)
```

Output on the screen:

```
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

Code:

```
x = "awesome"

def myfunc():
  print("Python is " + x)

myfunc()
```

Output on the screen:

```
Python is awesome
```

Python Built-in Data Types

Text Type	**str**
Numeric Types	int, float, complex
Sequence Types	list, tuple, range
Mapping Type	dict
Set Types	set, frozenset
Boolean Type	bool
Binary Types	bytes, bytearray, memoryview

Print the data type of the variable x:

Code:

```
x = 5
print(type(x))
```

Output on the screen:

```
<class 'int'>
```

Value of x	**Data Type**
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set

`x = frozenset({"apple", "banana", "cherry"})`	`frozenset`
`x = True`	`bool`
`x = b"Hello"`	`bytes`
`x = bytearray(5)`	`bytearray`
`x = memoryview(bytes(5))`	`memoryview`

Python Numbers

Display a random number between 1 and 9:

Code:

```
import random

print(random.randrange(1, 10))
```

Output on the screen:

```
8
```

Python Casting

Code:

```
x = int(1)
y = float(2.8)
```

```
z = str("s1")
print(x)
print(y)
print(z)
```

Output on the screen:

```
1
2.8
s1
```

Python Strings

Code:

```
a = "Hello"
print(a)
```

Output on the screen:

```
Hello
```

Code:

```
a = """Python """
print(a)
```

Output on the screen:

```
Python
```

Code:

```
a = '''Python'''
print(a)
```

Output on the screen:

```
Python
```

Python Booleans

Code:

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

Output on the screen:

```
True

False

False
```

Python Operators

- **Arithmetic Operators**

Operator	Name	Example

+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

- **Assignment Operators**

Operator	**Example**	**Same As**
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3

%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

- **Comparison Operators**

Operator	**Name**	**Example**
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y

>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

- **Logical Operators**

Operator	**Description**	**Example**
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Returns False if the result is true	not(x < 5 and x < 10)

- **Identity Operators**

Operator	**Description**	**Example**
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

- **Membership Operators**

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

- **Bitwise Operators**

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Python Lists

Code:

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

Output on the screen:

```
['apple', 'banana', 'cherry']
```

Code:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])
```

Output on the screen:

```
banana
```

Code:

```
thislist = ["apple", "banana", "cherry"]
print(thislist[-1])
```

Output on the screen:

```
Cherry
```

Code:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:5])
```

Output on the screen:

```
['cherry', 'orange', 'kiwi']
```

Code:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[:4])
```

Output on the screen:

```
['apple', 'banana', 'cherry', 'orange']
```

Code:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
print(thislist[2:])
```

Output on the screen:

```
['cherry', 'orange', 'kiwi', 'melon', 'mango']
```

Code:

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]
```

```
print(thislist[-4:-1])
```

Output on the screen:

```
['orange', 'kiwi', 'melon']
```

Code:

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

Output on the screen:

```
apple

banana

cherry
```

Code:

```
thislist = ["apple", "banana", "cherry"]
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

Output on the screen:

```
Yes, 'apple' is in the fruits list
```

Code:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"

print(thislist)
```

Output on the screen:

```
['apple', 'blackcurrant', 'cherry']
```

Code:

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))

# List Length
```

Output on the screen:

```
3
```

Code:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)

# add an item to the end of the list
```

Output on the screen:

```
['apple', 'banana', 'cherry', 'orange']
```

Code:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)

# Insert an item as the second position
```

Output on the screen:

```
['apple', 'orange', 'banana', 'cherry']
```

Code:

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

Output on the screen:

```
['apple', 'cherry']
```

Code:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop()
print(thislist)
```

Output on the screen:

```
['apple', 'banana']
```

Code:

```
thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

Output on the screen:

```
['banana', 'cherry']
```

Code:

```
thislist = ["apple", "banana", "cherry"]
del thislist
print(thislist)

# This will cause an error [NameError: name 'thislist' is not defined] because you have
successfully deleted "thislist".
```

Code:

```
thislist = ["apple", "banana", "cherry"]
thislist.clear()
print(thislist)
```

Output on the screen:

```
[]
```

Code:

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)

# copy of a list with the copy() method
```

Output on the screen:

```
['apple', 'banana', 'cherry']
```

Code:

```
thislist = ["apple", "banana", "cherry"]
mylist = list(thislist)
print(mylist)
# copy of a list with the list() method
```

Output on the screen:

```
['apple', 'banana', 'cherry']
```

Code:

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]
```

```
list3 = list1 + list2
print(list3)
```

Output on the screen:

```
['a', 'b', 'c', 1, 2, 3]
```

Code:

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

for x in list2:
  list1.append(x)

print(list1)

# Append list2 into list1
```

Output on the screen:

```
['a', 'b', 'c', 1, 2, 3]
```

Code:

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]

list1.extend(list2)

print(list1)
```

```
# add list2 at the end of list1 using extend() method
```

Output on the screen:

```
['a', 'b', 'c', 1, 2, 3]
```

Code:

```
thislist = list(("apple", "banana", "cherry"))
print(thislist)
```

Output on the screen:

```
['apple', 'banana', 'cherry']
```

Python List Methods

Method	**Description**
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value

extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Python Tuples

Code:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Output on the screen:

```
('apple', 'banana', 'cherry')
```

Code:

```
thistuple = ("apple", "banana", "cherry")
```

```
print(thistuple[1])
```

Output on the screen:

```
banana
```

Code:

```
thistuple = ("apple", "banana", "cherry")
print(thistuple[-1])
```

Output on the screen:

```
cherry
```

Code:

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:5])
```

Output on the screen:

```
('cherry', 'orange', 'kiwi')
```

Code:

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[-4:-1])
```

Output on the screen:

```
('orange', 'kiwi', 'melon')
```

Code:

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

Output on the screen:

```
("apple", "kiwi", "cherry")
```

Code:

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x)
```

Output on the screen:

```
apple

banana

cherry
```

Code:

```
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")
```

Output on the screen:

```
Yes, 'apple' is in the fruits tuple
```

Code:

```
thistuple = ("apple", "banana", "cherry")
print(len(thistuple))

# Tuple Length
```

Output on the screen:

```
3
```

```
# You cannot add items to a tuple

thistuple = ("apple", "banana", "cherry")
thistuple[3] = "orange"
print(thistuple)
```

Error

Code:

```
thistuple = ("apple",)
print(type(thistuple))

thistuple = ("apple")
print(type(thistuple))
```

Output on the screen:

```
<class 'tuple'>

<class 'str'>
```

Code:

```
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple)

# this will raise an error [NameError: name 'thistuple' is not defined] because the
tuple no longer exists
```

Code:

```
tuple1 = ("a", "b", "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

Output on the screen:

```
('a', 'b', 'c', 1, 2, 3)
```

Code:

```
thistuple = tuple(("apple", "banana", "cherry"))
print(thistuple)
```

Output on the screen:

```
('apple', 'banana', 'cherry')
```

Code:

```
thistuple = tuple(("apple", "banana", "cherry"))
print(thistuple)
```

Output on the screen:

```
('apple', 'banana', 'cherry')
```

Python Tuple Methods

Method	**Description**
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Python Sets

Code:

```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

Output on the screen:

```
{'banana', 'apple', 'cherry'}
```

Code:

```
thisset = {"apple", "banana", "cherry"}

for x in thisset:
  print(x)
```

Output on the screen:

```
cherry

banana

apple
```

Code:

```
thisset = {"apple", "banana", "cherry"}

print("banana" in thisset)
```

Output on the screen:

```
True
```

Code:

```
thisset = {"apple", "banana", "cherry"}

thisset.add("orange")

print(thisset)
```

Output on the screen:

```
{'apple', 'cherry', 'orange', 'banana'}
```

Code:

```
thisset = {"apple", "banana", "cherry"}

thisset.update(["orange", "mango", "grapes"])

print(thisset)
```

Output on the screen:

```
{'mango', 'banana', 'orange', 'grapes', 'cherry', 'apple'}
```

Code:

```
thisset = {"apple", "banana", "cherry"}

print(len(thisset))
```

Output on the screen:

```
3
```

Code:

```
thisset = {"apple", "banana", "cherry"}

thisset.remove("banana")

print(thisset)
```

Output on the screen:

```
{'cherry', 'apple'}
```

Code:

```
thisset = {"apple", "banana", "cherry"}

thisset.discard("banana")

print(thisset)
```

Output on the screen:

```
{'apple', 'cherry'}
```

Code:

```
thisset = {"apple", "banana", "cherry"}

x = thisset.pop()

print(x) #removed item

print(thisset) #the set after removal
```

Output on the screen:

```
apple

{'banana', 'cherry'}
```

Code:

```
thisset = {"apple", "banana", "cherry"}

thisset.clear()

print(thisset)
```

Output on the screen:

```
set()
```

Code:

```
thisset = {"apple", "banana", "cherry"}
```

```
del thisset

print(thisset)

# this will raise an error [NameError: name 'thisset' is not defined] because the set
no longer exists
```

Code:

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}

set3 = set1.union(set2)
print(set3)
```

Output on the screen:

```
{3, 1, 'c', 'b', 2, 'a'}
```

Code:

```
set1 = {"a", "b", "c"}
set2 = {1, 2, 3}

set1.update(set2)
print(set1)
```

Output on the screen:

```
{1, 3, 'a', 'c', 'b', 2}
```

Code:

```
thisset = set(("apple", "banana", "cherry")) # note the double round-brackets
print(thisset)
```

Output on the screen:

```
{'banana', 'apple', 'cherry'}
```

Python Set Methods

Method	Description
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets

intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Python Dictionaries

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Output on the screen:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict["model"]
print(x)
```

Output on the screen:

```
Mustang
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict.get("model")
print(x)
```

Output on the screen:

```
Mustang
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

thisdict["year"] = 2018

print(thisdict)
```

Output on the screen:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x in thisdict:
  print(x)
```

Output on the screen:

```
brand

model

year
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x in thisdict:
  print(thisdict[x])
```

Output on the screen:

```
Ford

Mustang

1964
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x in thisdict.values():
  print(x)
```

Output on the screen:

```
Ford

Mustang

1964
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x, y in thisdict.items():
  print(x, y)
```

Output on the screen:

```
brand Ford

model Mustang

year 1964
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
if "model" in thisdict:
  print("Yes, 'model' is one of the keys in the thisdict dictionary")
```

Output on the screen:

```
Yes, 'model' is one of the keys in the thisdict dictionary
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

print(len(thisdict))
```

Output on the screen:

```
3
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
print(thisdict)
```

Output on the screen:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.pop("model")
print(thisdict)
```

Output on the screen:

```
{'brand': 'Ford', 'year': 1964}
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.popitem()
print(thisdict)
```

Output on the screen:

```
{'brand': 'Ford', 'model': 'Mustang'}
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del thisdict["model"]
print(thisdict)
```

Output on the screen:

```
{'brand': 'Ford', 'year': 1964}
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.clear()
print(thisdict)
```

Output on the screen:

```
{}
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
```

```
mydict = thisdict.copy()
print(mydict)
```

Output on the screen:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Code:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = dict(thisdict)
print(mydict)
```

Output on the screen:

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Code:

```
myfamily = {
  "child1" : {
    "name" : "Emil",
    "year" : 2004
  },
  "child2" : {
    "name" : "Tobias",
    "year" : 2007
  },
  "child3" : {
    "name" : "Linus",
    "year" : 2011
  }
}

print(myfamily)
```

Output on the screen:

```
{'child1': {'name': 'Emil', 'year': 2004}, 'child2': {'name': 'Tobias', 'year': 2007},
                      'child3': {'name': 'Linus', 'year': 2011}}
```

Code:

```
thisdict = dict(brand="Ford", model="Mustang", year=1964)
```

Output on the screen:

```
{'model': 'Mustang', 'year': 1964, 'brand': 'Ford'}
```

Python Dictionary Methods

Method	**Description**
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary

fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Java List Methods:

- **int size()**: to get the number of elements in the list.
- **boolean isEmpty()**: to check if list is empty or not.
- **boolean contains(Object o)**: Returns true if this list contains the specified element.
- **Iterator<E> iterator()**: Returns an iterator over the elements in this list in proper sequence.
- **Object[] toArray()**: Returns an array containing all of the elements in this list in proper sequence
- **boolean add(E e)**: Appends the specified element to the end of this list.
- **boolean remove(Object o)**: Removes the first occurrence of the specified element from this list.
- **boolean retainAll(Collection<?> c)**: Retains only the elements in this list that are contained in the specified collection.

- **void clear()**: Removes all the elements from the list.
- **E get(int index)**: Returns the element at the specified position in the list.
- **E set(int index, E element)**: Replaces the element at the specified position in the list with the specified element.
- **ListIterator<E> listIterator()**: Returns a list iterator over the elements in the list.
- **List<E> subList(int fromIndex, int toIndex)**: Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive. The returned list is backed by this list, so non-structural changes in the returned list are reflected in this list, and vice-versa.
- **default void replaceAll(UnaryOperator<E> operator)**: Replaces each element of this list with the result of applying the operator to that element.
- **default void sort(Comparator<super E> c)**: Sorts this list according to the order induced by the specified Comparator.
- **default Spliterator<E> spliterator()**: Creates a Spliterator over the elements in this list.

Python Lambda

Code:

```
x = lambda a: a + 10
print(x(5))
```

Output on the screen:

```
15
```

Code:

```
x = lambda a, b: a * b
print(x(5, 6))
```

Output on the screen:

```
30
```

Code:

```
x = lambda a, b, c: a + b + c
print(x(5, 6, 2))
```

Output on the screen:

```
13
```

Code:

```
def myfunc(n):
  return lambda a : a * n

mydoubler = myfunc(2)

print(mydoubler(11))
```

Output on the screen:

```
22
```

Python Arrays

Code:

```
cars = ["Ford", "Volvo", "BMW"]
```

```
print(cars)
```

Output on the screen:

```
['Ford', 'Volvo', 'BMW']
```

Code:

```
cars = ["Ford", "Volvo", "BMW"]

x = cars[0]

print(x)
```

Output on the screen:

```
Ford
```

Code:

```
cars = ["Ford", "Volvo", "BMW"]

cars[0] = "Toyota"

print(cars)
```

Output on the screen:

```
['Toyota', 'Volvo', 'BMW']
```

Code:

```
cars = ["Ford", "Volvo", "BMW"]

x = len(cars)

print(x)
```

Output on the screen:

```
3
```

Code:

```
cars = ["Ford", "Volvo", "BMW"]

for x in cars:
  print(x)
```

Output on the screen:

```
Ford

Volvo

BMW
```

Code:

```
cars = ["Ford", "Volvo", "BMW"]
cars.append("Honda")
print(cars)
```

Output on the screen:

```
['Ford', 'Volvo', 'BMW', 'Honda']
```

Code:

```
cars = ["Ford", "Volvo", "BMW"]

cars.remove("Volvo")

print(cars)
```

Output on the screen:

```
['Ford', 'BMW']
```

Code:

```
cars = ["Ford", "Volvo", "BMW"]

cars.pop(1)

print(cars)
```

Output on the screen:

```
['Ford', 'BMW']
```

Python Array Methods

Method	**Description**
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Python Classes

Code:

```
class MyClass:
  x = 5

p1 = MyClass()
print(p1.x)
```

Output on the screen:

```
5
```

Python Iterators

Code:

```
mytuple = ("apple", "banana", "cherry")
myit = iter(mytuple)

print(next(myit))
print(next(myit))
print(next(myit))
```

Output on the screen:

```
apple

banana

cherry
```

Code:

```
mystr = "banana"
myit = iter(mystr)

print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
print(next(myit))
```

Output on the screen:

```
b

a

n

a

n

a
```

Code:

```
mytuple = ("apple", "banana", "cherry")

for x in mytuple:
  print(x)
```

Output on the screen:

```
apple

banana
```

```
cherry
```

Code:

```
mystr = "banana"

for x in mystr:
  print(x)
```

Output on the screen:

```
b

a

n

a

n

a
```

Python Math

Code:

```
x = min(5, 10, 25)
y = max(5, 10, 25)

print(x)
print(y)
```

Output on the screen:

```
5
25
```

Code:

```
x = pow(4, 3)
print(x)
```

Output on the screen:

```
64
```

Code:

```
import math
x = math.sqrt(64)
print(x)
```

Output on the screen:

```
8.0
```

Code:

```
import math
```

```
# Round a number upward to its nearest integer
x = math.ceil(1.4)

# Round a number downward to its nearest integer
y = math.floor(1.4)

print(x)
print(y)
```

Output on the screen:

```
2

1
```

Code:

```
import math

x = math.pi

print(x)
```

Output on the screen:

```
3.141592653589793
```

Math Methods

Method	Description

math.acos(x)	Returns the arc cosine value of x
math.acosh(x)	Returns the hyperbolic arc cosine of x
math.asin(x)	Returns the arc sine of x
math.asinh(x)	Returns the hyperbolic arc sine of x
math.atan(x)	Returns the arc tangent value of x
math.atan2(y, x)	Returns the arc tangent of y/x in radians
math.atanh(x)	Returns the hyperbolic arctangent value of x
math.ceil(x)	Rounds a number upwards to the nearest integer, and returns the result
math.comb(n, k)	Returns the number of ways to choose k items from n items without repetition and order
math.copysign(x, y)	Returns a float consisting of the value of the first parameter and the sign of the second parameter
math.cos(x)	Returns the cosine of x
math.cosh(x)	Returns the hyperbolic cosine of x
math.degrees(x)	Converts an angle from radians to degrees
math.dist(p, q)	Calculates the euclidean distance between two specified points (p and q), where p and q are the coordinates of that point
math.erf(x)	Returns the error function of x
math.erfc(x)	Returns the complementary error function of x
math.exp(x)	Returns the value of E^x, where E is Euler's number (approximately 2.718281...), and x is the number passed to it
math.expm1(x)	Returns the value of E^x - 1, where E is Euler's number (approximately 2.718281...),

	and x is the number passed to it
math.fabs(x)	Returns the absolute value of a number
math.factorial()	Returns the factorial of a number
math.floor(x)	Rounds a number downwards to the nearest integer, and returns the result
math.fmod(x, y)	Returns the remainder of specified numbers when a number is divided by another number
math.frexp()	Returns the mantissa and the exponent, of a specified value
math.fsum(iterable)	Returns the sum of all items in an iterable (tuples, arrays, lists, etc.)
math.gamma(x)	Returns the gamma value of x
math.gcd()	Returns the highest value that can divide two integers
math.hypot()	Find the Euclidean distance from the origin for *n* inputs
math.isclose()	Checks whether two values are close, or not
math.isfinite(x)	Checks whether x is a finite number
math.isinf(x)	Check whether x is a positive or negative infinty
math.isnan(x)	Checks whether x is NaN (not a number)
math.isqrt(n)	Returns the nearest integer square root of n
math.ldexp(x, i)	Returns the expression x * 2i where x is mantissa and i is an exponent
math.lgamma(x)	Returns the log gamma value of x
math.log(x, base)	Returns the natural logarithm of a number, or

	the logarithm of number to base
math.log10(x)	Returns the base-10 logarithm of x
math.log1p(x)	Returns the natural logarithm of 1+x
math.log2(x)	Returns the base-2 logarithm of x
math.perm(n, k)	Returns the number of ways to choose k items from n items with order and without repetition
math.pow(x, y)	Returns the value of x to the power of y
math.prod(iterable, *, start=1)	Returns the product of an iterable (lists, array, tuples, etc.)
math.radians(x)	Converts a degree value (x) to radians
math.remainder(x, y)	Returns the closest value that can make numerator completely divisible by the denominator
math.sin(x)	Returns the sine of x
math.sinh(x)	Returns the hyperbolic sine of x
math.sqrt(x)	Returns the square root of x
math.tan(x)	Returns the tangent of x
math.tanh(x)	Returns the hyperbolic tangent of x
math.trunc(x)	Returns the truncated integer parts of x

Math Constants

Constant	**Description**

math.e	Returns Euler's number (2.7182...)
math.inf	Returns a floating-point positive infinity
math.nan	Returns a floating-point NaN (Not a Number) value
math.pi	Returns PI (3.1415...)
math.tau	Returns tau (6.2831...)

Python JSON

Code:

```
import json

x = '{"name":"John", "age":30, "city":"New York"}'

y = json.loads(x)

print(y["age"])
```

Output on the screen:

```
30
```

Python RegEx

Code:

```
import re

# Check if the string starts with "The" and ends with "Spain":
```

```
txt = "The rain in Spain"
x = re.search("^The.*Spain$", txt)

if x:
  print("YES! We have a match!")
else:
  print("No match")
```

Output on the screen:

```
YES! We have a match!
```

Python Try Except

Code:

```
# The try block will generate an error, because x is not defined:

try:
  print(x)
except:
  print("An exception occurred")
```

Output on the screen:

```
An exception occurred
```

Code:

```
# The try block will generate a NameError, because x is not defined:
```

```
try:
  print(x)
except NameError:
  print("Variable x is not defined")
except:
  print("Something else went wrong")
```

Output on the screen:

```
Variable x is not defined
```

Python String Formatting

Code:

```
price = 49
txt = "The price is {} dollars"
print(txt.format(price))
```

Output on the screen:

```
The price is 49 dollars
```

Code:

```
price = 49
txt = "The price is {:.2f} dollars"
print(txt.format(price))
```

Output on the screen:

```
The price is 49.00 dollars
```

Code:

```
quantity = 3
itemno = 567
price = 49
myorder = "I want {0} pieces of item number {1} for {2:.2f} dollars."
print(myorder.format(quantity, itemno, price))
```

Output on the screen:

```
I want 3 pieces of item number 567 for 49.00 dollars.
```

Python If ... Else

Code:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Output on the screen:

```
a is greater than b
```

Python Loops

Code:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":
    break
```

Output on the screen:

```
apple

banana
```

Code:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

Output on the screen:

```
apple
```

Code:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

Output on the screen:

```
apple

cherry
```

Code:

```
for x in range(6):
  print(x)
```

Output on the screen:

```
0

1

2

3

4

5
```

Code:

```
for x in range(2, 6):
  print(x)
```

Output on the screen:

```
2

3

4

5
```

Code:

```
for x in range(2, 30, 3):
  print(x)
```

Output on the screen:

```
2

5

8

11

14

17

20

23

26
```

Code:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Output on the screen:

```
0

1

2

3

4

5

Finally finished!
```

Code:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

Output on the screen:

```
red apple

red banana

red cherry

big apple

big banana

big cherry

tasty apple

tasty banana

tasty cherry
```

Code:

```
for x in [0, 1, 2]:
  pass

# having an empty for loop like this, would raise an error without the pass statement
```

Code:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Output on the screen:

```
1

2

3

4

5
```

Code:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Output on the screen:

```
1

2

3

4

5
```

Code:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i is no longer less than 6")
```

Output on the screen:

```
1

2

3

4

5

i is no longer less than 6
```

Code:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

Output on the screen:

```
1

2

4

5

6
```

Code:

```
i = 1
while i < 6:
  print(i)
  if (i == 3):
    break
  i += 1
```

Output on the screen:

```
1

2

3
```

Python User Input

Code:

```
username = input("Enter username:")
print("Username is: " + username)
```

Output on the screen:

```
Enter username:

If you enter the word "ram"

Username is: ram

will be outputted on the console screen.
```

Timeline of computer security hacker history

1900

1903

- Magician and inventor Nevil Maskelyne disrupts John Ambrose Fleming's public demonstration of Guglielmo Marconi's purportedly secure wireless telegraphy technology, sending insulting Morse code messages through the auditorium's projector.

1930s

1932

- Polish cryptologists Marian Rejewski, Henryk Zygalski and Jerzy Różycki broke the Enigma machine code.

1939

- Alan Turing, Gordon Welchman and Harold Keen worked together to develop the Bombe (on the basis of Rejewski's works on Bomba). The Enigma machine's use of a reliably small key space makes it vulnerable to brute force.

1940s

1943

- René Carmille, comptroller general of the Vichy French Army, hacked the punched card system used by the Nazis to locate Jews.

1949

- The theory that underlies computer viruses was first made public in 1949, when computer pioneer John von Neumann presented a paper titled "Theory and Organization of Complicated Automata." In the paper von Neumann speculated that computer programs could reproduce themselves.

1950s

1955

- At MIT, "hack" first came to mean fussing with machines. The minutes of an April, 1955, meeting of the Tech Model Railroad Club state that "Mr. Eccles requests that anyone working or hacking on the electrical system turn the power off to avoid fuse blowing."

1957

- Joe "Joybubbles" Engressia, a blind seven-year-old boy with perfect pitch, discovered that whistling the fourth E above middle C (a frequency of 2600 Hz) would interfere with AT&T's automated telephone systems, thereby inadvertently opening the door for phreaking.

1960s

- Various phreaking boxes are used to interact with automated telephone systems.

1963

- The first ever reference to malicious hacking is 'telephone hackers' in MIT's student newspaper, *The Tech* of hackers tying up the lines with Harvard, configuring the PDP-1 to make free calls, war dialing and accumulating large phone bills.

1965

- William D. Mathews from MIT found a vulnerability in a **CTSS** running on an IBM 7094. The standard text editor on the system was designed to be used by one user at a time, working in one directory, and so created a temporary file with a constant name for all instantiations of the editor. The flaw was discovered when two system programmers were editing at the same time and the temporary files for the message-of-the day and the password file became swapped, causing the contents of the system CTSS password file to display to any user logging into the system.

1967

- The first known incidence of network penetration hacking took place when members of a computer club at a suburban Chicago area high school were provided access to IBM's APL network. In the Fall of 1967, IBM (through Science Research Associates) approached Evanston Township High School with the offer of four 2741 Selectric teletypewriter based terminals with dial-up modem connectivity to an experimental computer system which implemented an early version of the APL programming language. The APL network system was structured in Workspaces which were assigned to various clients using the system. Working independently, the students quickly learned the language and the system. They were free to explore the system, often using existing code available in public Workspaces as models for their own creations. Eventually, curiosity drove the students to explore the system's wider context. This first informal network penetration effort was later acknowledged as helping harden the security of one of the first publicly accessible networks:

> Science Research Associates undertook to write a full APL system for the IBM 1500. They modeled their system after APL/360, which had by that time been developed and seen substantial use inside of IBM, using code borrowed from MAT/1500 where possible. In their documentation they acknowledge their gratitude to "a number of high school students for their compulsion to bomb the system". This was an early example of a kind of sportive, but very effective, debugging that was often repeated in the evolution of APL systems.

1970s

1971

- John T. Draper (later nicknamed Captain Crunch), his friend Joe Engressia (also known as Joybubbles), and blue box phone phreaking hit the news with an *Esquire Magazine* feature story.

1979

- Kevin Mitnick breaks into his first major computer system, the Ark, the computer system Digital Equipment Corporation (DEC) used for developing their RSTS/E operating system software.

1980s

1980

- The FBI investigates a breach of security at National CSS (NCSS). *The New York Times*, reporting on the incident in 1981, describes hackers as

  > technical experts; skilled, often young, computer programmers, who almost whimsically probe the defenses of a computer system, searching out the limits and the possibilities of the machine. Despite their seemingly subversive role, hackers are a recognized asset in the computer industry, often highly prized

  The newspaper describes white hat activities as part of a "mischievous but perversely positive 'hacker' tradition". When a National CSS employee revealed the existence of his password cracker, which he had used on customer accounts, the company chastised him not for writing the software but for not disclosing it sooner. The letter of reprimand stated that "The Company realizes the benefit to NCSS and in fact encourages the efforts of employees to identify security weaknesses to the VP, the directory, and other sensitive software in files".

1981

- Chaos Computer Club forms in Germany.
- Ian Murphy aka Captain Zap, was the first cracker to be tried and convicted as a felon. Murphy broke into AT&T's computers in 1981 and changed the internal clocks that metered billing rates. People were getting late-night discount rates when they called at midday. Of course, the bargain-seekers who waited until midnight to call long distance were hit with high bills.

1983

- The 414s break into 60 computer systems at institutions ranging from the Los Alamos National Laboratory to Manhattan's Memorial Sloan-Kettering Cancer Center. The incident appeared as the cover story of Newsweek with the title "Beware: Hackers at play". As a result, the U.S. House of Representatives held hearings on computer security and passed several laws.
- The group KILOBAUD is formed in February, kicking off a series of other hacker groups which form soon after.
- The movie *WarGames* introduces the wider public to the phenomenon of hacking and creates a degree of mass paranoia of hackers and their supposed abilities to bring the world to a screeching halt by launching nuclear ICBMs.
- The U.S. House of Representatives begins hearings on computer security hacking.
- In his Turing Award lecture, Ken Thompson mentions "hacking" and describes a security exploit that he calls a "Trojan horse".

1984

- Someone calling himself Lex Luthor founds the Legion of Doom. Named after a Saturday morning cartoon, the LOD had the reputation of attracting "the best of the best"—until one of the most talented members called Phiber Optik feuded with Legion of Doomer Erik Bloodaxe and got 'tossed out of the clubhouse'. Phiber's friends formed a rival group, the Masters of Deception.
- The Comprehensive Crime Control Act gives the Secret Service jurisdiction over computer fraud.
- Cult of the Dead Cow forms in Lubbock, Texas, and begins publishing its ezine.
- The hacker magazine *2600* begins regular publication, right when TAP was putting out its final issue. The editor of *2600*, "Emmanuel Goldstein" (whose real name is Eric Corley), takes his handle from the leader of the resistance in George Orwell's *1984*. The publication provides tips for would-be hackers and phone

phreaks, as well as commentary on the hacker issues of the day. Today, copies of *2600* are sold at most large retail bookstores.

- The Chaos Communication Congress, the annual European hacker conference organized by the Chaos Computer Club, is held in Hamburg, Germany.
- William Gibson's groundbreaking science fiction novel *Neuromancer*, about "Case", a futuristic computer hacker, is published. Considered the first major cyberpunk novel, it brought into hacker jargon such terms as "cyberspace", "the matrix", "simstim", and "ICE".

1985

- KILOBAUD is re-organized into The P.H.I.R.M., and begins sysopping hundreds of BBSs throughout the United States, Canada, and Europe.
- The online 'zine *Phrack* is established.
- *The Hacker's Handbook* is published in the UK.
- The FBI, Secret Service, Middlesex County NJ Prosecutor's Office and various local law enforcement agencies execute seven search warrants concurrently across New Jersey on July 12, 1985, seizing equipment from BBS operators and users alike for "complicity in computer theft", under a newly passed, and yet untested criminal statute. This is famously known as the Private Sector Bust, or the 2600 BBS Seizure, and implicated the Private Sector BBS sysop, Store Manager (also a BBS sysop), Beowulf, Red Barchetta, The Vampire, the NJ Hack Shack BBS sysop, and the Treasure Chest BBS sysop.

1986

- After more and more break-ins to government and corporate computers, Congress passes the Computer Fraud and Abuse Act, which makes it a crime to break into computer systems. The law, however, does not cover juveniles.
- Robert Schifreen and Stephen Gold are convicted of accessing the Telecom Gold account belonging to the Duke of Edinburgh under the Forgery and Counterfeiting Act 1981 in the United Kingdom, the first conviction for illegally accessing a computer system. On appeal, the conviction is overturned as hacking is not within the legal definition of forgery.
- Arrest of a hacker who calls himself The Mentor. He published a now-famous treatise shortly after his arrest that came to be known as the Hacker's Manifesto in the e-zine Phrack. This still serves as the most famous piece of hacker literature and is frequently used to illustrate the mindset of hackers.
- Astronomer Clifford Stoll plays a pivotal role in tracking down hacker Markus Hess, events later covered in Stoll's 1990 book *The Cuckoo's Egg*.

1987

- The Christmas Tree EXEC "worm" causes major disruption to the VNET, BITNET and EARN networks.

1988

- The *Morris Worm*. Graduate student Robert T. Morris, Jr. of Cornell University launches a worm on the government's ARPAnet (precursor to the Internet). The worm spreads to 6,000 networked computers, clogging government and university systems. Robert Morris is dismissed from Cornell, sentenced to three years probation, and fined $10,000.
- First National Bank of Chicago is the victim of $70-million computer theft.
- The Computer Emergency Response Team (CERT) is created by DARPA to address network security.
- The Father Christmas (computer worm) spreads over DECnet networks.

1989

- Jude Milhon (aka St Jude) and R. U. Sirius launch Mondo 2000, a major '90s tech-lifestyle magazine, in Berkeley, California.
- The politically motivated WANK worm spreads over DECnet.
- Dutch magazine Hack-Tic begins.
- The Cuckoo's Egg by Clifford Stoll is published.
- The detection of AIDS (Trojan horse) is the first instance of a ransomware detection.

1990s

1990

- Operation Sundevil introduced. After a prolonged sting investigation, Secret Service agents swoop down on organizers and prominent members of BBSs in 14 U.S. cities including the Legion of Doom, conducting early-morning raids and arrests. The arrests involve and are aimed at cracking down on credit-card theft and telephone and wire fraud. The result is a breakdown in the hacking community, with members informing on each other in exchange for immunity. The offices of Steve Jackson Games are also raided, and the role-playing sourcebook GURPS Cyberpunk is confiscated, possibly because the government fears it is a "handbook for computer crime". Legal battles arise that prompt the formation of the Electronic Frontier Foundation, including the trial of Knight Lightning.
- Australian federal police tracking *Realm* members *Phoenix*, *Electron* and *Nom* are the first in the world to use a remote data intercept to gain evidence for a computer crime prosecution.
- The Computer Misuse Act 1990 is passed in the United Kingdom, criminalising any unauthorised access to computer systems.

1992

- Release of the movie *Sneakers*, in which security experts are blackmailed into stealing a universal decoder for encryption systems.
- One of the first ISPs, MindVox, opens to the public.
- Bulgarian virus writer Dark Avenger wrote 1260, the first known use of polymorphic code, used to circumvent the type of pattern recognition used by antivirus software, and nowadays also intrusion detection systems.
- Publication of a hacking instruction manual for penetrating TRW credit reporting agency by Infinite Possibilities Society (IPS) gets Dr. Ripco, the sysop of Ripco BBS mentioned in the IPS manual, arrested by the United States Secret Service.

1993

- The first DEF CON hacking conference takes place in Las Vegas. The conference is meant to be a one-time party to say good-bye to BBSs (now replaced by the Web), but the gathering was so popular it became an annual event.
- AOL gives its users access to Usenet, precipitating Eternal September.

1994

- Summer: Russian crackers siphon $10 million from Citibank and transfer the money to bank accounts around the world. Vladimir Levin, the 30-year-old ringleader, used his work laptop after hours to transfer the funds to accounts in Finland and Israel. Levin stands trial in the United States and is sentenced to three years in prison. Authorities recover all but $400,000 of the stolen money.
- Hackers adapt to emergence of the World Wide Web quickly, moving all their how-to information and hacking programs from the old BBSs to new hacker web sites.
- AOHell is released, a freeware application that allows a burgeoning community of unskilled script kiddies to wreak havoc on America Online. For days, hundreds of thousands of AOL users find their mailboxes flooded with multi-megabyte email bombs and their chat rooms disrupted with spam messages.
- December 27: After experiencing an IP spoofing attack by Kevin Mitnick, computer security expert Tsutomu Shimomura started to receive prank calls that popularized the phrase "My kung fu is stronger than yours".

1995

- The movies *The Net* and *Hackers* are released.

- The Canadian ISP dlcwest.com is hacked and website replaced with a graphic and the caption "You've been hacked MOFO"
- February 22: The FBI raids the "Phone Masters".

1996

- Hackers alter Web sites of the United States Department of Justice (August), the CIA (October), and the U.S. Air Force (December).
- Canadian hacker group, Brotherhood, breaks into the Canadian Broadcasting Corporation.
- Arizona hacker, John Sabo A.K.A FizzleB/Peanut, was arrested for hacking Canadian ISP dlcwest.com claiming the company was defrauding customers through over billing.
- The U.S. General Accounting Office reports that hackers attempted to break into Defense Department computer files some 250,000 times in 1995 alone. About 65 percent of the attempts were successful, according to the report.
- Cryptovirology is born with the invention of the cryptoviral extortion protocol that would later form the basis of modern ransomware.

1997

- A 15-year-old Croatian youth penetrates computers at a U.S. Air Force base in Guam.
- June: Eligible Receiver 97 tests the American government's readiness against cyberattacks.
- December: Information Security publishes first issue.
- First high-profile attacks on Microsoft's Windows NT operating system

1998

- January: Yahoo! notifies Internet users that anyone visiting its site in the past month might have downloaded a logic bomb and worm planted by hackers claiming a "logic bomb" will go off if computer hacker Kevin Mitnick is not released from prison.
- February: The Internet Software Consortium proposes the use of DNSSEC (domain-name system security extensions) to secure DNS servers.
- May 19: The seven members of the hacker think tank known as L0pht testify in front of the US congressional Government Affairs committee on "Weak Computer Security in Government".
- June: Information Security publishes its first annual Industry Survey, finding that nearly three-quarters of organizations suffered a security incident in the previous year.
- September: Electronic Disturbance Theater, an online political performance-art group, attacks the websites of The Pentagon, Mexican president Ernesto Zedillo, and the Frankfurt Stock Exchange, calling it conceptual art and claiming it to be a protest against the suppression of the Zapatista Army of National

Liberation in southern Mexico. EDT uses the FloodNet software to bombard its opponents with access requests.

- October: "U.S. Attorney General Janet Reno announces National Infrastructure Protection Center."

1999

- Software security goes mainstream In the wake of Microsoft's Windows 98 release, 1999 becomes a banner year for security (and hacking). Hundreds of advisories and patches are released in response to newfound (and widely publicized) bugs in Windows and other commercial software products. A host of security software vendors release anti-hacking products for use on home computers.
- U.S. President Bill Clinton announces a $1.46 billion initiative to improve government computer security. The plan would establish a network of intrusion detection monitors for certain federal agencies and encourage the private sector to do the same.
- January 7: The "Legion of the Underground" (LoU) declares "war" against the governments of Iraq and the People's Republic of China. An international coalition of hackers (including Cult of the Dead Cow, *2600*'s staff, *Phrack*'s staff, L0pht, and the Chaos Computer Club) issued a joint statement () condemning the LoU's declaration of war. The LoU responded by withdrawing its declaration.
- March: The Melissa worm is released and quickly becomes the most costly malware outbreak to date.
- July: Cult of the Dead Cow releases Back Orifice 2000 at DEF CON.
- August: Kevin Mitnick, sentenced to 5 years, of which over 4 years had already been spent pre-trial including 8 months solitary confinement.
- September: Level Seven Crew hacks the U.S. Embassy in China's website and places racist, anti-government slogans on embassy site in regards to 1998 U.S. embassy bombings.
- September 16: The United States Department of Justice sentences the "Phone Masters".
- October: American Express introduces the "Blue" smart card, the industry's first chip-based credit card in the US.
- November 17: A hacker interviewed by Hilly Rose during the radio show *Coast to Coast AM* (then hosted by Art Bell) exposes a plot by al-Qaeda to derail Amtrak trains. This results in all trains being forcibly stopped over Y2K as a safety measure.

2000s

2000

- May: The ILOVEYOU worm, also known as VBS/Loveletter and Love Bug worm, is a computer worm written in VBScript. It infected millions of computers worldwide within a few hours of its release. It is considered to be one of the most damaging worms ever. It originated in the Philippines; made by an AMA Computer College student Onel de Guzman for his thesis.

- September: Computer hacker Jonathan James became the first juvenile to serve jail time for hacking.

2001

- Microsoft becomes the prominent victim of a new type of hack that attacks the domain name server. In these denial-of-service attacks, the DNS paths that take users to Microsoft's websites are corrupted.
- February: A Dutch cracker releases the Anna Kournikova virus, initiating a wave of viruses that tempts users to open the infected attachment by promising a sexy picture of the Russian tennis star.
- April: FBI agents trick two Russian crackers into coming to the U.S. and revealing how they were hacking U.S. banks.
- July: Russian programmer Dmitry Sklyarov is arrested at the annual Def Con hacker convention. He was the first person criminally charged with violating the Digital Millennium Copyright Act (DMCA).
- August: Code Red worm, infects tens of thousands of machines.
- The National Cyber Security Alliance (NCSA) is established in response to the September 11 attacks on the World Trade Center.

2002

- January: Bill Gates decrees that Microsoft will secure its products and services, and kicks off a massive internal training and quality control campaign.
- May: Klez.H, a variant of the worm discovered in November 2001, becomes the biggest malware outbreak in terms of machines infected, but causes little monetary damage.
- June: The Bush administration files a bill to create the Department of Homeland Security, which, among other things, will be responsible for protecting the nation's critical IT infrastructure.
- August: Researcher Chris Paget publishes a paper describing "shatter attacks", detailing how Windows' unauthenticated messaging system can be used to take over a machine. The paper raises questions about how securable Windows could ever be. It is however largely derided as irrelevant as the vulnerabilities it described are caused by vulnerable applications (placing windows on the desktop with inappropriate privileges) rather than an inherent flaw within the Operating System.
- October: The International Information Systems Security Certification Consortium - (ISC)² - confers its 10,000th CISSP certification.

2003

- The hacktivist group Anonymous was formed.
- March: Cult of the Dead Cow and Hacktivismo are given permission by the United States Department of Commerce to export software utilizing strong encryption.

2004

- March: New Zealand's Government (National Party) website defaced by hacktivist group BlackMask
- July: North Korea claims to have trained 500 hackers who successfully crack South Korean, Japanese, and their allies' computer systems.
- October: National Cyber Security Awareness Month was launched by the National Cyber Security Alliance and U.S. Department of Homeland Security.

2005

- April 2: Rafael Núñez (aka RaFa), a notorious member of the hacking group World of Hell, is arrested following his arrival at Miami International Airport for breaking into the Defense Information Systems Agency computer system in June 2001.
- September 13: Cameron Lacroix is sentenced to 11 months for gaining access to T-Mobile's network and exploiting Paris Hilton's Sidekick.
- November 3: Jeanson James Ancheta, whom prosecutors say was a member of the "Botmaster Underground", a group of script kiddies mostly noted for their excessive use of bot attacks and propagating vast amounts of spam, was taken into custody after being lured to FBI offices in Los Angeles.

2006

- January: One of the few worms to take after the old form of malware, destruction of data rather than the accumulation of zombie networks to launch attacks from, is discovered. It had various names, including Kama Sutra (used by most media reports), Black Worm, Mywife, Blackmal, Nyxem version D, Kapser, KillAV, Grew and CME-24. The worm would spread through e-mail client address books, and would search for documents and fill them with garbage, instead of deleting them to confuse the user. It would also hit a web page counter when it took control, allowing the programmer who created it as well as the world to track the progress of the worm. It would replace documents with random garbage on the third of every month. It was hyped by the media but actually affected relatively few computers, and was not a real threat for most users.
- May: Jeanson James Ancheta receives a 57-month prison sentence, and is ordered to pay damages amounting to $15,000.00 to the Naval Air Warfare Center in China Lake and the Defense Information Systems Agency, for damage done due to DDoS attacks and hacking. Ancheta also had to forfeit his gains to the government, which include $60,000 in cash, a BMW, and computer equipment.
- May: The largest defacement in Web History as of that time is performed by the Turkish hacker iSKORPiTX who successfully hacked 21,549 websites in one shot.
- July: Edwin Pena is the first person to be charged by U.S. authorities with VoIP hacking. He was sentenced to 10 years and a $1 million restitution.

- September: Viodentia releases FairUse4WM tool which would remove DRM information off Windows Media Audio (WMA) files downloaded from music services such as Yahoo! Unlimited, Napster, Rhapsody Music and Urge.

2007

- May 17: Estonia recovers from massive denial-of-service attack
- June 13: FBI Operation Bot Roast finds over 1 million botnet victims
- June 21: A spear phishing incident at the Office of the Secretary of Defense steals sensitive U.S. defense information, leading to significant changes in identity and message-source verification at OSD.

- August 11: United Nations website hacked by Turkish Hacker Kerem125.
- November 14: Panda Burning Incense which is known by several other names, including Fujacks and Radoppan.T lead to the arrest of eight people in China. Panda Burning Incense was a parasitic virus that infected executable files on a PC. When infected, the icon of the executable file changes to an image of a panda holding three sticks of incense. The arrests were the first for virus writing in China.

2008

- January 17: Project Chanology; Anonymous attacks Scientology website servers around the world. Private documents are stolen from Scientology computers and distributed over the Internet.
- March 7: Around 20 Chinese hackers claim to have gained access to the world's most sensitive sites, including The Pentagon. They operated from an apartment on a Chinese Island.
- March 14: Trend Micro website successfully hacked by Turkish hacker Janizary (aka Utku).

2009

- April 4: Conficker worm infiltrated millions of PCs worldwide including many government-level top-security computer networks.

2010s

2010

- January 12: Operation Aurora Google publicly reveals that it has been on the receiving end of a *"highly sophisticated and targeted attack on our corporate infrastructure originating from China that resulted in the theft of intellectual property from Google"*
- June: Stuxnet The Stuxnet worm is found by VirusBlokAda. Stuxnet was unusual in that while it spread via Windows computers, its payload targeted just one specific model and type of SCADA systems. It slowly

became clear that it was a cyber attack on Iran's nuclear facilities - with most experts believing that Israel was behind it - perhaps with US help.

- December 3: The first Malware Conference, MALCON took place in India. Founded by Rajshekhar Murthy, malware coders are invited to showcase their skills at this annual event supported by the Government of India. An advanced malware for Symbian OS is released by hacker A0drul3z.

2011

- The hacker group Lulz Security is formed.
- April 9: Bank of America website got hacked by a Turkish hacker named JeOPaRDY. An estimated 85,000 credit card numbers and accounts were reported to have been stolen due to the hack. Bank officials say no personal customer bank information is available on that web-page. Investigations are being conducted by the FBI to trace down the incriminated hacker.
- April 17: An "external intrusion" sends the PlayStation Network offline, and compromises personally identifying information (possibly including credit card details) of its 77 million accounts, in what is claimed to be one of the five largest data breaches ever.
- Computer hacker sl1nk releases information of his penetration in the servers of the Department of Defense (DoD), Pentagon, NASA, NSA, US Military, Department of the Navy, Space and Naval Warfare System Command and other UK/US government websites.
- September: Bangladeshi hacker TiGER-M@TE made a world record in defacement history by hacking 700,000 websites in a single shot.
- October 16: The YouTube channel of *Sesame Street* was hacked, streaming pornographic content for about 22 minutes.
- November 1: The main phone and Internet networks of the Palestinian territories sustained a hacker attack from multiple locations worldwide.
- November 7: The forums for Valve's Steam service were hacked. Redirects for a hacking website, Fkn0wned, appeared on the Steam users' forums, offering "hacking tutorials and tools, porn, free giveaways and much more."
- December 14: Five members of the Norwegian hacker group, Noria, were arrested, allegedly suspected for hacking into the email account of the militant extremist Anders Behring Breivik (who perpetrated the 2011 attacks in the country).

2012

- A Saudi hacker, 0XOMAR, published over 400,000 credit cards online, and threatened Israel to release 1 million credit cards in the future. In response to that incident, an Israeli hacker published over 200 Saudi's credit cards online.

- January 7: "Team Appunity", a group of Norwegian hackers, were arrested for breaking into Norway's largest prostitution website then publishing the user database online.
- February 3: Marriott was hacked by a New Age ideologist, Attila Nemeth who was resisting against the New World Order where he said that corporations are allegedly controlling the world. As a response Marriott reported him to the United States Secret Service.
- February 8: Foxconn is hacked by a hacker group, "Swagg Security", releasing a massive amount of data including email and server logins, and even more alarming - bank account credentials of large companies like Apple and Microsoft. Swagg Security stages the attack just as a Foxconn protest ignites against terrible working conditions in southern China.
- May 4: The websites of several Turkish representative offices of international IT-companies are defaced within the same day by F0RTYS3V3N (Turkish Hacker), including the websites of Google, Yandex, Microsoft, Gmail, MSN, Hotmail, PayPal.
- May 24: WHMCS is hacked by UGNazi, they claim that the reason for this is because of the illegal sites that are using their software.
- May 31: MyBB is hacked by newly founded hacker group, UGNazi, the website was defaced for about a day, they claim their reasoning for this was because they were upset that the forum board Hackforums.net uses their software.
- June 5: The social networking website LinkedIn has been hacked and the passwords for nearly 6.5 million user accounts are stolen by cybercriminals. As a result, a United States grand jury indicted Nikulin and three unnamed co-conspirators on charges of aggravated identity theft and computer intrusion.
- August 15: The most valuable company in the world Saudi Aramco is crippled by a cyber warfare attack for months by malware called Shamoon. Considered the biggest hack in history in terms of cost and destructiveness . Carried out by an Iranian attacker group called Cutting Sword of Justice. Iranian hackers retaliated against Stuxnet by releasing Shamoon. The malware destroyed over 35,000 Saudi Aramco computers, affecting business operations for months.
- December 17: Computer hacker sl1nk announced that he has hacked a total of 9 countries' SCADA systems. The proof includes 6 countries: France, Norway, Russia, Spain, Sweden and the United States.

2013

- The social networking website Tumblr is attacked by hackers. Consequently, 65,469,298 unique emails and passwords were leaked from Tumblr. The data breach's legitimacy is confirmed by computer security researcher Troy Hunt.

2014

- February 7: The bitcoin exchange Mt. Gox filed for bankruptcy after $460 million was apparently stolen by hackers due to "weaknesses in [their] system" and another $27.4 million went missing from its bank accounts.
- October: The White House computer system was hacked. It was said that the FBI, the Secret Service, and other U.S. intelligence agencies categorized the attacks "among the most sophisticated attacks ever launched against U.S. government systems."
- November 24: In response to the release of the film *The Interview*, the servers of Sony Pictures are hacked by a hacker group calling itself "Guardian of Peace".
- November 28: The website of the Philippine telecommunications company Globe Telecom was hacked in response to the poor internet service they are distributing.

2015

- June: the records of 21.5 million people, including social security numbers, dates of birth, addresses, fingerprints, and security-clearance-related information, are stolen from the United States Office of Personnel Management (OPM). Most of the victims are employees of the United States government and unsuccessful applicants to it. The *Wall Street Journal* and the *Washington Post* report that government sources believe the hacker is the government of China.
- July: The servers of extramaritial affairs website Ashley Madison were breached.

2016

- February: The 2016 Bangladesh Bank heist attempted to steal US$951 million from a Bangladesh Bank, and succeeded in getting $101 million - although some of this was later recovered.
- July 22: WikiLeaks published the documents from the 2016 Democratic National Committee email leak.
- July 29: a group suspected coming from China launched hacker attacks on the website of Vietnam Airlines.
- August 13: The Shadow Brokers (TSB) started publishing several leaks containing hacking tools from the National Security Agency (NSA), including several zero-day exploits. Ongoing leaks until April 2017 (The Shadow Brokers)
- September: Hacker Ardit Ferizi is sentenced to 20 years in prison after being arrested for hacking U.S. servers and passing the leaked information to members of ISIL terrorist group back in 2015.
- October: The 2016 Dyn cyberattack is being conducted with a botnet consisting of IOTs infected with Mirai by the hacktivist groups SpainSquad, Anonymous, and New World Hackers, reportedly in retaliation for Ecuador's rescinding Internet access to WikiLeaks founder Julian Assange at their embassy in London, where he has been granted asylum.
- Late 2016: Hackers steal international personal user data from the company Uber, including phone numbers, email addresses, and names, of 57 million people and 600,000 driver's license numbers of drivers

for the company. Uber's GitHub account was accessed through Amazon's cloud-based service. Uber paid the hackers $100,000 for assurances the data was destroyed.

2017

- April: A hacker group calling itself "The Dark Overlord" posted unreleased episodes of *Orange Is the New Black* TV series online after failing to extort the online entertainment company Netflix.
- May: WannaCry ransomware attack started on Friday, 12 May 2017, and has been described as unprecedented in scale, infecting more than 230,000 computers in over 150 countries. A hacked unreleased Disney film is held for ransom, to be paid in Bitcoin.
- May: 25,000 digital photos and ID scans relating to patients of the Grozio Chirurgija cosmetic surgery clinic in Lithuania were obtained and published without consent by an unknown group demanding ransoms. Thousands of clients from more than 60 countries were affected. The breach brought attention to weaknesses in Lithuania's information security.
- June: 2017 Petya cyberattack.
- June: TRITON (TRISIS), a malware framework designed to reprogram Triconex safety instrumented systems (SIS) of industrial control systems (ICS), discovered in Saudi Arabian Petrochemical plant.
- August: Hackers demand $7.5 million in bitcoin to stop pre-releasing HBO shows and scripts, including Ballers, Room 104 and Game of Thrones.
- May–July 2017: The Equifax breach.
- September 2017: Deloitte breach.
- December: Mecklenburg County, North Carolina computer systems were hacked. They did not pay the ransom.

2018

- March: The city of Atlanta, Georgia USA computer systems are seized by hackers with ransomware. They did not pay the ransom, and two Iranians were indicted by the FBI on cyber crime charges for the breach.
- The town of Wasaga Beach in Ontario, Canada computer systems are seized by hackers with ransomware.
- October: West Haven, Connecticut USA computer systems are seized by hackers with ransomware, they paid $2,000 in ransom.
- November:
    - The first U.S. indictment of individual people for ransomware attacks occurs. The U.S. Justice Department indicted two men Faramarz Shahi Savandi and Mohammad Mehdi Shah Mansouri who allegedly used the SamSam ransomware for extortion, netting them more than $6 million in ransom payments. The companies infected with the ransomware included Allscripts, Medstar Health, and Hollywood Presbyterian Medical Center. Altogether, the attacks caused victims to lose more than $30 million, in addition to the ransom payments.

  - Marriott disclosed that its Starwood Hotel brand had been subject to a security breach.

2019

- March: Jackson County computer systems in the U.S. state of Georgia are seized by hackers with ransomware, they paid $400,000 in ransom. The city of Albany in the U.S. state of New York experiences a ransomware cyber attack.
- April: Computer systems in the city of Augusta, in the U.S. state of Maine, are seized by hackers using ransomware. The City of Greenville (North Carolina)'s computer systems are seized by hackers using ransomware known as RobbinHood. Imperial County, in the U.S. state of California, computer systems are seized by hackers using Ryuk ransomware.
- May: computer systems belonging to the City of Baltimore are seized by hackers using ransomware known as RobbinHood that encrypts files with a "file-locking" virus, as well as the tool EternalBlue.
- June: The city of Riviera Beach, Florida paid roughly $600,000 ransom in Bitcoin to hackers who seized their computers using ransomware. Hackers stole 18 hours of unreleased music from the band Radiohead demanding $150,000 ransom. Radiohead released the music to the public anyway and did not pay the ransom.

2020s

2020

- February: Anonymous hacked the United Nation's website and created a page for Taiwan, a country which has not had a seat at the UN since 1971. The hacked page featured the Flag of Taiwan, the KMT emblem, a Taiwan Independence flag, the Anonymous logo, and embedded YouTube videos such as the Taiwanese national anthem and the closing score for the 2019 film *Avengers: Endgame* titled "It's Been a Long, Long Time", along with a caption. The hacked server belonged to the United Nations Department of Economic and Social Affairs.

- May: Anonymous declared a large hacking sequence on May 28, three days after the murder of George Floyd. An individual claiming to be Anonymous stated that "We are Legion. We do not forgive. We do not forget. Expect us." in a now-deleted video. Anonymous addressed police brutality and vowed that they "will be exposing your many crimes to the world". It is suspected that Anonymous are the cause for the downtime and public suspension of the Minneapolis Police Department website and its parent site, the website of the City of Minneapolis.

- June: Anonymous claimed responsibility for stealing and leaking a trove of documents collectively nicknamed 'BlueLeaks'. The 269-gigabyte collection was then published by a leak-focused activist group known as Distributed Denial of Secrets. Furthermore the collective took down Atlanta Police Department's website via DDoS and defaced sites such as a Filipino governmental webpage and that of Brookhaven National Labs. They expressed support for Julian Assange and press freedom while briefly took a swing against Facebook, Reddit and Wikipedia for having 'engaged in shady practices behind our prying eyes'. In the case of Reddit they posted a link to a court document describing the possible involvement of a moderator of a large traffic subreddit /r/news in an online harassment related case.

- June: The Buffalo NY website was supposedly hacked by Anonymous. While the website was up and running after a few minutes, Anonymous tweeted again on Twitter urging that it be taken down. A few minutes later, the Buffalo NY website was taken down again. They also hacked Chicago police radios to play NWA's 'Fuck Tha Police'.

Timeline of computer viruses and worms

Pre-1970

- John von Neumann's article on the "Theory of self-reproducing automata" is published in 1966. The article is based on lectures given by von Neumann at the University of Illinois about the "Theory and Organization of Complicated Automata" in 1949.

1971–1975

1970 (Fiction)

- The first story written about a computer virus is *The Scarred Man* by Gregory Benford.

1971

- The Creeper system, an experimental self-replicating program, is written by Bob Thomas at BBN Technologies to test John von Neumann's theory. Creeper infected DEC PDP-10 computers running the TENEX operating system. Creeper gained access via the ARPANET and copied itself to the remote system where the message "I'm the creeper, catch me if you can!" was displayed. The Reaper program was later created to delete Creeper.

1972 (Fiction)

- The science fiction novel, *When HARLIE Was One*, by David Gerrold, contains one of the first fictional representations of a computer virus, as well as one of the first uses of the word "virus" to denote a program that infects a computer.

1973 (Fiction)

- In fiction, the 1973 Michael Crichton movie *Westworld* made an early mention of the concept of a computer virus, being a central plot theme that causes androids to run amok. Alan Oppenheimer's character summarizes the problem by stating that "...there's a clear pattern here which suggests an analogy to an infectious disease process, spreading from one...area to the next." To which the replies are stated: "Perhaps there are superficial similarities to disease" and, "I must confess I find it difficult to believe in a disease of machinery." (Crichton's earlier work, the 1969 novel *The Andromeda Strain* and 1971 film were about an extraterrestrial biological virus-like disease that threatened the human race.)

1974

- The Rabbit (or Wabbit) virus, more a fork bomb than a virus, is written. The Rabbit virus makes multiple copies of itself on a single computer (and was named "Rabbit" for the speed at which it did so) until it clogs the system, reducing system performance, before finally reaching a threshold and crashing the computer.

1975

- April: ANIMAL is written by John Walker for the UNIVAC 1108. ANIMAL asked a number of questions of the user in an attempt to guess the type of animal that the user was thinking of, while the related program PERVADE would create a copy of itself and ANIMAL in every directory to which the current user had access. It spread across the multi-user UNIVACs when users with overlapping permissions discovered the game, and to other computers when tapes were shared. The program was carefully written to avoid damage to existing file or directory structures, and not to copy itself if permissions did not exist or if damage could result. Its spread was therefore halted by an OS upgrade which changed the format of the file status tables that PERVADE used for safe copying. Though non-malicious, "Pervading Animal" represents the first Trojan "in the wild".
- The novel *The Shockwave Rider* by John Brunner is published, coining the word "worm" to describe a program that propagates itself through a computer network.

1981–1989

1981

- A program called Elk Cloner, written for Apple II systems, was created by high school student Richard Skrenta, originally as a prank. The Apple II was particularly vulnerable due to the storage of its operating system computer virus outbreak in history.

1983

- November: The term "virus" is re-coined by Frederick B. Cohen in describing self-replicating computer programs. In 1984 Cohen uses the phrase "computer virus" (suggested by his teacher Leonard Adleman) to describe the operation of such programs in terms of "infection". He defines a "virus" as "a program that can 'infect' other programs by modifying them to include a possibly evolved copy of itself." Cohen demonstrates a virus-like program on a VAX11/750 system at Lehigh University. The program could install itself in, or infect, other system objects.[*failed verification*]

1984

- August: Ken Thompson publishes his seminal paper, *Reflections on Trusting Trust*, in which he describes how he modified a C compiler so that when used to compile a specific version of the Unix operating system, it inserts a backdoor into the login command, and when used to compile a new copy of itself, it inserts the backdoor insertion code, even if neither the backdoor nor the backdoor insertion code is present in the source code of this new copy.

1986

- January: The Brain boot sector virus is released. Brain is considered the first IBM PC compatible virus, and the program responsible for the first IBM PC compatible virus epidemic. The virus is also known as Lahore, Pakistani, Pakistani Brain, and Pakistani flu as it was created in Lahore, Pakistan by 19-year-old Pakistani programmer, Basit Farooq Alvi, and his brother, Amjad Farooq Alvi.
- December: Ralf Burger presented the Virdem model of programs at a meeting of the underground Chaos Computer Club in Germany. The Virdem model represented the first programs that could replicate themselves via addition of their code to executable DOS files in COM format.

1987

- Appearance of the Vienna virus, which was subsequently neutralized – the first time this had happened on the IBM platform.
- Appearance of Lehigh virus (discovered at its namesake university), boot sector viruses such as Yale from US, Stoned from New Zealand, Ping Pong from Italy, and appearance of first self-encrypting file virus, Cascade. Lehigh was stopped on campus before it spread to the "wild" (to computers beyond the university), and has never been found elsewhere as a result. A subsequent infection of Cascade in the

offices of IBM Belgium led to IBM responding with its own antivirus product development. Prior to this, antivirus solutions developed at IBM were intended for staff use only.

- October: The Jerusalem virus, part of the (at that time unknown) Suriv family, is detected in the city of Jerusalem. The virus destroys all executable files on infected machines upon every occurrence of Friday the 13th (except Friday 13 November 1987 making its first trigger date May 13, 1988). Jerusalem caused a worldwide epidemic in 1988.
- November: The SCA virus, a boot sector virus for Amiga computers, appears. It immediately creates a pandemic virus-writer storm. A short time later, SCA releases another, considerably more destructive virus, the Byte Bandit.
- December: Christmas Tree EXEC was the first widely disruptive replicating network program, which paralyzed several international computer networks in December 1987. It was written in Rexx on the VM/CMS operating system and originated in what was then West Germany. It re-emerged in 1990.

1988

- March 1: The Ping-Pong virus (also called Boot, Bouncing Ball, Bouncing Dot, Italian, Italian-A or VeraCruz), an MS-DOS boot sector virus, is discovered at the University of Turin in Italy.
- June: The CyberAIDS and Festering Hate Apple ProDOS viruses spreads from underground pirate BBS systems and starts infecting mainstream networks. Festering Hate was the last iteration of the CyberAIDS series extending back to 1985 and 1986. Unlike the few Apple viruses that had come before which were essentially annoying, but did no damage, the Festering Hate series of viruses was extremely destructive, spreading to all system files it could find on the host computer (hard drive, floppy, and system memory) and then destroying everything when it could no longer find any uninfected files.
- November 2: The Morris worm, created by Robert Tappan Morris, infects DEC VAX and Sun machines running BSD UNIX that are connected to the Internet, and becomes the first worm to spread extensively "in the wild", and one of the first well-known programs exploiting buffer overrun vulnerabilities.
- December: The Father Christmas worm attacks DEC VAX machines running VAX/VMS that are connected to the DECnet Internet (an international scientific research network using DECnet protocols), affecting NASA and other research centers. Its purpose was to deliver a Christmas greeting to all affected users.

1989

- October: Ghostball, the first multipartite virus, is discovered by Friðrik Skúlason. It infects both executable .COM-files and boot sectors on MS-DOS systems.
- December: Several thousand floppy disks containing the AIDS Trojan, the first known ransomware, are mailed to subscribers of PC Business World magazine and a WHO AIDS conference mailing list. This

DOS Trojan lies dormant for 90 boot cycles, then encrypts all filenames on the system, displaying a notice asking for $189 to be sent to a post office box in Panama in order to receive a decryption program.

1990–1999

1990

- Mark Washburn, working on an analysis of the Vienna and Cascade viruses with Ralf Burger, develops the first family of polymorphic viruses, the Chameleon family. Chameleon series debuted with the release of 1260.
- June: The Form computer virus is isolated in Switzerland. It would remain in the wild for almost 20 years and reappear afterwards; during the 1990s it tended to be the most common virus in the wild with 20 to more than 50 percent of reported infections.

1992

- March: The Michelangelo virus was expected to create a digital apocalypse on March 6, with millions of computers having their information wiped, according to mass media hysteria surrounding the virus. Later assessments of the damage showed the aftermath to be minimal. John McAfee had been quoted by the media as saying that 5 million computers would be affected. He later said that, pressed by the interviewer to come up with a number, he had estimated a range from 5 thousand to 5 million, but the media naturally went with just the higher number.

1993

- "Leandro" or "Leandro & Kelly" and "Freddy Krueger" spread quickly due to popularity of BBS and shareware distribution.

1994

- April: OneHalf is a DOS-based polymorphic computer virus.

1995

- The first Macro virus, called "Concept", is created. It attacked Microsoft Word documents.

1996

- "Ply" – DOS 16-bit based complicated polymorphic virus appeared with built-in permutation engine.
- Boza, the first virus designed specifically for Windows 95 files arrives.

- Laroux, the first Excel macro virus appears.
- Staog, the first Linux virus attacks Linux machines

1998

- June 2: The first version of the CIH virus appears. It is the first known virus able to erase flash ROM BIOS content.

1999

- January 20: The Happy99 worm first appeared. It invisibly attaches itself to emails, displays fireworks to hide the changes being made, and wishes the user a happy New Year. It modifies system files related to Outlook Express and Internet Explorer (IE) on Windows 95 and Windows 98.
- March 26: The Melissa worm was released, targeting Microsoft Word and Outlook-based systems, and creating considerable network traffic.
- June 6: The ExploreZip worm, which destroys Microsoft Office documents, was first detected.
- September: the CTX virus is isolated
- December 30: The Kak worm is a JavaScript computer worm that spread itself by exploiting a bug in Outlook Express.

2000–2009

2000

- May 5: The ILOVEYOU worm (also known as the Love Letter, VBS, or Love Bug worm), a computer worm written in VBScript and using social engineering techniques, infects millions of Windows computers worldwide within a few hours of its release.
- June 28: The Pikachu virus is believed to be the first computer virus geared at children. It contains the character "Pikachu" from the Pokémon series. The operating systems affected by this worm are Windows 95, Windows 98, and Windows ME.

2001

- February 11: The Anna Kournikova virus hits e-mail servers hard by sending e-mail to contacts in the Microsoft Outlook addressbook. Its creator, Jan de Wit, was sentenced to 150 hours of community service.
- May 8: The Sadmind worm spreads by exploiting holes in both Sun Solaris and Microsoft IIS.
- July: The Sircam worm is released, spreading through Microsoft systems via e-mail and unprotected network shares.

- July 13: The Code Red worm attacking the Index Server ISAPI Extension in Microsoft Internet Information Services is released.
- August 4: A complete re-write of the Code Red worm, Code Red II begins aggressively spreading onto Microsoft systems, primarily in China.
- September 18: The Nimda worm is discovered and spreads through a variety of means including vulnerabilities in Microsoft Windows and backdoors left by Code Red II and Sadmind worm.
- October 26: The Klez worm is first identified. It exploits a vulnerability in Microsoft Internet Explorer and Microsoft Outlook and Outlook Express.

2002

- February 11: The Simile virus is a metamorphic computer virus written in assembly.
- Beast is a Windows-based backdoor Trojan horse, more commonly known as a RAT (Remote Administration Tool). It is capable of infecting almost all versions of Windows. Written in Delphi and released first by its author Tataye in 2002, its most current version was released October 3, 2004.
- March 7: Mylife is a computer worm that spread itself by sending malicious emails to all the contacts in Microsoft Outlook.

2003

- January 24: The SQL Slammer worm, aka *Sapphire worm*, *Helkern* and other names, attacks vulnerabilities in Microsoft SQL Server and MSDE becomes the fastest spreading worm of all time (measured by doubling time at the peak rate of growth), causing massive Internet access disruptions worldwide just fifteen minutes after infecting its first victim.
- April 2: Graybird is a trojan horse also known as Backdoor.Graybird.
- June 13: ProRat is a Turkish-made Microsoft Windows based backdoor trojan horse, more commonly known as a RAT (Remote Administration Tool).
- August 12: The Blaster worm, aka the *Lovesan* worm, rapidly spreads by exploiting a vulnerability in system services present on Windows computers.
- August 18: The Welchia (Nachi) worm is discovered. The worm tries to remove the Blaster worm and patch Windows.
- August 19: The Sobig worm (technically the Sobig.F worm) spreads rapidly through Microsoft systems via mail and network shares.
- September 18: Swen is a computer worm written in C++.
- October 24: The Sober worm is first seen on Microsoft systems and maintains its presence until 2005 with many new variants. The simultaneous attacks on network weakpoints by the Blaster and Sobig worms cause massive damage.

- November 10: Agobot is a computer worm that can spread itself by exploiting vulnerabilities on Microsoft Windows. Some of the vulnerabilities are MS03-026 and MS05-039.
- November 20: Bolgimo is a computer worm that spread itself by exploiting a buffer overflow vulnerability at Microsoft Windows DCOM RPC Interface.

2004

- January 18: Bagle is a mass-mailing worm affecting all versions of Microsoft Windows. There were 2 variants of Bagle worm, Bagle.A and Bagle.B. Bagle.B was discovered on February 17, 2004.
- Late January: The MyDoom worm emerges, and currently holds the record for the fastest-spreading mass mailer worm. The worm was most notable for performing a distributed denial-of-service (DDoS) attack on www.sco.com, which belonged to The SCO Group.
- February 16: The Netsky worm is discovered. The worm spreads by email and by copying itself to folders on the local hard drive as well as on mapped network drives if available. Many variants of the Netsky worm appeared.
- March 19: The Witty worm is a record-breaking worm in many regards. It exploited holes in several Internet Security Systems (ISS) products. It was the fastest computer issue to be categorized as a worm, and it was the first internet worm to carry a destructive payload. It spread rapidly using a pre-populated list of ground-zero hosts.
- May 1: The Sasser worm emerges by exploiting a vulnerability in the Microsoft Windows LSASS service and causes problems in networks, while removing MyDoom and Bagle variants, even interrupting business.
- June 15: Caribe or Cabir is a computer worm that is designed to infect mobile phones that run Symbian OS. It is the first computer worm that can infect mobile phones. It spread itself through Bluetooth. More information can be found on F-Secure and Symantec.
- August 16: Nuclear RAT (short for Nuclear Remote Administration Tool) is a backdoor trojan that infects Windows NT family systems (Windows 2000, Windows XP, Windows 2003).
- August 20: Vundo, or the Vundo Trojan (also known as Virtumonde or Virtumondo and sometimes referred to as MS Juan) is a trojan known to cause popups and advertising for rogue antispyware programs, and sporadically other misbehaviour including performance degradation and denial of service with some websites including Google and Facebook.
- October 12: Bifrost, also known as Bifrose, is a backdoor trojan which can infect Windows 95 through Vista. Bifrost uses the typical server, server builder, and client backdoor program configuration to allow a remote attack.
- December: Santy, the first known "webworm" is launched. It exploited a vulnerability in phpBB and used Google in order to find new targets. It infected around 40000 sites before Google filtered the search query used by the worm, preventing it from spreading.

2005

- August 2005: Zotob
- October 2005: The copy protection rootkit deliberately and surreptitiously included on music CDs sold by Sony BMG is exposed. The rootkit creates vulnerabilities on affected computers, making them susceptible to infection by worms and viruses.
- Late 2005: The Zlob Trojan, is a Trojan horse program that masquerades as a required video codec in the form of the Microsoft Windows ActiveX component. It was first detected in late 2005.

2006

- January 20: The Nyxem worm was discovered. It spread by mass-mailing. Its payload, which activates on the third of every month, starting on February 3, attempts to disable security-related and file sharing software, and destroy files of certain types, such as Microsoft Office files.
- February 16: discovery of the first-ever malware for Mac OS X, a low-threat trojan-horse known as OSX/Leap-A or OSX/Oompa-A, is announced.
- Late March: Brontok variant N was found in late March. Brontok was a mass-email worm and the origin for the worm was from Indonesia.
- June: Starbucks is a virus that infects StarOffice and OpenOffice.
- Late September: Stration or Warezov worm first discovered.
- Stuxnet

2007

- January 17: Storm Worm identified as a fast spreading email spamming threat to Microsoft systems. It begins gathering infected computers into the Storm botnet. By around June 30 it had infected 1.7 million computers, and it had compromised between 1 and 10 million computers by September. Thought to have originated from Russia, it disguises itself as a news email containing a film about bogus news stories asking you to download the attachment which it claims is a film.
- July: Zeus is a trojan that targets Microsoft Windows to steal banking information by keystroke logging.

2008

- February 17: Mocmex is a trojan, which was found in a digital photo frame in February 2008. It was the first serious computer virus on a digital photo frame. The virus was traced back to a group in China.
- March 3: Torpig, also known as Sinowal and Mebroot, is a Trojan horse that affects Windows, turning off anti-virus applications. It allows others to access the computer, modifies data, steals confidential

information (such as user passwords and other sensitive data) and installs more malware on the victim's computer.

- May 6: Rustock.C, a hitherto-rumoured spambot-type malware with advanced rootkit capabilities, was announced to have been detected on Microsoft systems and analyzed, having been in the wild and undetected since October 2007 at the very least.
- July 6: Bohmini.A is a configurable remote access tool or trojan that exploits security flaws in Adobe Flash 9.0.115 with Internet Explorer 7.0 and Firefox 2.0 under Windows XP SP2.
- July 31: The Koobface computer worm targets users of Facebook and Myspace. New variants constantly appear.
- November 21: Computer worm Conficker infects anywhere from 9 to 15 million Microsoft server systems running everything from Windows 2000 to the Windows 7 Beta. The French Navy, UK Ministry of Defence (including Royal Navy warships and submarines), Sheffield Hospital network, German Bundeswehr and Norwegian Police were all affected. Microsoft sets a bounty of US$250,000 for information leading to the capture of the worm's author(s). Five main variants of the Conficker worm are known and have been dubbed Conficker A, B, C, D and E. They were discovered 21 November 2008, 29 December 2008, 20 February 2009, 4 March 2009 and 7 April 2009, respectively. On December 16, 2008, Microsoft releases KB958644 patching the server service vulnerability responsible for the spread of Conficker.

2009

- July 4: The July 2009 cyber attacks occur and the emergence of the W32.Dozer attack the United States and South Korea.
- July 15: Symantec discovered Daprosy Worm. Said trojan worm is intended to steal online-game passwords in internet cafes. It could, in fact, intercept all keystrokes and send them to its author which makes it potentially a very dangerous worm to infect B2B (business-to-business) systems.
- August 24: Source code for MegaPanzer is released by its author under GPLv3. and appears to have been apparently detected in the wild.
- November 27: The virus called Kenzero is a virus that spreads online from peer-to-peer networks (P2P) taking browsing history.

2010–present

2010

- January: The Waledac botnet sent spam emails. In February 2010, an international group of security researchers and Microsoft took Waledac down.

- January: The Psyb0t worm is discovered. It is thought to be unique in that it can infect routers and high-speed modems.
- February 18: Microsoft announced that a BSoD problem on some Windows machines which was triggered by a batch of Patch Tuesday updates was caused by the Alureon Trojan.
- June 17: Stuxnet, a Windows Trojan, was detected. It is the first worm to attack SCADA systems. There are suggestions that it was designed to target Iranian nuclear facilities. It uses a valid certificate from Realtek.
- September 9: The virus, called "here you have" or "VBMania", is a simple Trojan horse that arrives in the inbox with the odd-but-suggestive subject line "here you have". The body reads "This is The Document I told you about, you can find it Here" or "This is The Free Download Sex Movies, you can find it Here".

2011

- SpyEye and Zeus merged code is seen. New variants attack mobile phone banking information.
- Anti-Spyware 2011, a Trojan horse that attacks Windows 9x, 2000, XP, Vista, and Windows 7, posing as an anti-spyware program. It disables security-related processes of anti-virus programs, while also blocking access to the Internet, which prevents updates.
- Summer 2011: The Morto worm attempts to propagate itself to additional computers via the Microsoft Windows Remote Desktop Protocol (RDP). Morto spreads by forcing infected systems to scan for Windows servers allowing RDP login. Once Morto finds an RDP-accessible system, it attempts to log into a domain or local system account named 'Administrator' using a number of common passwords. A detailed overview of how the worm works – along with the password dictionary Morto uses – was done by Imperva.
- July 13: the ZeroAccess rootkit (also known as Sirefef or max++) was discovered.
- September 1: Duqu is a worm thought to be related to the Stuxnet worm. The Laboratory of Cryptography and System Security (CrySyS Lab) of the Budapest University of Technology and Economics in Hungary discovered the threat, analysed the malware, and wrote a 60-page report naming the threat Duqu. Duqu gets its name from the prefix "~DQ" it gives to the names of files it creates.

2012

- May: Flame – also known as Flamer, sKyWIper, and Skywiper – a modular computer malware that attacks computers running Microsoft Windows. Used for targeted cyber espionage in Middle Eastern countries. Its discovery was announced on 28 May 2012 by MAHER Center of Iranian National Computer Emergency Response Team (CERT), Kaspersky Lab and CrySyS Lab of the Budapest University of Technology and Economics. CrySyS stated in their report that "sKyWIper is certainly the most sophisticated malware we encountered during our practice; arguably, it is the most complex malware ever found".
- August 16: Shamoon is a computer virus designed to target computers running Microsoft Windows in the energy sector. Symantec, Kaspersky Lab, and Seculert announced its discovery on August 16, 2012.

- September 20: NGRBot is a worm that uses the IRC network for file transfer, sending and receiving commands between zombie network machines and the attacker's IRC server, and monitoring and controlling network connectivity and intercept. It employs a user-mode rootkit technique to hide and steal its victim's information. This family of bot is also designed to infect HTML pages with inline frames (iframes), causing redirections, blocking victims from getting updates from security/antimalware products, and killing those services. The bot is designed to connect via a predefined IRC channel and communicate with a remote botnet.

2013

- September: The CryptoLocker Trojan horse is discovered. CryptoLocker encrypts the files on a user's hard drive, then prompts them to pay a ransom to the developer in order to receive the decryption key. In the following months, a number of copycat ransomware Trojans were also discovered.
- December: The Gameover ZeuS Trojan is discovered. This type of virus steals one's login details on popular Web sites that involve monetary transactions. It works by detecting a login page, then proceeds to inject a malicious code into the page, keystroke logging the computer user's details.
- December: Linux.Darlloz targets the Internet of things and infects routers, security cameras, set-top boxes by exploiting a PHP vulnerability.

2014

- November: The Regin Trojan horse is discovered. Regin is a dropper that is primarily spread via spoofed Web pages. Once downloaded, Regin quietly downloads extensions of itself, making it difficult to be detected via anti-virus signatures. It is suspected to have been created by the United States and United Kingdom over a period of months or years, as a tool for espionage and mass surveillance.

2015

- The BASHLITE malware is leaked leading to a massive spike in DDoS attacks.
- Linux.Wifatch is revealed to the general public. It is found to attempt to secure devices from other more malicious malware.

2016

- January: A trojan named "MEMZ" is created. The creator, Leurak, explained that the trojan was intended merely as a joke. The trojan alerts the user to the fact that it is a trojan and warns them that if they proceed, the computer may no longer be usable. It contains complex payloads that corrupt the system, displaying artifacts on the screen as it runs. Once run, the application cannot be closed without causing further damage to the computer, which will stop functioning properly regardless. When the computer is restarted, in place of the bootsplash is a message that reads "Your computer has been trashed by the MEMZ Trojan. Now enjoy the Nyan cat…", which follows with an animation of the Nyan Cat.

- February: Ransomware Locky with its over 60 derivatives spread throughout Europe and infected several million computers. At the height of the spread over five thousand computers per hour were infected in Germany alone. Although ransomware was not a new thing at the time, insufficient cyber security as well as a lack of standards in IT was responsible for the high number of infections. Unfortunately, even up to date antivirus and internet security software was unable to protect systems from early versions of Locky.
- February: Tiny Banker Trojan (Tinba) makes headlines. Since its discovery, it has been found to have infected more than two dozen major banking institutions in the United States, including TD Bank, Chase, HSBC, Wells Fargo, PNC and Bank of America. Tiny Banker Trojan uses HTTP injection to force the user's computer to believe that it is on the bank's website. This spoof page will look and function just as the real one. The user then enters their information to log on, at which point Tinba can launch the bank webpage's "incorrect login information" return, and redirect the user to the real website. This is to trick the user into thinking they had entered the wrong information and proceed as normal, although now Tinba has captured the credentials and sent them to its host.
- September: Mirai creates headlines by launching some of the most powerful and disruptive DDoS attacks seen to date by infecting the Internet of Things. Mirai ends up being used in the DDoS attack on 20 September 2016 on the Krebs on Security site which reached 620 Gbit/s. Ars Technica also reported a 1 Tbit/s attack on French web host OVH. On 21 October 2016 multiple major DDoS attacks in DNS services of DNS service provider Dyn occurred using Mirai malware installed on a large number of IoT devices, resulting in the inaccessibility of several high-profile websites such as GitHub, Twitter, Reddit, Netflix, Airbnb and many others. The attribution of the attack to the Mirai botnet was originally reported by BackConnect Inc., a security firm.

2017

- May: The WannaCry ransomware attack spreads globally. Exploits revealed in the NSA hacking toolkit leak of late 2016 were used to enable the propagation of the malware. Shortly after the news of the infections broke online, a UK cybersecurity researcher in collaboration with others found and activated a "kill switch" hidden within the ransomware, effectively halting the initial wave of its global propagation. The next day, researchers announced that they had found new variants of the malware without the kill switch.
- June: The Petya (malware) attack spreads globally affecting Windows systems. Researchers at Symantec reveal that this ransomware uses the EternalBlue exploit, similar to the one used in the WannaCry ransomware attack.
- September: The Xafecopy Trojan attacks 47 countries, affecting only Android operating systems. Kaspersky Lab identified it as a malware from the Ubsod family, stealing money through click based WAP billing systems.

- September: A new variety of Remote Access Trojan (RAT), Kedi RAT, is distributed in a Spear Phishing Campaign. The attack targeted Citrix users. The Trojan was able to evade usual system scanners. Kedi Trojan had all the characteristics of a common Remote Access Trojan and it could communicate to its Command and Control center via Gmail using common HTML, HTTP protocols.

2018

- February: Thanatos, a ransomware, becomes the first ransomware program to accept ransom payment in Bitcoin Cash.

2019

- November: Titanium is an advanced and insidious backdoor malware APT, developed by PLATINUM.

Python Datetime

Code:

```
# Import the datetime module and display the current date:

import datetime

x = datetime.datetime.now()

print(x)
```

Output on the screen:

```
2020-07-01 06:08:44.466006
```

Code:

```
# Return the year and name of weekday:

import datetime

x = datetime.datetime.now()

print(x.year)
print(x.strftime("%A"))
```

Output on the screen:

```
2020

Wednesday
```

Directive	Description	Example
%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12

%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	08
%f	Microsecond 000000-999999	548513
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018

%x	Local version of date	12/31/18
%X	Local version of time	17:41:00
%%	A % character	%

Python NumPy

Code:

```
import numpy

arr = numpy.array([1, 2, 3, 4, 5])

print(arr)
```

Output on the screen:

```
[1 2 3 4 5]
```

Code:

```
import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)
```

Output on the screen:

```
[1 2 3 4 5]
```

What will be the output of the following programs?

(a)

```
<script>

alert("Hello, world!");

</script>
```

(b)

```
<script>

var message;

message = "Hello, world!";

alert(message);

</script>
```

(c)

```
<script>
var message;
message = "Hello, world!" + "crazy world";
alert(message);

</script>
```

Android - Application Components

Which are the building blocks of android application?

The main components of the android application are:

- Activities
- Services
- Broadcast Receivers
- Content Providers
- Intent
- View
- Android Virtual Device (AVD)
- Android Emulator

Activities

If you open your phone application, you see number of activities such as received calls, dialed calls, missed calls etc.

If you click on received calls, then another activity (i.e., screen showing the list of received calls) is opened.

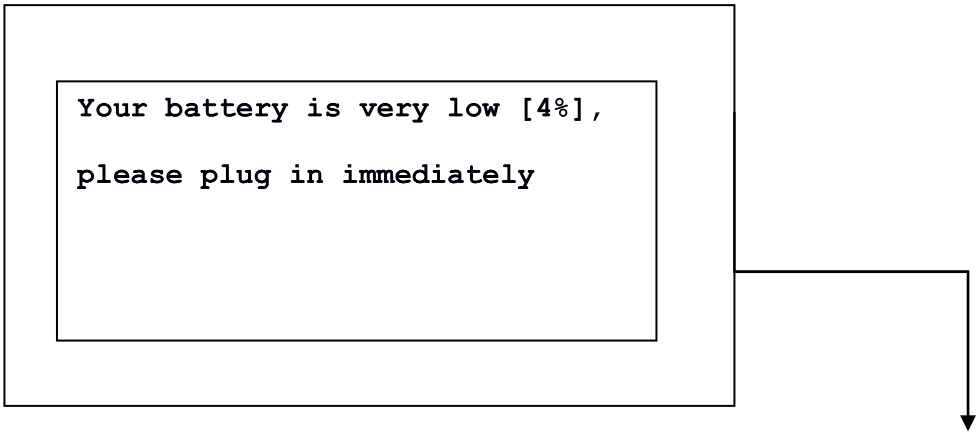
And if you click on one of the received call, then another activity showing the information about the received call (such as the phone number of received call, the time at which it was received etc.) is opened. And if you want to make a call, another activity showing the number keypad is opened.

Services

If you want the music to play in the background or if you want some video to be downloaded while you are browsing over the internet – services provide feasibility for the music to play in the background or video to be downloaded while you are browsing over internet.

Broadcast Receivers

Pop up notifications such as low battery, charging, Power got connected to the mobile device, Power got disconnected from the mobile device, A headset was plugged in, A headset was plugged out.



Your battery is very low [4%],

please plug in immediately

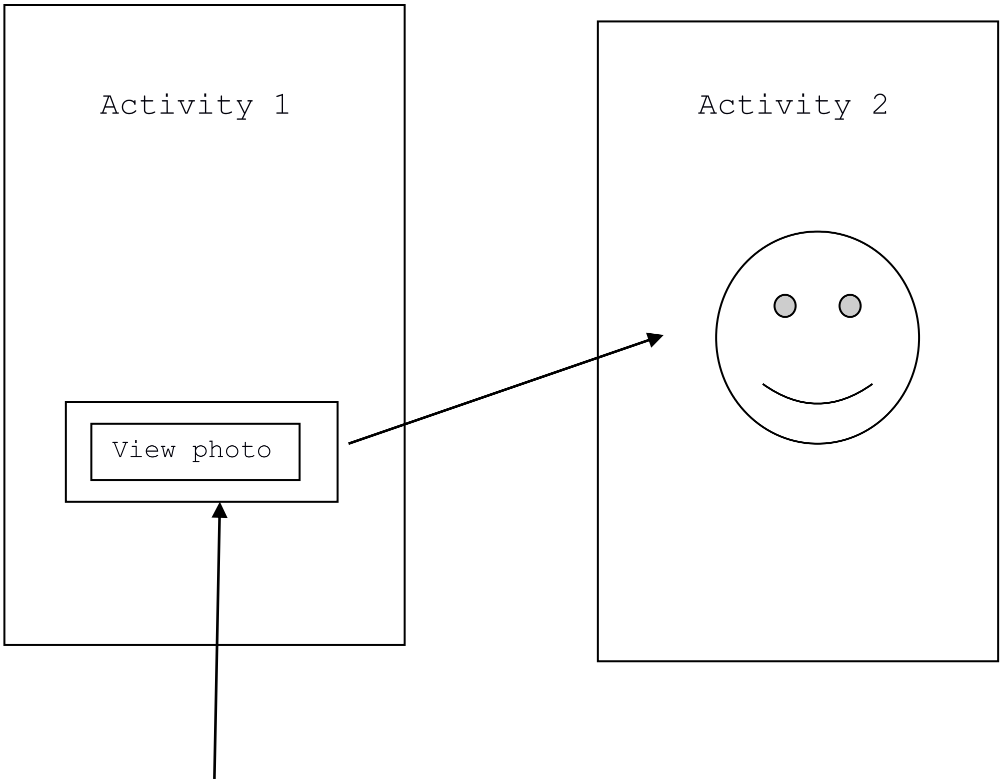
Popup notification
by broadcast
receiver

Content Providers

If you type a request for the meaning of a word in the search engine of user dictionary application

User dictionary application sends the request to content resolver and the content resolver sends the request to the content provider and the content provider fetches the information from the database and directs it to the content provider and then from content provider to content resolver and finally from content resolver to user application.

Intent



When you press view photo, intent (message) is sent to the android operating system to open another activity (i.e., activity 2) which display the photo

View (apps user interface)

Android Virtual Device (AVD) & Emulator

Different android mobile devices possess different configurations. After running and testing your android application on emulator (the component that allows the testing of android application without the necessity to install the application on a physical Android based mobile device) you need Android Virtual Device (AVD) to test whether the application is compatible with a particular android mobile device configuration before installation of the app into that mobile device.

JavaScript Number Properties

Property	Description
MAX_VALUE	Returns the largest number possible in JavaScript
MIN_VALUE	Returns the smallest number possible in JavaScript
POSITIVE_INFINITY	Represents infinity (returned on overflow)
NEGATIVE_INFINITY	Represents negative infinity (returned on overflow)
NaN	Represents a "Not-a-Number" value

Global JavaScript Methods

Method	Description
Number()	Returns a number, converted from its argument.
parseFloat()	Parses its argument and returns a floating point number
parseInt()	Parses its argument and returns an integer

Operation	Result	Same as	Result
5 & 1	1	0101 & 0001	0001
5 | 1	5	0101 | 0001	0101
~ 5	10	~0101	1010
5 << 1	10	0101 << 1	1010
5 ^ 1	4	0101 ^ 0001	0100
5 >> 1	2	0101 >> 1	0010
5 >>> 1	2	0101 >>> 1	0010

JavaScript Type Conversion Methods

Method	Description
toExponential()	Returns a string, with a number rounded and written using exponential notation.
toFixed()	Returns a string, with a number rounded and written

	with a specified number of decimals.
toPrecision()	Returns a string, with a number written with a specified length

JavaScript RegExp Reference

Modifiers

Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
g	Perform a global match (find all matches rather than stopping after the first match)
i	Perform case-insensitive matching
m	Perform multiline matching

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character NOT between the brackets

[0-9]	Find any character between the brackets (any digit)
[^0-9]	Find any character NOT between the brackets (any non-digit)
(x|y)	Find any of the alternatives specified

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	**Description**
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character

\b	Find a match at the beginning/end of a word, beginning like this: \bHI, end like this: HI\b
\B	Find a match, but not at the beginning/end of a word
\0	Find a NULL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\udddd	Find the Unicode character specified by a hexadecimal number dddd

Quantifiers

Quantifier	Description

n+	Matches any string that contains at least one *n*
n*	Matches any string that contains zero or more occurrences of *n*
n?	Matches any string that contains zero or one occurrences of *n*
n{X}	Matches any string that contains a sequence of *X* *n*'s
n{X,Y}	Matches any string that contains a sequence of X to Y *n*'s
n{X,}	Matches any string that contains a sequence of at least X *n*'s
n$	Matches any string with *n* at the end of it
^n	Matches any string with *n* at the beginning of it
?=n	Matches any string that is followed by a specific string *n*
?!n	Matches any string that is not followed by a specific string *n*

RegExp Object Properties

Property	**Description**

constructor	Returns the function that created the RegExp object's prototype
global	Checks whether the "g" modifier is set
ignoreCase	Checks whether the "i" modifier is set
lastIndex	Specifies the index at which to start the next match
multiline	Checks whether the "m" modifier is set
source	Returns the text of the RegExp pattern

RegExp Object Methods

Method	Description
compile()	Deprecated in version 1.5. Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false
toString()	Returns the string value of the regular expression

JavaScript Math Reference

Math Object Properties

Property	Description
E	Returns Euler's number (approx. 2.718)
LN2	Returns the natural logarithm of 2 (approx. 0.693)
LN10	Returns the natural logarithm of 10 (approx. 2.302)
LOG2E	Returns the base-2 logarithm of E (approx. 1.442)
LOG10E	Returns the base-10 logarithm of E (approx. 0.434)
PI	Returns PI (approx. 3.14)
SQRT1_2	Returns the square root of 1/2 (approx. 0.707)
SQRT2	Returns the square root of 2 (approx. 1.414)

Math Object Methods

Method	Description
abs(x)	Returns the absolute value of x

acos(x)	Returns the arccosine of x, in radians
acosh(x)	Returns the hyperbolic arccosine of x
asin(x)	Returns the arcsine of x, in radians
asinh(x)	Returns the hyperbolic arcsine of x
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y, x)	Returns the arctangent of the quotient of its arguments
atanh(x)	Returns the hyperbolic arctangent of x
cbrt(x)	Returns the cubic root of x
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
cosh(x)	Returns the hyperbolic cosine of x
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x, y, z,	Returns the number with the highest value

..., n)	
min(x, y, z, ..., n)	Returns the number with the lowest value
pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sinh(x)	Returns the hyperbolic sine of x
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle
tanh(x)	Returns the hyperbolic tangent of a number
trunc(x)	Returns the integer part of a number (x)

JavaScript JSON Reference

JSON Methods

Method	Description
parse()	Parses a JSON string and returns a JavaScript object
stringify()	Convert a JavaScript object to a JSON string

JavaScript Class Reference

Class Methods

Method	Description
constructor()	A special method for creating and initializing objects created within a class

Class Keywords

Keyword	Description
extends	Extends a class (inherit)
static	Defines a static method for a class
super	Refers to the parent class

JavaScript Boolean Reference

Boolean Properties

Property	Description
constructor	Returns the function that created JavaScript's Boolean prototype
prototype	Allows you to add properties and methods to the Boolean prototype

Boolean Methods

Method	Description
toString()	Converts a boolean value to a string, and returns the result
valueOf()	Returns the primitive value of a boolean

JavaScript Array Reference

Array Properties

Property	Description
constructor	Returns the function that created the Array object's prototype
length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to an Array object

Array Methods

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
copyWithin()	Copies array elements within the array, to and from specified positions
entries()	Returns a key/value pair Array Iteration Object
every()	Checks if every element in an array pass a test
fill()	Fill the elements in an array with a static value
filter()	Creates a new array with every element in an array that pass a test
find()	Returns the value of the first element in an array

	that pass a test
findIndex()	Returns the index of the first element in an array that pass a test
forEach()	Calls a function for each array element
from()	Creates an array from an object
includes()	Check if an array contains the specified element
indexOf()	Search the array for an element and returns its position
isArray()	Checks whether an object is an array
join()	Joins all elements of an array into a string
keys()	Returns a Array Iteration Object, containing the keys of the original array
lastIndexOf()	Search the array for an element, starting at the end, and returns its position
map()	Creates a new array with the result of calling a function for each array element
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length

reduce()	Reduce the values of an array to a single value (going left-to-right)
reduceRight()	Reduce the values of an array to a single value (going right-to-left)
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
some()	Checks if any of the elements in an array pass a test
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

"Simplicity is prerequisite for reliability."

— Edsger W. Dijkstra

References

- Let Us C by Yashavant Kanetkar.
- C for Dummies by Dan Gookin.
- C ++: The complete reference by Herbert Schildt.
- Programming with C by Byron S. Gottfried.
- INTRODUCTION to JAVA by Jane Meyerowitz.
- Java 2: The Complete Reference by Herbert Schildt.
- Java For Dummies by Barry Burd.
- Computer Concepts and C Programming by P.B. Kotur.
- C PROGRAMMING TUTORIAL: Simply Easy Learning by tutorialspoint.com.
- C PROGRAMMING NOTE by T K Rajan.
- An Introduction to the C Programming Language and Software Design by Tim Bailey.
- JAVA Elements: Principles of Programming in JAVA by Bailey.
- C ++: A Beginners Guide, Teach Yourself C++ by Herbert Schildt.
- C, C ++ & Java (www. w3 schools. com).
- A programming with class: A C++ introduction to computer science by Kamin.
- JAVA hand book by Naughton.
- Teach yourself JAVA by O’Neil.
- AT & T Bell laboratories: The C programmer’s hand book.
- C++ Program Design by Cohoon.
- An introduction to object oriented programming with Java by WU THOMAS.
- [Stroustrup,1994] Bjarne Stroustrup: The Design and Evolution of C++. Addison Wesley. 1994.
- [Stroustrup,1991] Bjarne Stroustrup: The C++Programming Language.AddisonWesley.1991.
- J. Gosling, B. Joy, and G. Steele. The Java Language Specification. Java Series. Sun Microsystems, 1996.
- Ashok N Kamthane, Programming and Data structures, Pearson Education.
- A programming language independent companion to Roberge/Bauer/Smith, "Engaged Learning for Programming in C++: A Laboratory Course", Jones and Bartlett Publishers, 2nd Edition, ©2001, ISBN 0763714232.

- A TUTORIAL ON POINTERS AND ARRAYS IN C by Ted Jensen.
- Pure basic A beginners guide to Computer Programming by Gary Willoughby.
- C++ for dummies by Stephen Randy Davis.
- C to Java: Converting Pointers into References by Erik D. Demaine.
- Why C++ is not just an Object Oriented Programming Language by Bjarne Stroustrup.
- [Stroustrup,1994] Bjarne Stroustrup: The Design and Evolution of C++. Addison Wesley.1994.
- Java Programming: A Practical Approach by Xavier.
- Herb Schildt's Java Programming Cookbook by Herbert Schildt.
- The Java Programming Language by Arnold.
- Computer Concepts and Programming in C by A.P. Godse, D.A. Godse .
- Programming in C by Stephen G. Kochan.
- Why learn Linux? By Candace Hazlett
- 11 Reasons Why Linux Is Better Than Windows By Ankush Das
- Linux for Dummies By Richard Blum
- Linux: The Complete Reference By Richard Petersen