Introduction to Python

Introduction:

- Using Programming languages and technologies we develop applications.
- Applications are used to store data and perform operations on data.

Types of applications:

Standalone apps:

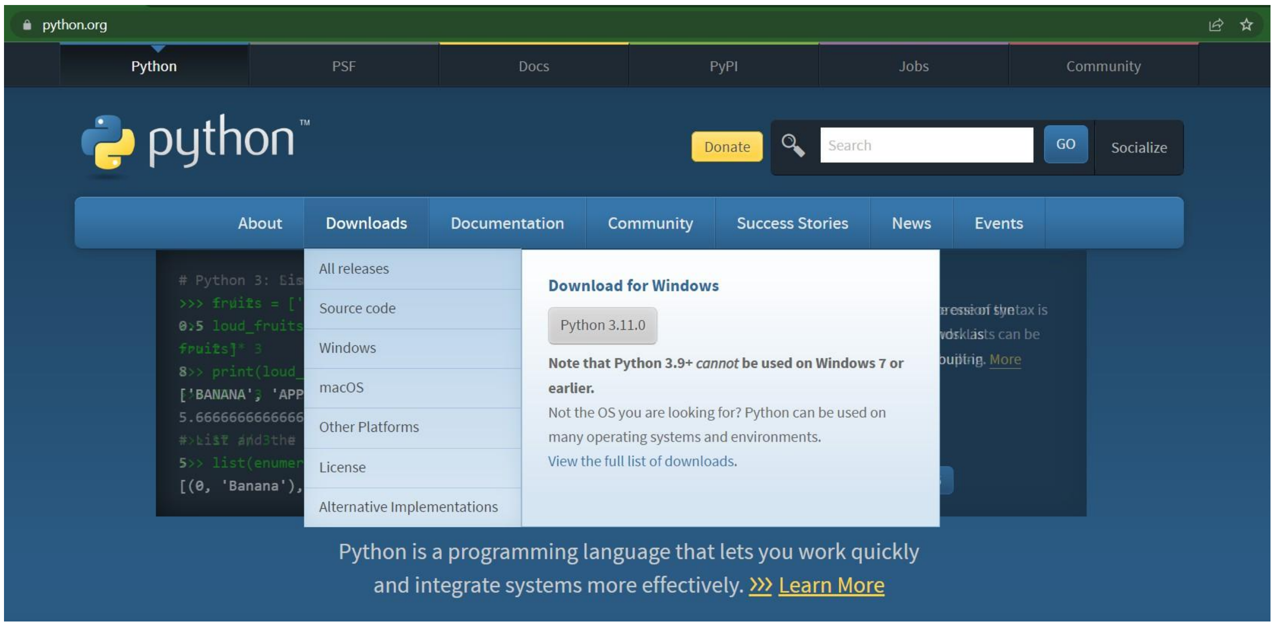
- The application runs from single machine.
- Internet connection is not required to run the application.
- Application needs to be installed on machine.
- **Examples:** VLC, MS-office, Anti-virus, Browser, **Programming languages**.

Web apps:

- The application runs from multiple machines in a network.
- Internet connection is required to run the application.
- Application installed in server and run from the clients.
- **Examples:** Gmail, YouTube, IRCTC, Flip Kart etc.

Download python:

- Python is an Open-Source Technology.
- We can download and install Python software from official website www.python.org



Download for Windows

Note that Python 3.9+ *cannot* be used on Windows 7 or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.
View the full list of downloads.

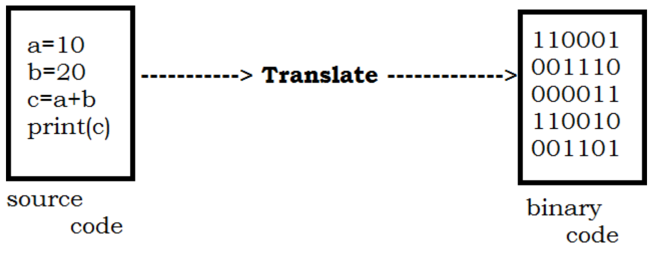
Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

Python is used to develop both Standalone and Web applications:

- Core + Advance python + GUI + DBMS = Standalone app development
- Core + Advance + DBMS + HTML + CSS + JavaScript + Django = Web app develop

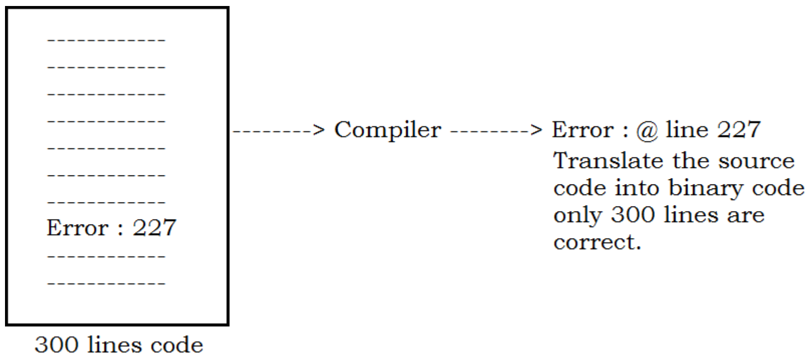
Translators:

- Programmer can define only source code.
- We need to convert the source code into binary code before run.
- We use 2 translators to convert Source code into byte code.
    - Compiler
    - Interpreter



Compiler:

- Compiler checks the source code syntactically correct or not.
- If we define the code correctly, it converts source code into byte code.
- Compiler shows error message with line number if there is a syntax error.


Error : 227
Compiler
Error : @ line 227
Translate the source code into binary code only 300 lines are correct.
300 lines code

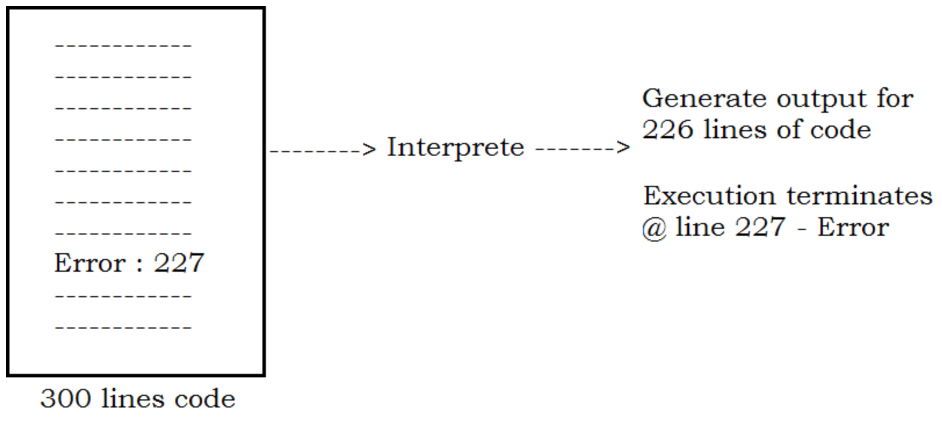
Note: Java programming language use compilation

```
class Program
{
	public static void main(String[] args)
	{
		int a=10;
		System.out.println("a val : " + a);
		int b=20;
		System.out.println("b val : " + b);
		System.out.println("c val : " + c);
	}
}
```

Compile: Error @ line – 11 (variable "c" not present)

Interpreter:

- Line by line translation of source code into binary code.
- Python uses interpreter for program execution.



Note: Python programming uses interpretation

```
a=10
print("a val :",a)
b=20
print("b val :",b)
print("c val :",c)
```

Output:

```
a val : 10
b val : 20
NameError: name 'c' is not defined
```

Python(Programming & Scripting):

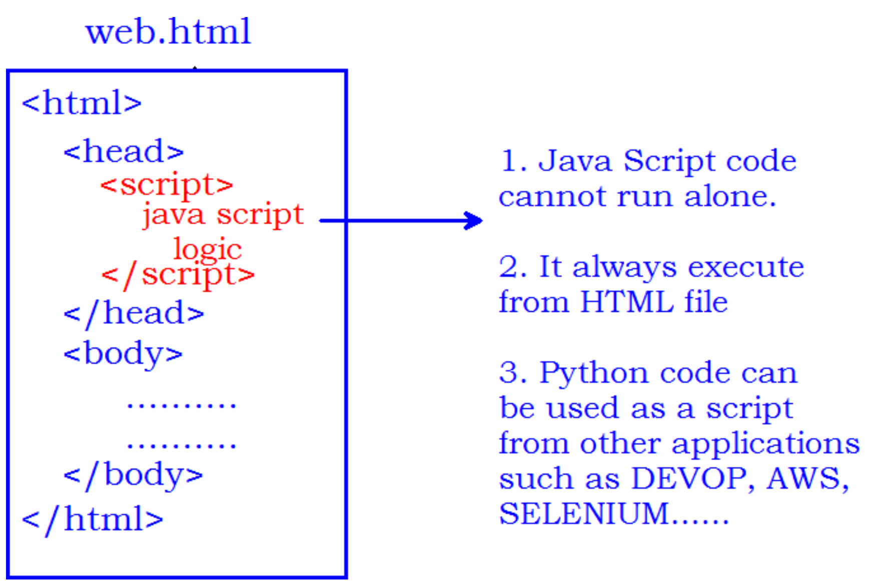
- Programming language are directly used to develop applications.
  - **Examples:** C, C++, PythonJava, .Net etc.
- Scripting languages always run from another program.
  - **Examples:** JavaScript, TypeScript, Python….

Program:

- A set of instructions.
- Program runs alone.

Script:

- Script is a set of Instructions
- Scripts is a program that always execute from another program.
- JavaScript is the best example of Scritping language.
- JavaScript code always run with HTML program.


web.html
<html>
<head>
<script>
java script
logic
</script>
</head>
<body>
....
....
</body>
</html>
1. Java Script code cannot run alone.
2. It always execute from HTML file
3. Python code can be used as a script from other applications such as DEVOP, AWS, SELENIUM......

Python is Dynamic:

- Every programming language is used to develop applications
- Application is used to store and process the data.
- Generally we allocate memory to variables in 2 ways
    1. Static memory allocation
    2. Dynamic memory allocation

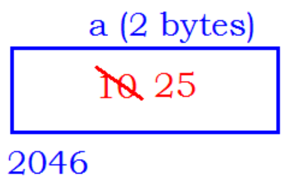
Static memory:

- Static means "fixed memory"
- The languages which are supporting primitive types allowed allocating static memory.
- Primitive variable size and type are fixed.
- Primitive variable stores data directly.
- Compiler raises error when data limit or type is deviated from specified.

Primitive : Variable stores the data

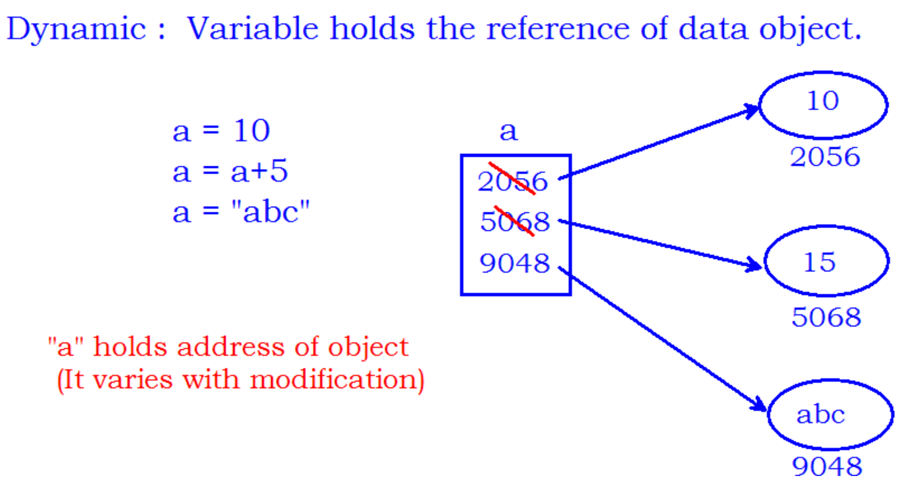
In C :

```
int  a ;
a = 10 ;
a = a+15 ;
a = 23.45 ; -> Error : only integer allowed
a = 50000 ; -> Error : Can store a value between -32768 to +32767
```



Dynamic memory:

- Python is dynamic.
- Dynamic memory means type and size of data can vary.
- Python can store information in Object format.
- Dynamic variables cannot store the data directly.
- Dynamic variables store the reference of Object which holds data.
- In Python, object location changes every time when we modify the data.


Dynamic : Variable holds the reference of data object.
a = 10
a = a+5
a = "abc"
a
~~2056~~
~~5068~~
9048
10
2056
15
5068
abc
9048
"a" holds address of object
(It varies with modification)

```
>>> a=10
>>> print(a)
10
>>> print("Address :",id(a))
Address : 1628169120

>>> a=a+15
>>> print(a)
25
>>> print("Address :",id(a))
Address : 1628169360

>>> a="python"
>>> print(a)
python
>>> print("Address :",id(a))
Address : 48576832
```

Python Variables

Variable:

- Variable is an identity of memory location.
- Variables used to store values
- You can assign any value to a variable using the "=" operator.

```
# Example:
x = 10
```

Variables can be of different types in Python, such as integer, float, string, boolean, etc.

```
# Example:
age = 25
height = 5.7
name = "Amar"
is_student = True
```

You can assign the same value to multiple variables at once using the "=" operator.

```
# Example:
x = y = z = 0
```

Variables can be updated with new values as the program runs.

```
# Example:
x = 10
x = x + 1
```

Variables can be deleted using the "del" keyword.

```
# Example:
x = 10
del x
```

Python allows you to assign values to variables in a single line

```
# Example:
x, y, z = 10, 20, 30
```

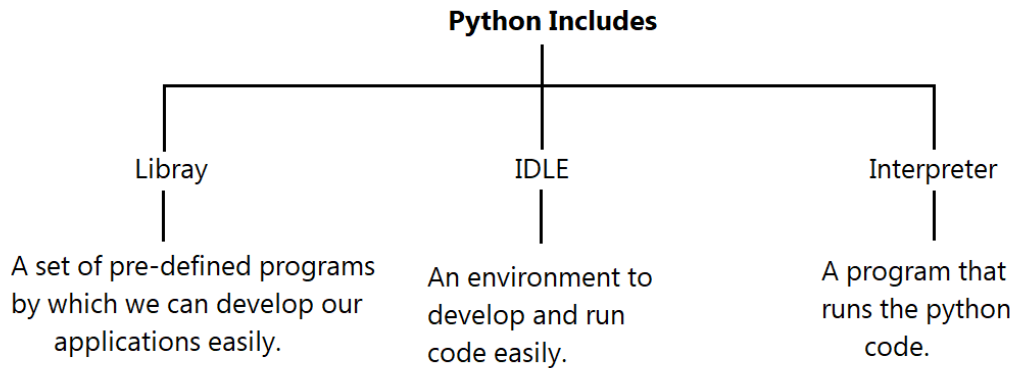
Python variables are case-sensitive, which means "a" and "A" are different variables.

```
# Example:
a = 10
A = 20
```

You can use underscores in variable names for better readability.

```
# Example:
my_variable = 10
```

Edit and Run python program:



Python Includes

Libray

A set of pre-defined programs by which we can develop our applications easily.

IDLE

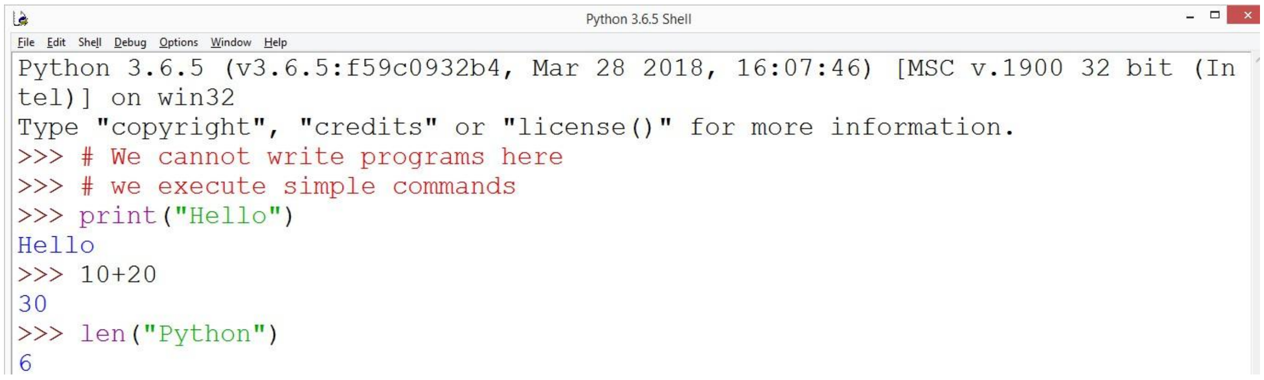
An environment to develop and run code easily.

Interpreter

A program that runs the python code.

Working with IDLE:

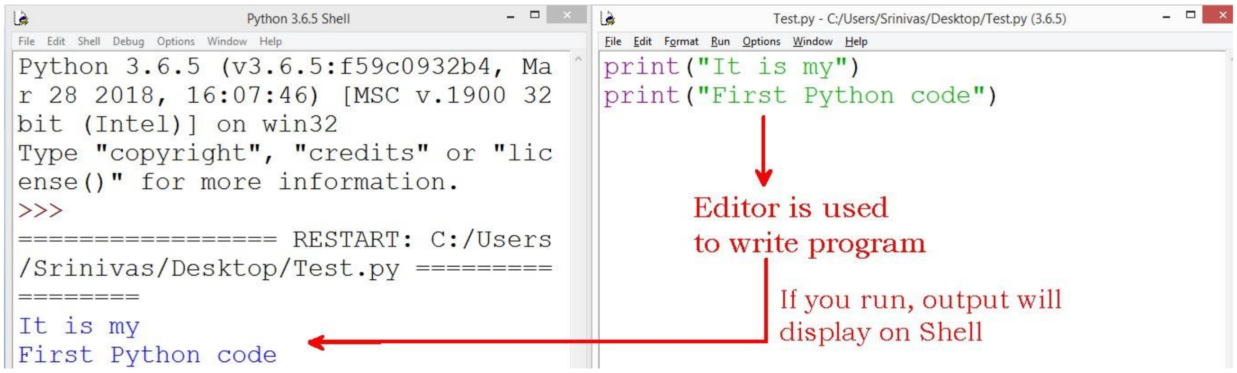
- Once python installed, we can open IDLE by searching its name.
- A shell window will be opened where we can run only commands.



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> # We cannot write programs here
>>> # we execute simple commands
>>> print("Hello")
Hello
>>> 10+20
30
>>> len("Python")
6
```

Writing and executing programs:

- We can write and execute programs from editor
- Go to File menu
- Select “new” file – Opens an editor
- Write code and Save with .py extension
- Run – with shortcut f5


Python 3.6.5 Shell
File Edit Shell Debug Options Window Help
Test.py - C:/Users/Srinivas/Desktop/Test.py (3.6.5)
File Edit Format Run Options Window Help
Editor is used to write program
If you run, output will display on Shell

```
Python 3.6.5 (v3.6.5:f59c0932b4, Ma
r 28 2018, 16:07:46) [MSC v.1900 32
bit (Intel)] on win32
Type "copyright", "credits" or "lic
ense()" for more information.
>>>
================= RESTART: C:/Users
/Srinivas/Desktop/Test.py =========
========
It is my
First Python code
```

```
print("It is my")
print("First Python code")
```

Operators and Control Statements

Operator:

- Operator is a symbol that performs operation on one or more operands.
- The member on which operator operates is called the operand.
- In the expression a = 5+9,
  a, 5,9 are operands
  =, + are operators

Python supports the following operators:

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Assignment Operators
4. Logical Operators
5. Bitwise Operators
6. Membership Operators
7. Identity Operators

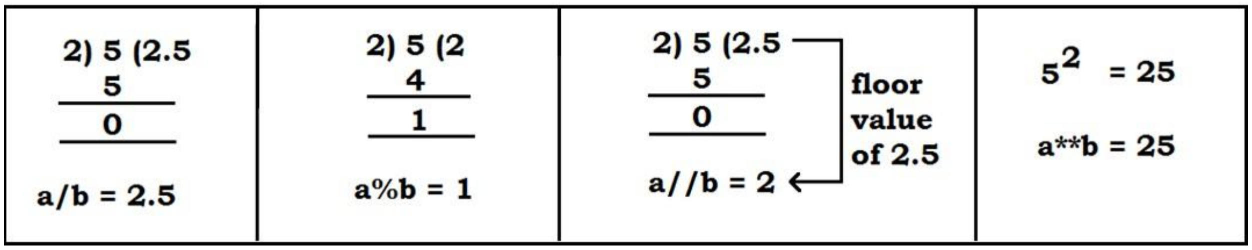
Arithmetic operators:

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication etc.

Operators are +, - , * , / , % , // , **

Operator	Meaning	Example
+	Add 2 operands	X+Y
-	Subtract right from left	X-Y
*****	Multiply 2 operands	X*Y
/	Divide left with right	X/Y
%	Divide and returns Remainder	X%Y
//	Floor division - division that results into whole number adjusted to the left in the number line	X//Y
******	Exponent - left operand raised to the power of right	X**Y

a = 5 , b = 2



Division (/) : Operator divide and returns quotient. Result is float value.
Remainder (%): It returns the remainder after division. It performs operation only on integers.

Division	Mod

>>> 5/2
2.5
 	
>>> 10/3
3.3333333333333335
 	
>>> 10/5
2.0
	>>> 5%2
	1
	>>> 10%4
	2
	>>> 5.0%2.5
	0.0
	>>> 5.0%2
	1.0

Floor division (//): Returns floor value after divide.
Exponent (**): returns the power value of specified base.

>>> 10/3
3.3333333333333335
>>> 10//3
3
>>> 10/4
2.5
>>> 10//4
2
	>>> 2**2
	4
	>>> 2**4
	16
	>>> 3**3
	27

```
print("Arithmetic operations")
print("5+2 :", 5+2)
print("5-2 :", 5-2)
print("5*2 :", 5*2)
print("5/2 :", 5/2)
print("5%2 :", 5%2)
print("5//2 :", 5//2)
print("5**2 :", 5**2)
```

Complete this work sheet Arithmetic Operators

a=10
a=20
a=30
a=40
a=50
print(a)
a ▭
	a=5
	a=a+1
	a=a+1
	a=a+1
	a=a+1
	print(a)
	a ▭

a=5
a=a+1
a=a+2
a=a+3
a=a+4
print(a)
a ▭
	a=15
	a=a+5
	a=a+4
	a=a+3
	a=a+4
	print(a)
	a ▭

a, x = 5, 1
a=a+x
x = x+1
a=a+x
x = x+1
a=a+x
print(a, x)
	a
	x
		a, b = 5, 1
		a=a+b
		b = b-1
		a=a+b
		b= b-1
		a=a+b
		print(a, b)
			a
			b

n=2;
int s=n*n;
print(s);
	n s	n=2;
		int c=n*n*n;
		print(c);
			n c

n=2;
int s=n*n;
int c=n*n*n;
print(s+c);
	s c	bal=5000;
		int amt=3500;
		bal = bal + amt;
		print(bal);
			bal amt

a=5, b=3;
int c=a+b;
print(c);
	a b c	a=5, b=3;
		a=b;
		b=a;
		print(a,b);
			a b

a=5, b=3, c;
c=a;
a=b;
b=c;
print(a,b);
	a b c	a=2, b=3;
		a=a+b;
		b=a+a;
		print(a,b);
			a b

a=2, b=3;
a=a+b;
b=a-b;
a=a-b;
print(a,b);
	a b	a=2, b=3;
		a=a*b;
		b=a//b;
		a=a//b;
		print(a,b);
			a b

n=234;
int d=n%10;
print(d);
	n d	n=234;
		int d=n//10;
		print(d);
			n d

sum=0, i=1;
sum=sum+i;
i=i+1;
sum=sum+i;
i=i+1;
sum=sum+i;
print(sum);
	i sum	fact=1, i=1;
		fact=fact*i;
		i=i+1;
		fact=fact*i;
		i=i+1;
		fact=fact*i;
		print(fact);
			i fact

n=2345, rev=0;
rev=rev*10+n%10;
n=n//10;
Print(rev, n);
rev=rev*10+n%10;
n=n//10;
Print(rev, n);
		rev=rev*10+n%10;
		n=n//10;
		Print(rev, n);
		rev=rev*10+n%10;
		n=n//10;
		Print(rev, n);	

Python Input()

Reading input from End-user :

- The input() function is used read data from user.
- The function prompts the message to enter the value
    - **input(prompt)**
- The function waits for the user to enter the value followed by pressing the "Enter" key.
- The function reads the input as string.

Reading the name and display:

```
print("Enter your name :")
name = input()
print("Hello,",name)
```

We can give the prompt while reading input

```
name = input("Enter your name : ")
print("Hello,",name)
```

Every input value will be returned in String format only.

```
print("Enter 2 numbers :")
a = input()
b = input()
c = a+b   # "5" + "6" = "56"
print("Sum :",c)
```

We need to convert the string type input values into corresponding type to perform operations.

int( ) :

- It is pre-defined function
- It can convert input value into integer type.
- On success, it returns integer value
- On failure(if the input is not valid, raised error)

Adding 2 numbers

```
print("Enter 2 numbers :")
a = input()
b = input()
c = int(a)+int(b)
print("Sum :",c)
```

Data Conversion Functions

int( ) :

- It is pre-defined function
- It can convert input value into integer type.
- On success, it returns integer value
- On failure(if the input is not valid, raised error)

```
>>> int(10)
10
>>> int(23.45)
23
>>> int(True)
1
>>> int(False)
0
>>> int("45")
45
>>> int("python") # Error : Invalid input
```

Adding 2 numbers:

```
print("Enter 2 numbers :")
a = input()
b = input()
c = int(a)+int(b)
print("Sum :",c)
```

We can give the prompt directly while calling input() function.

```
x = int(input("First Num :"))
y = int(input("Second Num :"))
print("Sum : ",x+y)
```

float() :

- converts the input value into float type.
- Raise error if the input is not valid.

```
>>> float(2.3)
2.3
>>> float(5)
5.0
>>> float(True)
1.0
>>> float("3.4")
3.4
```

```
>>> float("abc")
ValueError: could not convert string to float: 'abc'
```

bool():

- Returns a boolean value depends on input value.
- boolean values are pre-defiend (True , False)

```
>>> bool(True)
True
>>> bool(-13)
True
>>> bool(0.0013)
True
>>> bool(0)
False
>>> bool("abc")
True
>>> bool("   ")
True
>>> bool("")
False
>>> bool(False)
False
>>> bool("False")
True
```

str(): convert any input int string type.

```
>>> str(3)
'3'
>>> str(2.3)
'2.3'
>>> str(True)
'True'
```

bin(): Returns binary value for specified decimal value.

```
>>> bin(10)
'0b1010'
>>> bin(8)
'0b1000'
```

Character System:

- File is a collection of bytes.
- Every symbol occupies 1 byte memory in File.
- Every symbol stores into memory in binary format.
- Symbol converts into binary based on its ASCII value.
- Character system is the representation of all symbols of a language using constant integer values.
- Examples are ASCII and UNICODE.

ASCII : (Americans Standard Code for Information Interchange)

- Represents all symbols 1 language using constants
- The range is 0 - 255
- A language is at most having 256 symbols.
- 1 byte range is (0-255) - 2^8 value
- Hence we represent a symbol using 1 byte memory.

A-65	a-97	0-48	#-35	
B-66	b-98	1-49	$-36	
..	..		..	..
..	..		..	..
Z-90	z-122	9-57	..	

26 + 26 + 10 + 150 < 256 symbols

chr(): Return the symbol for specified integer value.
ord(): Returns the integer for specified symbol.

```
>>> chr(65)
'A'
>>> chr(50)
'2'
>>> ord('a')
97
>>> ord('$')
36
>>> ord('1')
49
```

Programs On Arithmetic Operators

Adding 2 numbers:

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))
sum = num1 + num2
print("The sum of", num1, "and", num2, "is", sum)
```

Arithmetic Operations:

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

sum = num1 + num2
difference = num1 - num2
product = num1 * num2
quotient = num1 / num2
remainder = num1 % num2
floordiv = num1 // num2

print("Sum:", sum)
print("Difference:", difference)
print("Product:", product)
print("Quotient:", quotient)
print("Remainder :", remainder)
print("Floor Division :", floordiv)
```

Program to display the last digit of given number:

```
num = int(input("Enter a number: "))
last_digit = num % 10
print("The last digit of", num, "is", last_digit)
```

Progam to remove last digit of given number:

```
num = int(input("Enter a number: "))
num = num//10
print("The number with the last digit removed is", num)
```

Find Total and Average of 4 numbers:

```
mark1 = float(input("Enter the first mark: "))
mark2 = float(input("Enter the second mark: "))
mark3 = float(input("Enter the third mark: "))
mark4 = float(input("Enter the fourth mark: "))
average = (mark1 + mark2 + mark3 + mark4) / 4
print("The average of the four marks is", average)
```

Find sum of square and cube of given number:

```
num = int(input("Enter a number: "))
square = num ** 2
cube = num ** 3
sum = square + cube
print("The sum of the square and cube of", num, "is", sum)
```

Calculate Total Salary for given basic Salary:

```
basic_salary = float(input("Enter the basic salary: "))

# Calculate the allowances and deductions
hra = 0.2 * basic_salary
da = 0.1 * basic_salary
pf = 0.05 * basic_salary

# Calculate the gross and net salary
gross_salary = basic_salary + hra + da
net_salary = gross_salary - pf

# Print the result
print("Basic salary:", basic_salary)
print("HRA:", hra)
print("DA:", da)
print("PF:", pf)
print("Gross salary:", gross_salary)
print("Net salary:", net_salary)
```

Swapping 2 numbers:

```
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Before swapping
print("Before swapping:")
print("num1 =", num1)
print("num2 =", num2)

# Swap the values
temp = num1
num1 = num2
num2 = temp

# After swapping
print("After swapping:")
print("num1 =", num1)
print("num2 =", num2)
```

Swapping 2 number without third variable:

```
# Take input from the user
num1 = float(input("Enter the first number: "))
num2 = float(input("Enter the second number: "))

# Swap the values without using a third variable
num1, num2 = num2, num1

# After swapping
print("After swapping:")
print("num1 =", num1)
print("num2 =", num2)
```

Another Way:

```
// Swap the values without using a third variable
num1 = num1 + num2;
num2 = num1 - num2;
num1 = num1 - num2;
```

Relational operators:

- Operators are > , < , >= , <= , == , !=
- These operators validate the relation among operands and return a boolean value.
- If relation is valid returns True else False

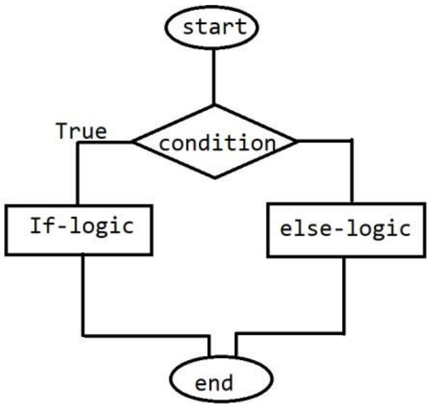
Program to understand Relational operators:

```
print("Relational operations")
print("5>3 :", 5>3)
print("5==3 :", 5==3)
print("5<3 :", 5<3)
print("5!=3 :", 5!=3)
print("5>=3 :", 5>=3)
print("5<=3 :", 5<=3)
```

If-else Conditional Statement:

```
Syntax :

    if(condition) :
        ......
        if-logic
        ......
    else:
        ......
        else-logic
        ......
```



Check the Number is Zero or Not

```
n = int(input("Enter number : "))
if(n==0):
    print("Number equals to zero")
else:
    print("Number is not zero")
```

Check the Number is Positive or Not

```
n = int(input("Enter number : "))
if(n>=0):
    print("Positive Number")
else:
    print("Negative Number")
```

Check the 2 numbers equal or not

```
n1 = int(input("Enter First num : "))
n2 = int(input("Enter Second num : "))
if(n1==n2):
    print("Equal numbers")
else:
    print("Not equal Numbers")
```

Check the first number greater than second number or not

```
n1 = int(input("Enter First num : "))
n2 = int(input("Enter Second num : "))
if(n1>n2):
    print("First Number is big")
else:
    print("Second Number is big")
```

Check the person eligible for vote or not

```
age = int(input("Enter age : "))
if(age>=18):
    print("Eligible for vote")
else:
    print("Not eligible for vote")
```

Check the number is divisible by 7 or not

```
num = int(input("Enter number : "))
if(num%7==0):
    print("Divisible by 7")
else:
    print("Not divisible by 7")
```

Check the number is even or not

```
num = int(input("Enter number : "))
if(num%2==0):
    print("Even Number")
else:
    print("Not Even")
```

Check the last digit of number is zero or not

```
num = int(input("Enter number : "))
if(num%10==0):
    print("Last digit is zero")
else:
    print("Last digit is not zero")
```

Check the sum of 2 numbers equal to 10 or not

```
n1 = int(input("Enter First number : "))
n2 = int(input("Enter Second number : "))
if(n1+n2==10):
    print("Equal to 10")
else:
    print("Not equal to 10")
```

Check last digits of given 2 numbers equal or not

```
n1 = int(input("Enter First number : "))
n2 = int(input("Enter Second number : "))
if(n1%10 == n2%10):
    print("Equal")
else:
    print("Not equal")
```

Check the average of 3 numbers greater than 60 or not

```
print("Enter 3 numbers :")
n1 = int(input())
n2 = int(input())
n3 = int(input())

if((n1+n2+n3)/3 > 60):
    print("avg Greater than 60")
else:
    print("Not")
```

Check the last digit of number is divisible by 3 or not

```
n = int(input("Enter num : "))
if((n%10)%3==0):
    print("Last digit divisible by 3")
else:
    print("Not divisible")
```

Logical operators: These operators returns True or False be validating more than one expression

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

And examples:	Or examples:	Not examples:

>>> True and True
True
>>> 5>3 and 3>2
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
>>> 5>3 and 5!=5
False
	>>> False or False
	False
	>>> False or True
	True
	>>> True or False
	True
	>>> True or True
	True
	>>> 3>5 or 5>2
	True
		>>> not True
		False
		>>> not False
		True
		>>> not 5>3
		False
		>>> not 3!=3
		True

Check the Number Divisible by 3

N%3==0 (3, 6, 9, 12….)

Check the Number Divisible by 5

N%5==0 (5, 10, 15, 20….)

Check the Number Divisible by both 3 and 5

N%3==0 and N%5==0 (15, 30, 45, 60….)

Check the Number Divisible by either 3 or 5

N%3==0 and N%5==0 (3, 5, 6, 9, 10, 12, 15…)

Program to check the Number Divisible by both 3 and 5:

```
n = int(input("enter number : "))
if n%3==0 and n%5==0:
    print("Divisible by 3 and 5")
else:
    print("Not divisible")
```

Check the person age between 20 and 50:

```
age = int(input("enter age : "))
if age>=20 and age<=50:
    print("Age between 20 and 50")
else:
    print("Not in between")
```

Check the Number is Single Digit or Not:

```
n = int(input("enter num : "))
if n>=0 and n<=9:
    print("Single Digit")
else:
    print("Not Sigle Digit")
```

Check the Number is Two Digit or Not:

```
n = int(input("enter num : "))
if n>=10 and n<=99:
    print("Two Digit")
else:
    print("Not Two Digit")
```

Check the Character is Upper case Alphabet or Not:

```
ch = input("enter character : ")
if ch>='A' and ch<='Z':
    print("Upper case Alphabet")
else:
    print("Not")
```

Check the Character is Lower case Alphabet or Not:

```
ch = input("enter character : ")
if ch>='a' and ch<='z':
    print("Lower case Alphabet")
else:
    print("Not")
```

Check the Character is Digit or Not:

```
ch = input("enter character : ")
if ch>='0' and ch<='9':
    print("Digit")
else:
    print("Not")
```

Character is Vowel or Not:

```
ch = input("enter character : ")
if ch=='a' or ch=='e' or ch=='i' or ch=='o' or ch=='u':
    print("Vowel")
else:
    print("Not")
```

Check the Character is Alphabet or Not:

```
ch = input("enter character : ")
if((ch>='A' and ch<='Z') or (ch>='a' and ch<='z')):
    print("Alphabet")
else:
    print("Not")
```

Check the Student passed in all 3 subjects or not with minimum 35 marks:

```
subj1 = int(input("Enter subj1 score: "))
subj2 = int(input("Enter subj2 score: "))
subj3 = int(input("Enter subj3 score: "))
if subj1 >= 35 and subj2 >= 35 and subj3 >= 35:
    print("Pass")
else:
    print("Fail")
```

Check A greater than both B and C:

```
print("Enter 3 numbers : ")
x = int(input())
y = int(input())
z = int(input())
if(x>y and x>z):
    print("Yes")
else:
    print("No")
```

Check given 3 numbers equal or not:

```
print("Enter 3 numbers : ")
x = int(input())
y = int(input())
z = int(input())
if(x==y and y==z and z==x):
    print("Equal numbers")
else:
    print("Not equal numbers")
```

Check given 3 numbers unique (not equal):

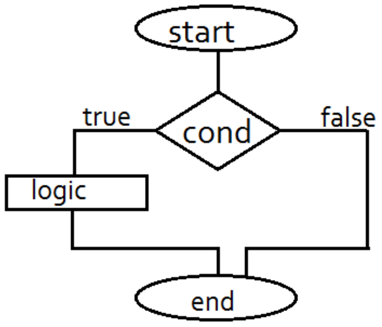
```
print("Enter 3 numbers : ")
x = int(input())
y = int(input())
z = int(input())
if(x!=y and y!=z and z!=x):
    print("Unique numbers")
else:
    print("Not unique numbers")
```

Check any 2 numbers are equal among the given 3 numbers:

```
print("Enter 3 numbers : ")
x = int(input())
y = int(input())
z = int(input())
if(x==y or y==z or z==x):
    print("Any 2 equal")
else:
    print("Not equal numbers")
```

If-block: Execute a block of instructions only if the given condition if true

```
Syntax :
    if (condition) :
        .......
        logic
        .......
```

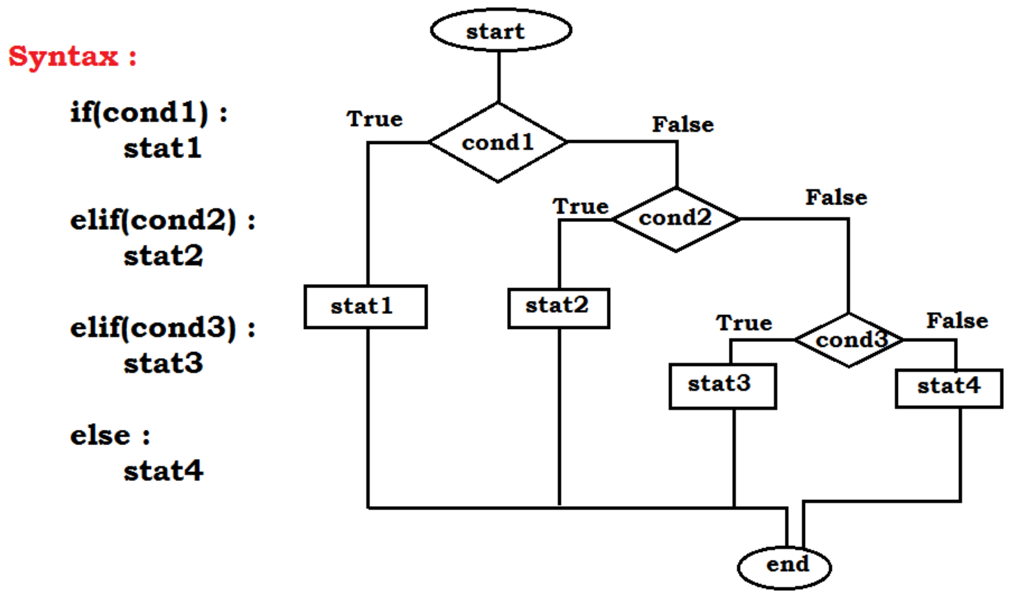


Program to give 20% discount to customer if the bill amount is > 5000

```
print("Enter bill amount :")
bill = float(input())
if(bill>5000):
    discount = 0.2*bill
    bill = bill-discount

print("Plz pay : ", bill)
```

if-elif-else: if-elif-else is a control flow structure in programming that allows a program to execute different blocks of code based on one or more conditions.



Syntax :

```
if(cond1) :
    stat1

elif(cond2) :
    stat2

elif(cond3) :
    stat3

else :
    stat4
```

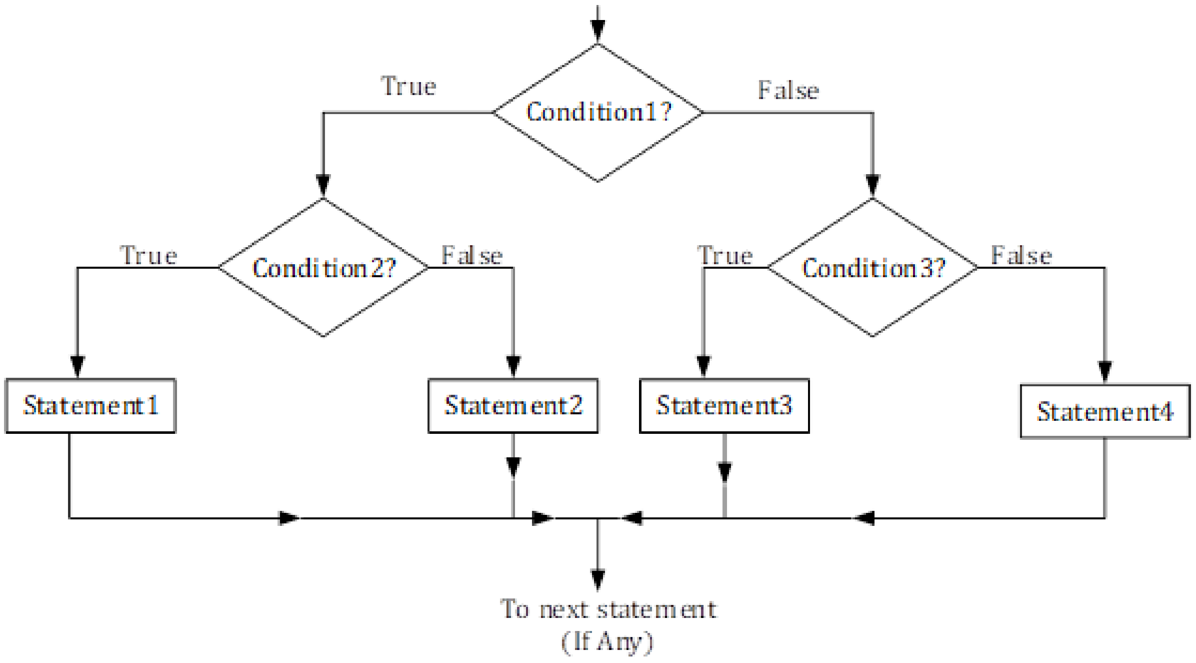
Check the Given number is Single Digit or Two Digit or Three Digit or Other

```
n = int(input("Enter number :"))
if(n>=0 and n<=9):
    print("Single digit")
elif(n>=10 and n<=99):
    print("Two digit")
elif(n>=100 and n<=999):
    print("Three digit")
else:
    print("Other digits number")
```

Check the given character is Upper case or Lower case or Digit or Symbol:

```
ch = input("Enter character :")
if(ch>='A' and ch<='Z'):
    print("Upper case alphabet")
elif(ch>='a' and ch<='z'):
    print("Lower case alphabet")
elif(ch>='0' and ch<='9'):
    print("Digit")
else:
    print("Symbol")
```

Nested-If: Writing if block inside another if block



Check the number is even or not only if the Number is positive

```
n = int(input("Enter number :"))
if n>=0:
    if n%2==0:
        print("Even number")
    else:
        print("Not even number")
else:
    print("Negative")
```

Check the biggest of 2 numbers only if the 2 numbers are not equal:

```
print("Enter 2 integers :")
a = int(input())
b = int(input())
if(a!=b):
    if(a>b):
        print("a is big")
    else:
        print("b is big")
else:
    print("equal numbers given")
```

Display Student Grade only if the Student passed in all subjects:

```
print("Enter 3 subject marks :")
m1 = int(input())
m2 = int(input())
m3 = int(input())

if(m1>=40 and m2>=40 and m3>=40):
    avg = (m1+m2+m3)/3
    if(avg>=75):
        print("Distinction")
    elif(avg>=60):
        print("A-Grade")
    elif(avg>=50):
        print("B-Grade")
    else:
        print("C-Graade")
else:
    print("Fail")
```

Bitwise operators:

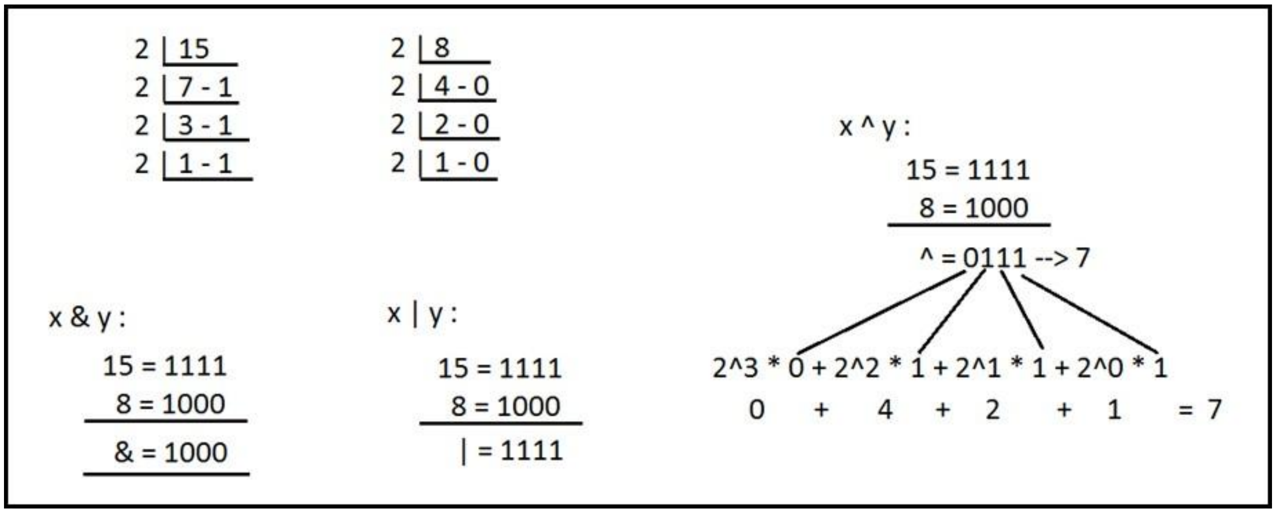
- Bitwise operators act on operands as if they were string of binary digits. It operates bit by bit, hence the name.
- For example, 2 is 10 in binary and 7 is 111.
- In the table below: Let x = 10 (0000 1010 in binary) and y = 4 (0000 0100 in binary)

Operator	Meaning	Example
&	Bitwise AND	x& y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x>> 2 = 2 (0000 0010)
<<	Bitwise left shift	x<< 2 = 40 (0010 1000)

Bitwise truth table:

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

```
>>> x=15
>>> y=8
>>> x&y
8
>>> x|y
15
>>> x^y
```



Shift operators:

- These are used to move the bits in the memory either to right side or to left side.
- Moving binary bits in the memory change the value of variable.
- These operators return the result in decimal format only.
- Operators are Right shift (>>) and Left shift (<<)

```
>>> x=8
>>> x>>2
2
>>> x<<2
32
```

Right shift: n/2^s → 8/2^2 → 8/4 → 2
Left shift : n*2^s → 8*2^2 → 8*4 → 32