

JOIN



Telegram
@PuneEngineers

For more Subjects

<https://www.studymedia.in/fe/notes>



SCAN ME



Programming and Problem Solving [110005]

Unit I-Problem Solving, Programming and Python Programming

Contents: General Problem Solving Concepts- Problem solving in everyday life, types of problems, problem solving with computers, difficulties with problem solving, problem solving aspects, top down design. Problem Solving Strategies,

Program Design Tools: Algorithms, Flowcharts and Pseudo-codes, implementation of algorithms.

Basics of Python Programming: Features of Python, History and Future of Python, Writing and executing Python program, Literal constants, variables and identifiers, Data Types, Input operation, Comments, Reserved words, Indentation, Operators and expressions, Expressions in Python.

PROBLEM FACED IN EVERYDAY IN LIFE

- ▶ People make decisions everyday
- ▶ Examples:
 - Should I wear casual or formal today?
 - Should I watch TV or go out to cinema?
 - what career?
 - what course?
 - What shoes?
 - Everything needs a DECISION AS A SOLUTION TO THE PROBLEM
- ▶ What happens when bad decisions are made?
 - WASTAGE OF TIME AND RESOURCES

SIX STEPS TO ENSURE A BEST DECISION IN PROBLEM SOLVING

1. Identify the problem
2. Understand the problem
3. Identify alternative ways to solve the problem
4. List instructions that enable you to solve the problem using selected solution
5. Select the best way to solve the problem from the list of alternative solutions
6. Evaluate the solution

WHAT MAKES A GOOD DECISION?

- Well identified problem
- All alternatives considered
- Information overloaded – appropriate alternatives
- Can the person carry out steps/instructions

TYPES OF PROBLEM

- Based on Approach of solving problem:
 - Algorithmic
 - Heuristic
- Solutions that can be solved with a series of known actions are called Algorithmic Solutions.
- Employing a self-learning approach to the solution of a problems is known as Heuristic Solutions

EXAMPLES

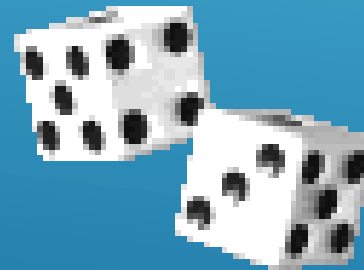
Algorithmic solution:

- ▶ To make a cup of coffee
- ▶ To find largest of three numbers



Heuristic solutions:

- ▶ how to buy the best stock?
- ▶ How to play chess?



PROBLEM SOLVING WITH COMPUTERS

Computers use algorithmic solutions

- ▶ **Solution**- set of instructions that make up solution to a problem
- ▶ **Program** – set of instructions that make up solution after coding in a computer language
- ▶ **Results** – outcome of running the program
- ▶ **Testing** – Are the outcomes what you expected and correct

People are better at developing heuristic solutions

Artificial Intelligence: Computer deals with heuristic problems

DIFFICULTIES WITH PROBLEM SOLVING

- ▶ Some have not been taught how to solve problem
- ▶ Afraid to make decision
- ▶ Complete steps inadequately:
 - ▶ Not define problem correctly
 - ▶ Not generate a sufficient list of alternatives
 - ▶ Not able to fine good alternative or list pros & cons
 - ▶ Focus on details before framework for solution is in place
- ▶ Writing Instructions : find maximum.

THE PROBLEM SOLVING ASPECTS

1. Problem definition phase

- Workout what must be done rather than how to do

2. Getting started on a problem

- Workout on implementation independent solution

3. The use of specific examples

- Use props & heuristics(i.e. rules of thumb)
- Workout the mechanism
- Ex. To find maximum in the set of no.

4. Similarities among problems

- Independently solve the problem

5. Working backwards from the solution

6. General problem solving strategies

- Divide & conquer
- Dynamic Programming

greedy search ,backtracking, branch & bound

Other Subjects: <https://www.studymedia.in/fe/notes>

Problem Solving Strategies:

Problem solving with python is a module where students learn computational thinking strategies, problem solving techniques and fundamental coding concepts using python prog.

Problem solving happens on three different levels:

Strategy: A high level idea for finding a solution.

Tactics: methods or patterns that work in many different settings.

Tools: Tricks and techniques that are used in specific situations

COMMUNICATING WITH COMPUTER

What is a program?

- ▶ A set of step-by-step instructions that directs the computer to perform tasks and produce results.

What is a Programming Language?

- ▶ A specific code language used to write instructions to perform task

Syntax:

- ▶ Rules or functions used in specific programming language for governing the computer operating system, the language and the application

BUG:

- ▶ It is an error

Debugging:

- ▶ The process of locating and correcting an error.

PROGRAM DEVELOPMENT PROCESS

- ▶ No. of phases
- ▶ Each perform well defined task
- ▶ Output of each phase is input of next phase
- ▶ Various program design tools are:
 - ▶ Algorithm
 - ▶ Flowchart
 - ▶ Pseudocodes

Phases in S/W development



Algorithm

Writing Algorithms

- ▶ After using the Interactivity chart and the IPO(Input-Process-Output) chart the next step in organising the solution is to for the programmer to develop a set of instructions for the computer – called algorithms.

Algorithm:

- ▶ **Def:** It is defined as a finite set of instructions that describe a method for solving a problem. In other words it is a step by step procedure for solving a problem.
- ▶ Systematic procedure that produces - in a finite number of steps - the answer to a question or the solution of a problem.
- ▶ Sequence of instructions which can be used to solve a given problem

Example 1:

Write an algorithm to check whether he is eligible to vote?

Step 1: Start

Step 2: Get age

Step 3: if age \geq 18 print "Eligible to vote"

Step 4: else print "Not eligible to vote"

Step 6: Stop

Example 2: (For Iteration)

Write an algorithm to print all natural numbers up to n

Step 1: Start

Step 2: get n value.

Step 3: initialize $i=1$

Step 4: if ($i \leq n$) go to step 5 else go to step 7

Step 5: Print i value and increment i value by 1

Step 6: go to step 4

Step 7: Stop

EXAMPLES

- ▶ Addition of Digits of Number
- ▶ Conversion from decimal to binary
- ▶ The process of boiling an egg
- ▶ The process of mailing a letter
- ▶ Sorting
- ▶ Searching

Let us write down the algorithm for a problem that is familiar to us.

Converting a decimal number into binary

Other Subjects: <https://www.studymedia.in/fe/notes>

ALGORITHM FOR ADDITION OF DIGITS OF NUMBER

1. Start
2. Accept the integer number
3. number mod by 10
4. Perform addition of remainder
5. number divide by 10
6. Repeat steps 2 to 4 ; keep on repeating until the number becomes zero
7. Print final result; sum of digits
8. Stop

ALGORITHM- POINTS TO NOTE

1. The process consists of repeated application of simple steps
2. All steps are unambiguous (clearly defined)
3. We are capable of doing all those steps
4. Only a limited no. of steps needs to be taken
5. Once all those steps are taken according to the prescribed sequence, the required result will be found
6. Moreover, the process will stop at that point








THREE REQUIREMENTS

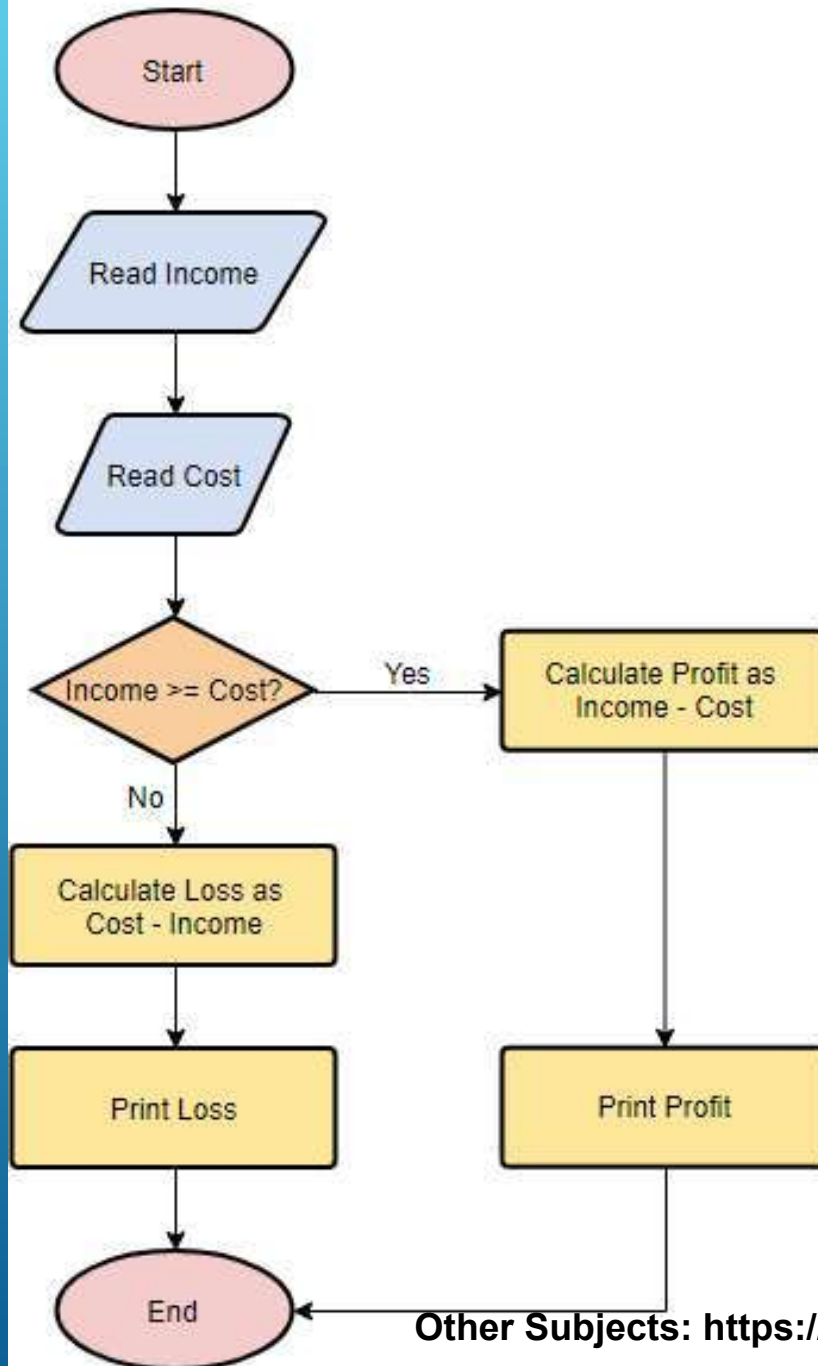
1. Sequence is:
 - a. accurate
 - b. Consists of a limited number of steps
2. Each step is:
 - a. Unambiguous
 - b. Executable
3. The sequence of steps terminates in the form of a solution

FLOWCHART

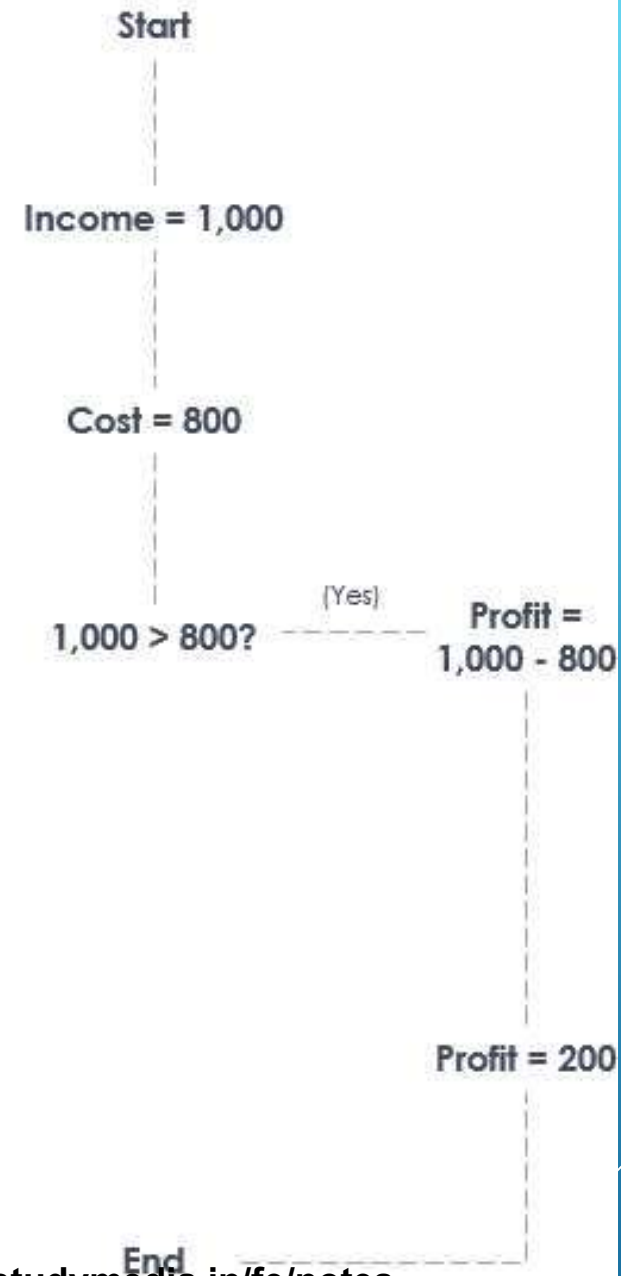
- ▶ Flowchart is a graphical representation of algorithm, in which graphic objects are used to indicate the steps & decisions that are taken as the process moves along from start to finish.
- ▶ Individual steps are represented by boxes and other shapes on the flowchart, with arrows between those shapes indicating the order in which the steps are taken.

Symbols of flowchart

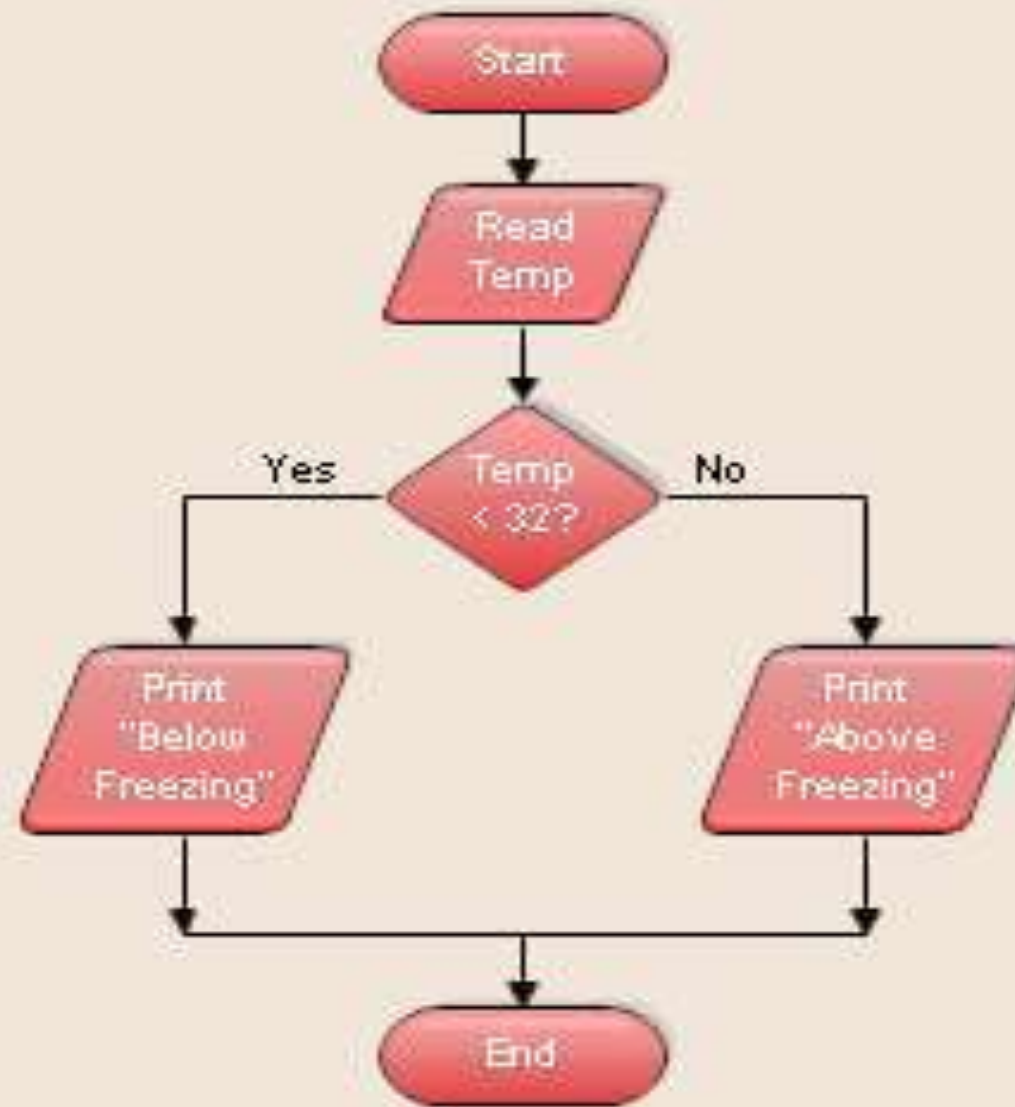
Start or stop	
Process	
Input or output	
Decision	
Flow line	
Connector	
Off-page connector	

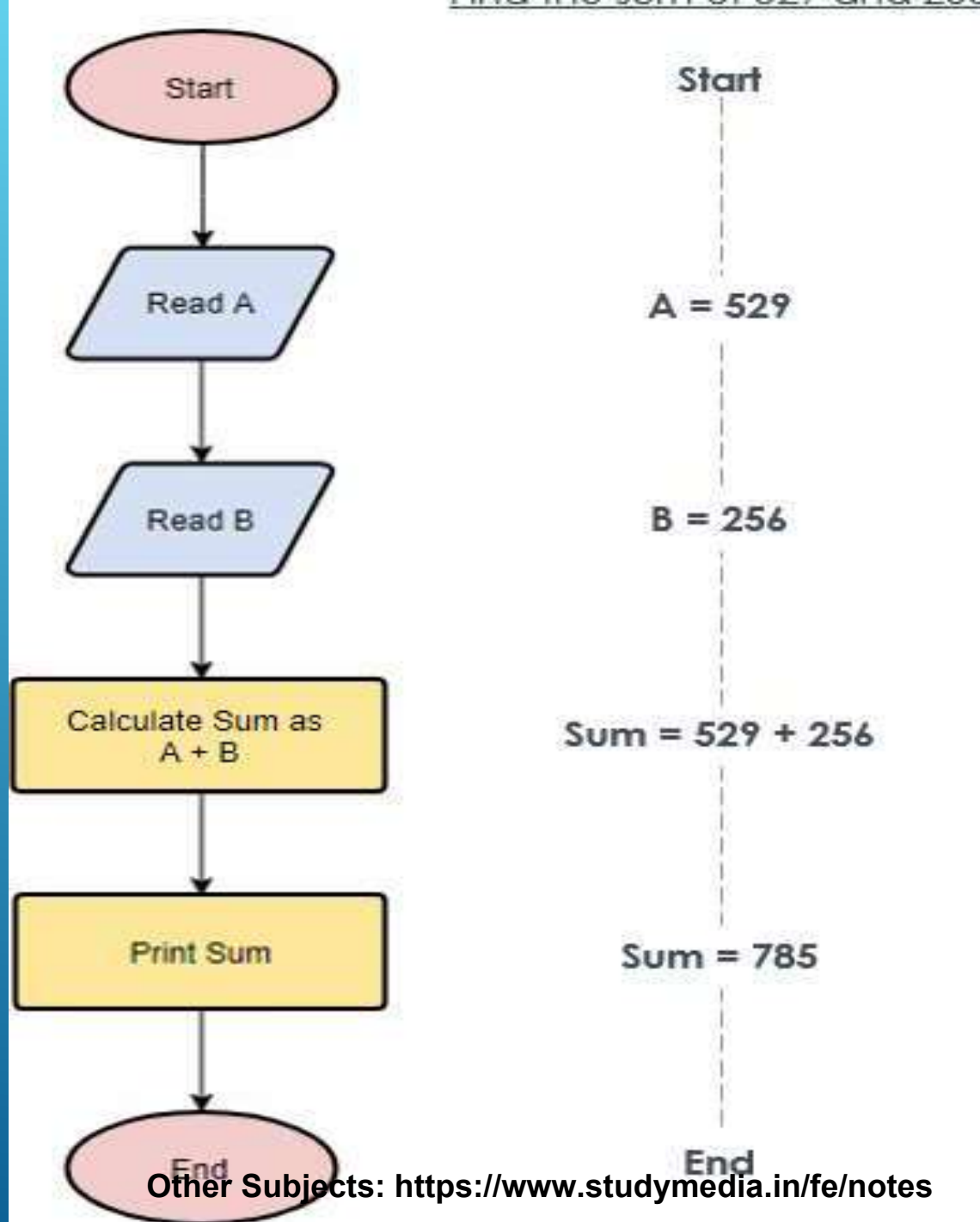


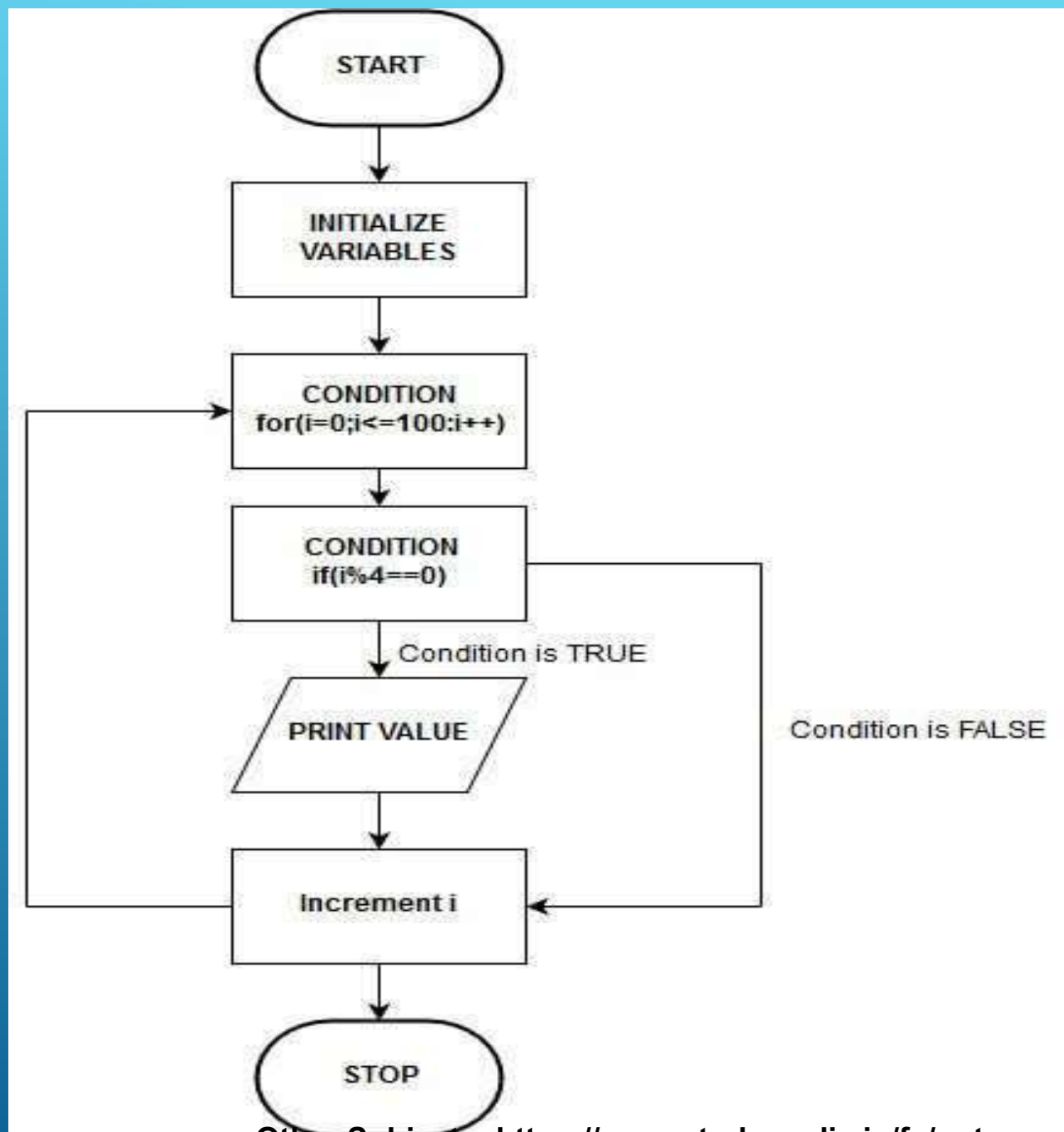
income = 1,000, cost = 800



Flowchart







DRAW FLOWCHART

1. Addition of two Nos.
2. Find maximum of three Nos.
3. Addition of Digits of No.
4. Print Color name RED, BLACK & GREEN wrt to character R or r, B or b & G or g

PSEUDO CODE

- ▶ Pseudo code is the artificial and informal language that is typically used for writing algorithms
- ▶ Similar to a programming language, but not as rigid
- ▶ It is text based algorithmic design tool.
- ▶ Pseudo code can be developed using different computer operations

COMPUTER OPERATIONS IN PSEUDO CODE

1. Input Data

- ▶ Read
 - ▶ Read student name
- ▶ Get
 - ▶ Get student name

2. Output Information

- ▶ Print
 - ▶ Print 'Program Complete'
- ▶ Write
 - ▶ Write record to file
- ▶ Output
 - ▶ Output totalAmount
- ▶ Display
 - ▶ Display 'Program Complete'

Computer operations in Pseudo code

3. Perform Arithmetic

- ▶ **Add**
 - ▶ Add num1 to num2
- ▶ **Subtract**
 - ▶ Subtract num1 from num2
- ▶ **Multiply**
 - ▶ Multiply num1 by num2
- ▶ **Divide**
 - ▶ Divide num1 by num2

4. Assign Values

- ▶ **Initialise**
 - ▶ Initialise totalPrice to zero
- ▶ **Set**
 - ▶ Set totalPrice to zero
- ▶ **Store**
 - ▶ Store zero in totalPrice

COMPUTER OPERATIONS IN PSEUDO CODE

5. Compare Values

▶ IF...THEN...ELSE

```
▶ IF num1 > num 2 THEN
    ADD num1 to toal
ELSE
    ADD num2 to total
ENDIF
```

6. Repeat Actions

▶ DOWHILE

```
▶ DOWHILE Num1 > Num2
    ADD num1 to total
    Multiply total by 3
    Subtract Num2 by 3
ENDDO
```

WRITE PSEUDO CODE

► Addition of two nos.

1. Read A
2. Read B
3. $C := A + B$
4. Print C

► Find Maximum of Three Nos.

1. Read A
2. Read B
3. Read C
4. IF $A > B$
5. THEN If $A > C$
6. THEN Print A is maximum
7. ELSE Print B is maximum
8. EndIF
9. ELSE IF $B > C$
10. THEN Print B is maximum
11. ELSE Print C is maximum
12. ENDIF
13. ENDIF

► **Addition of digits of No.**

1. **Sum := 0**
2. **Read Number**
3. **Do steps 2 to 6 WHILE
 Number > 0**
4. **Digit := Number % 10**
5. **Sum :=Sum + Digit**
6. **Number := Number / 10**
7. **Print Addition of Digits Sum**

► **Print Color name RED, BLACK & GREEN wrt
to character R or r,B or b & G or g**

1. **Read Ch**
2. **IF Ch=='R' || Ch=='r' //checking for R or r**
3. **THEN Print "RED"**
4. **ELSE IF Ch=='B' || Ch=='b'**
5. **THEN Print "BLACK"**
6. **ELSE IF Ch=='G' || Ch=='g'**
7. **THEN Print "GREEN"**
8. **ELSE Print Invalid Input**
9. **ENDIF**
10. **ENDIF**
11. **ENDIF**

Keywords used in pseudo code

- ▶ The following gives common keywords used in pseudo codes.
- ▶ 1. **//** : This keyword used to represent a comment.
- ▶ 2. **BEGIN,END**: Begin is the first statement and end is the last statement.
- ▶ 3. **INPUT, GET, READ**: The keyword is used to inputting data.
- ▶ 4. **COMPUTE, CALCULATE**: used for calculation of the result of the given expression.
- ▶ 5. **ADD, SUBTRACT, INITIALIZE** used for addition, subtraction and initialization.
- ▶ 6. **OUTPUT, PRINT, DISPLAY**: It is used to display the output of the program.
- ▶ 7. **IF, ELSE, ENDIF**: used to make decision.
- ▶ 8. **WHILE, ENDWHILE**: used for iterative statements.
- ▶ 9. **FOR, ENDFOR**: Another iterative incremented/decremented tested automatically.

10/11/2022

EXAMPLES OF PSEUDO CODE

▶ Example 1:

▶ Print 1 to n numbers

▶ BEGIN

▶ GET n

▶ INITIALIZE i=1

▶ FOR (i<=n) DO

▶ PRINT i

▶ i=i+1

▶ ENDFOR

▶ END

▶ Example 2:

▶ Addition of two numbers:

▶ BEGIN

▶ GET a,b

▶ ADD c=a+b

▶ PRINT c

▶ END

▶ TOP DOWN DESIGN

- ▶ Top down design approach in stepwise fashion:
 - ▶ Breaking a problem into sub problems
 - ▶ Choice of suitable data structures
 - ▶ Construction of loops
 - ▶ Establishing initial conditions for loops
 - ▶ Finding iterative construct
 - ▶ Termination of loops

TOP-DOWN DESIGN

Top-Down Design

Problem-solving technique in which the problem is divided into Subproblems; the process is applied to each subproblem.

Modules

Self-contained collection of steps, that solve a problem or subproblem.

Abstract Step

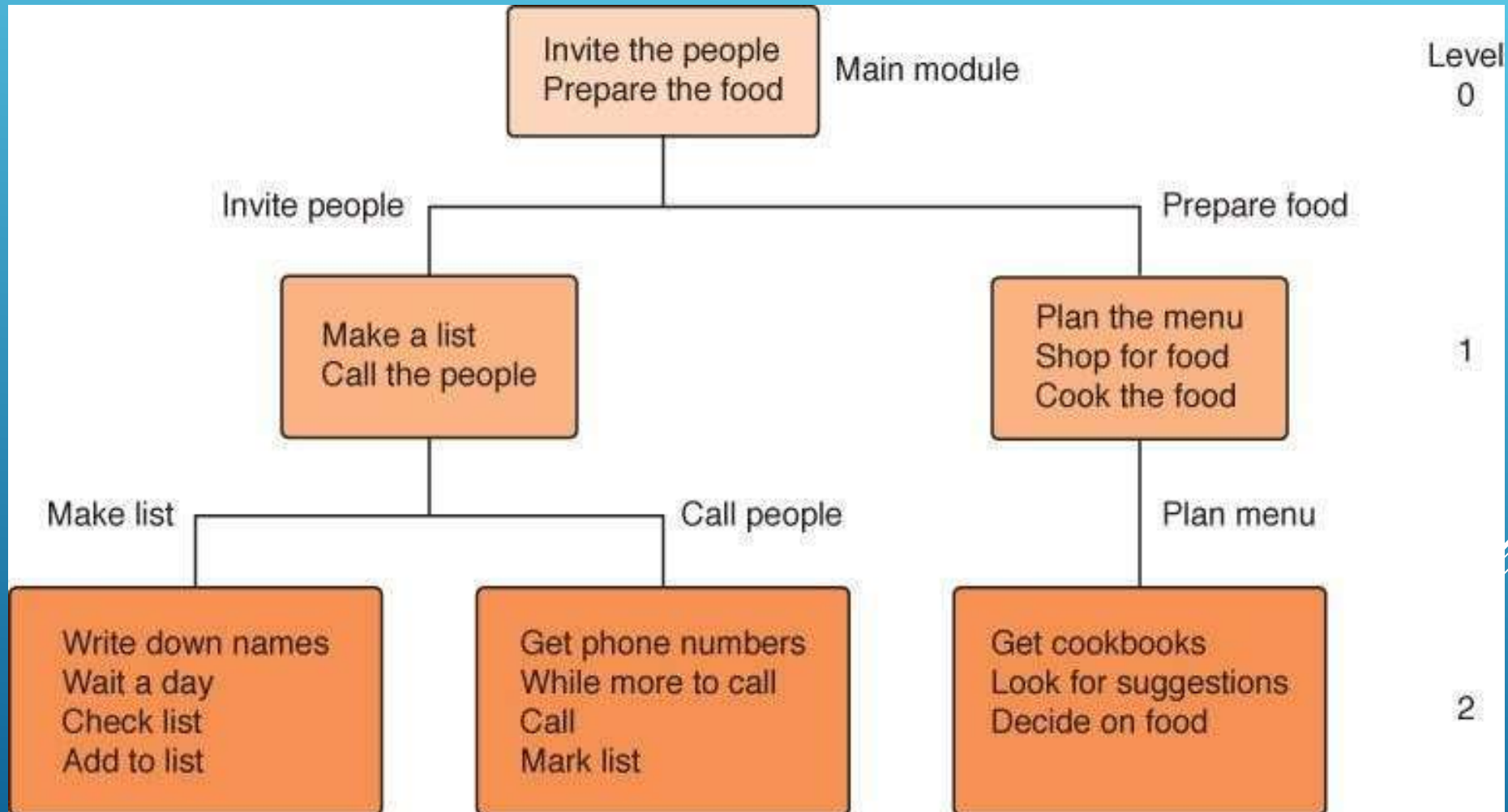
An algorithmic step containing unspecified details.

Concrete Step

An algorithm step in which all details are specified

A General Example

Planning a large party



HISTORY OF PYTHON

- Python was **developed by** Guido van Rossum in the late 1980 and early 1990 at the National Research Institute for Mathematics and Computer Science in the Netherlands.


- **Python is derived** from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages.

Python is a **general-purpose interpreted, interactive, object-oriented, and high-level programming language.**



PROGRAMMING BASICS

- **code or source code**: The sequence of instructions in a program.
- **syntax**: The set of legal structures and commands that can be used in a particular programming language.
- **Compiler**-is a program which takes a high level programming language and translates that language to a machine level programming language.
- **Interpreter**-An interpreter is a program which you interact with, and you feed the instructions in your language, in this case python.
- **output**: The messages printed to the user by a program.
- **console**: The text box onto which output is printed.
 - Some source code editors pop up the console as an external window, and others contain their own console window.

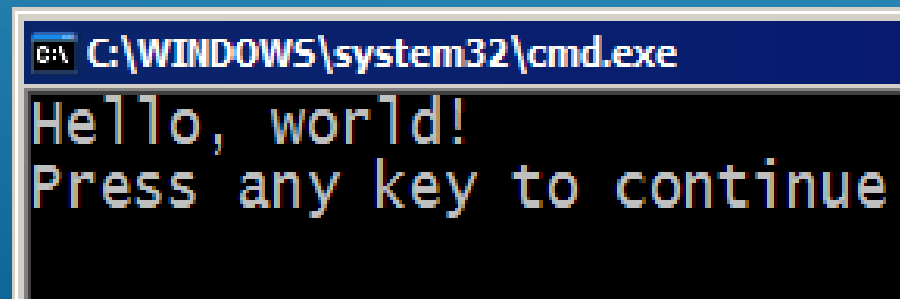


The image shows a screenshot of a Python Shell window and an editor window. The Python Shell window, titled "Python Shell", displays the Python 2.5.1 version information and a copyright notice. It also shows the output of a script: "Hello, world!" and "4". The editor window, titled "example.py - C:/Docum...", shows the source code of the script: `print "Hello, world!"` and `print 2 + 2`. The status bar of the editor window indicates the current line and column: "Ln: 2 Col: 11".

```
Python 2.5.1 (r251:54863, Apr 18 2006) [AMD64]
win32
Type "copyright", "credits" or "license()" for more

>>>
>>>
Hello, world!
4
>>>
```

```
print "Hello, world!"
print 2 + 2
```



The image shows a screenshot of a Windows command prompt window. The title bar indicates the path "C:\WINDOWS\system32\cmd.exe". The prompt displays the output of the script: "Hello, world!" and "Press any key to continue".

```
C:\WINDOWS\system32\cmd.exe
Hello, world!
Press any key to continue
```

FEATURES OF PYTHON

► Simple

Python is a simple and minimalistic language.

Reading a good Python program feels almost like reading English.

► Easy to Learn

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

► Free and Open Source

Python is an example of FLOSS (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs.

FEATURES OF PYTHON

► High-level Language

When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

► Portable

Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

FEATURES OF PYTHON

► Interpreted

Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code.

Internally, Python converts the source code into an intermediate form called byte codes and then translates this into the MACHINE language of your computer and then runs it.

All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc.

This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

FEATURES OF PYTHON

► Object Oriented

Python supports procedure-oriented programming as well as object-oriented programming.

In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs.

In *object-oriented* languages, the program is built around objects which combine data and functionality.

► Extensible

If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.

FEATURES OF PYTHON

► Embeddable

You can embed Python within your C/C++ programs to give scripting capabilities for y program's users.

► Extensive Libraries

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC(**Extensible Markup Language remote procedure call**), HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python.

► So there is huge scope for *python* in the *future* ahead.

► You can rely on it for a better career as a programmer/software developer.

INSTALLATION OF PYTHON ON WINDOWS

we are selecting the version 3.7 of python for installation .

First go to the authentic web site of Python <https://www.python.org>

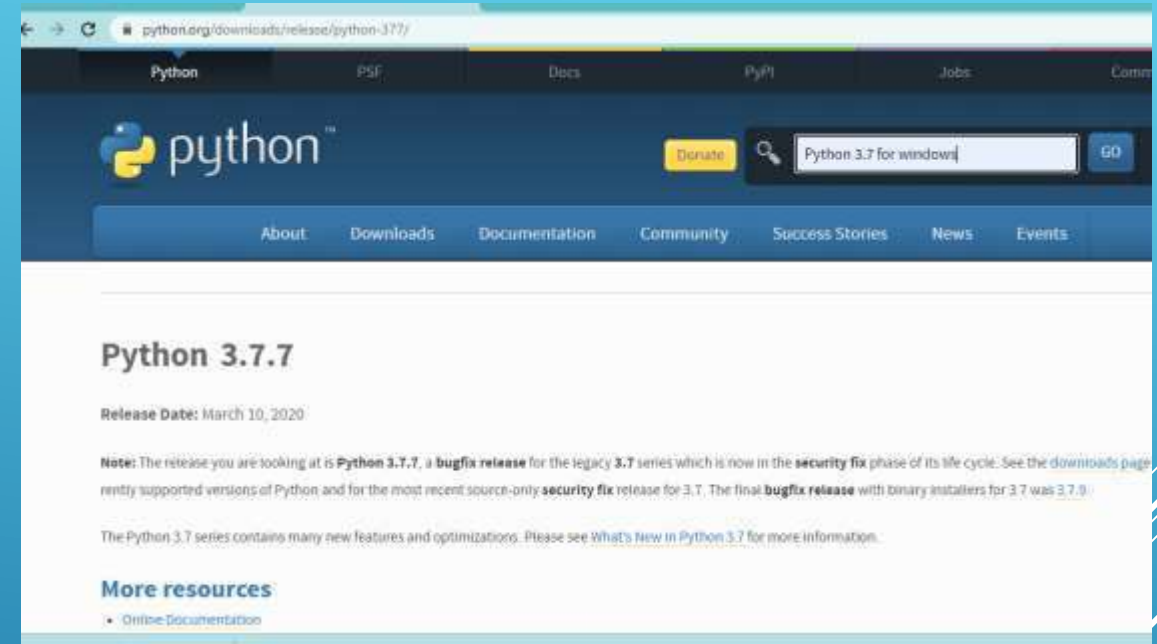
Click on **Downloads** then Search for **Python 3.7 for windows**

Scroll down and see the **table of files**.

From this click on the **Windows x86-64 executable installer file**

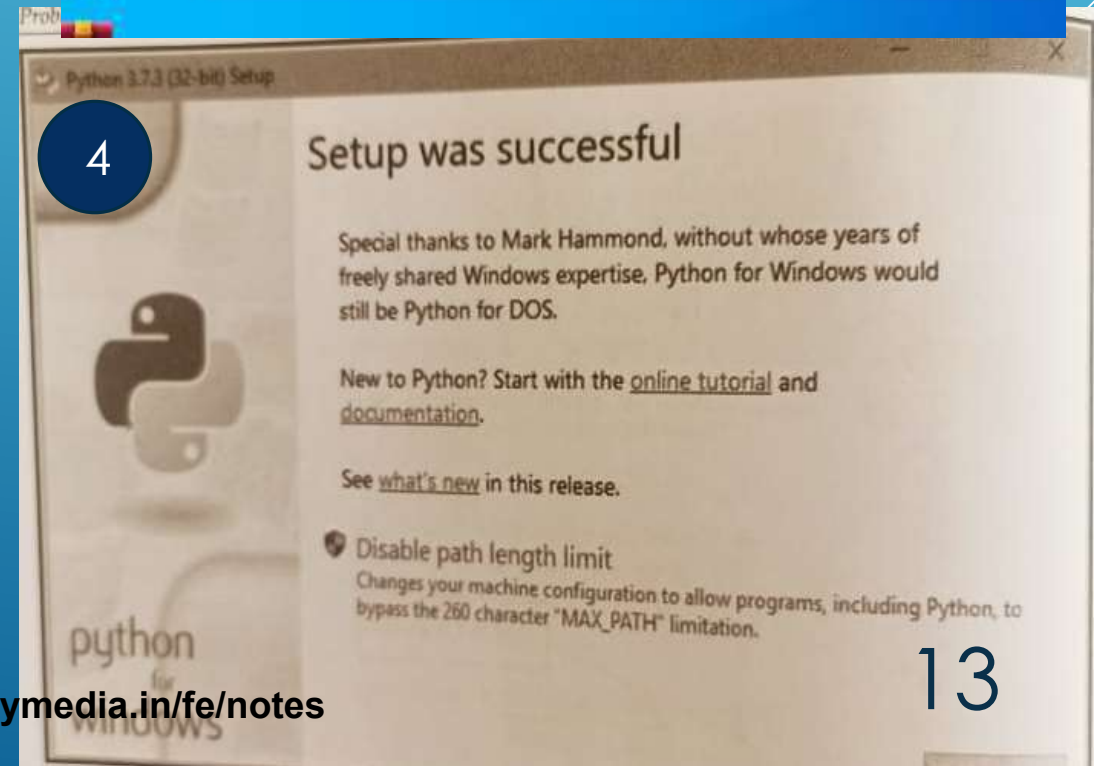
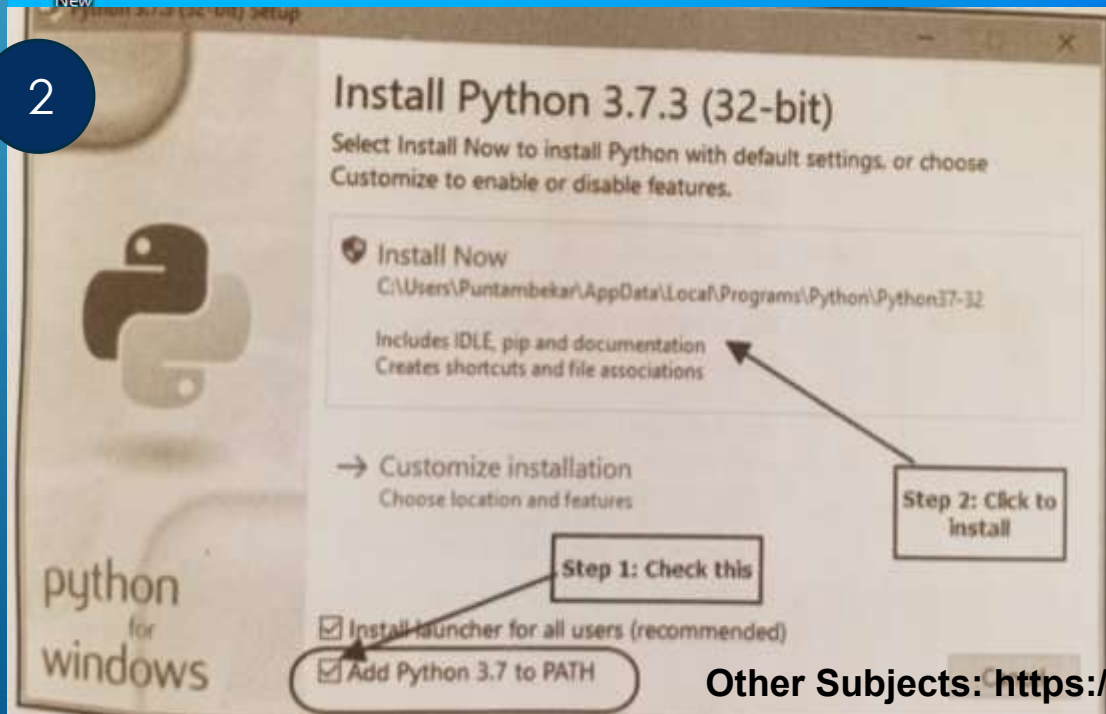
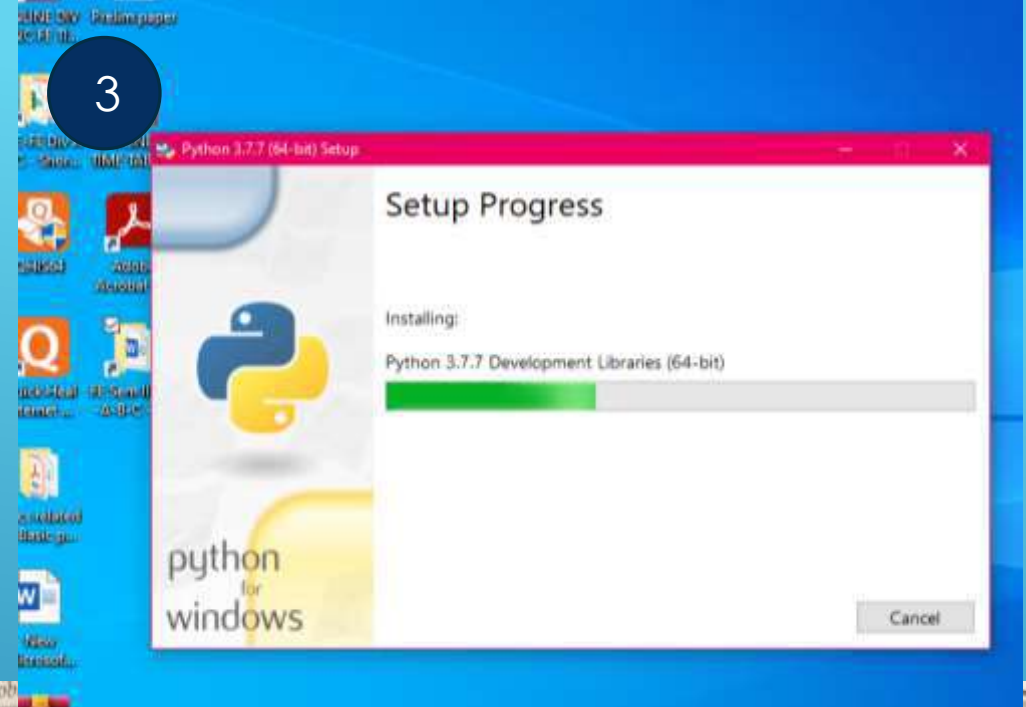
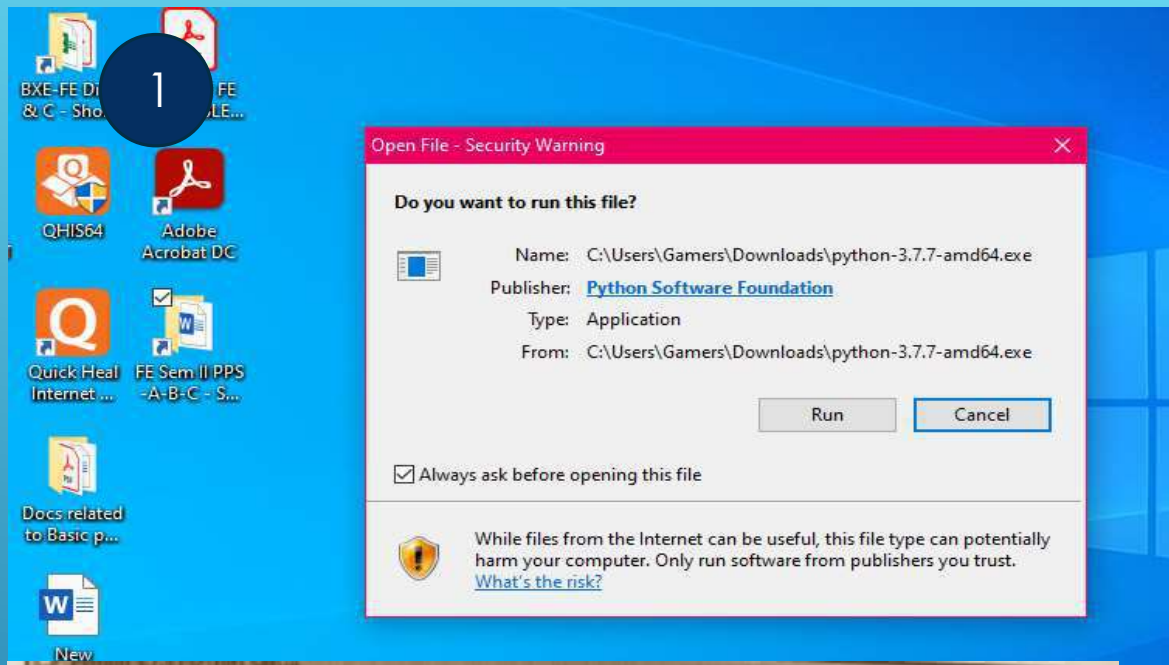
An **.exe** file will be get downloaded

Run that file on your PC or Laptop by double clicking on that



Version	Operating System	Description	MD5 Sum	File Size	GPG
Gzipped source tarball	Source release		d348d978a5307512fbc7d7d52dd3a5ef	23161893	SIG
XZ compressed source tarball	Source release		172c650156f7bea68ce31b2fd01fa766	17268888	SIG
macOS 64-bit installer	macOS	for OS X 10.9 and later	47b06433e242c8eb848e035965a860ac	29163525	SIG
Windows help file	Windows		89bb2ea8c5838bd2612de600bd301d32	8183265	SIG
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64	6aa3b1c327561bda256f2deebf038dc9	7444654	SIG
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64	e0c910087459d778d827eb1554489663	797616	SIG
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64	d1609dad50247738d8ac2479e4cde4af	1348896	SIG
Windows x86-64 embeddable zip file	Windows		29672b400490ea21995c6dbae4c4e1c8	6614968	SIG
Windows x86-64 executable installer	Windows		#9d7b/c43b4f2472d75a0553808719e5	25747128	SIG

Other Subjects: <https://www.studymedia.in/fe/notes>



INSTALL PYTHON ON ANDROID

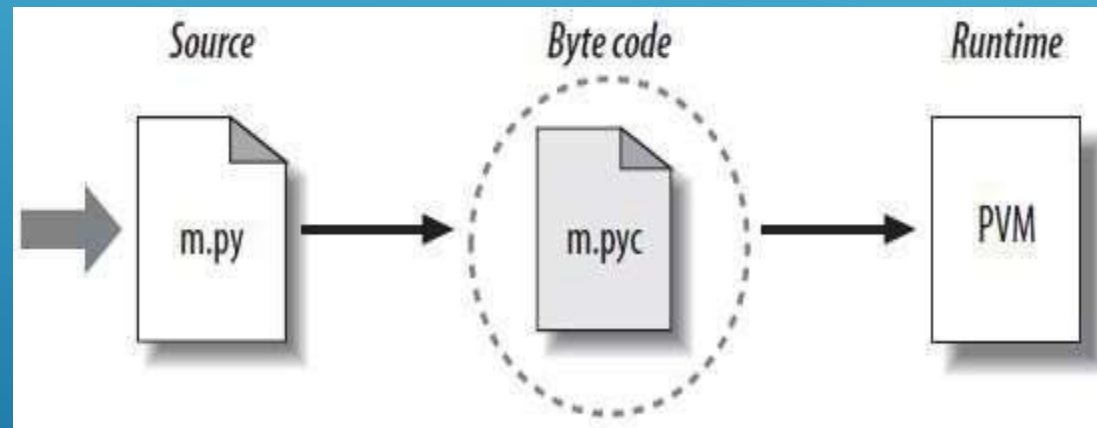
► Download Pydroid 3 – IDE for Python 3 app from Play store

1. To install Pydroid app go to play store link here – Pydroid 3 – IDE for Python 3.
2. After installation is complete, Run the app and it will show as installing python.
3. Wait for a minute and it will show the ide. ...
4. Click on the yellow button to run the code.

► <https://www.youtube.com/watch?v=nz-z-cbgZb18>

PYTHON CODE EXECUTION

Python's traditional runtime execution model: source code you type is translated to **byte code**, which is then run by the **Python Virtual Machine**. Your code is automatically compiled, but then it is interpreted.



Source code extension is **.py**

Byte code extension is **.pyc (compiled python code)**

YOUR FIRST PROGRAM

- To develop the Python program ,click on the **File** and select **NewFile**.
- This will open a **new text editor** where you can write your first program.

Prints the words Hello Python

print("Hello Python")

print("Its nice learning Python")

print("Python is easy to learn")

MODES OF WORKING IN PYTHON

Python has two modes of working

Normal(or Script) mode-In this mode python command are typed in a text file and after completion the file is saved with extension **.py** and then complete file is executed to get the results.

Interactive mode- Command line shell which immediately gives result for each statement. It runs the current command and also previously fed command .The new line fed to interpreter can be evaluated in part and in whole.

VARIABLES AND IDENTIFIERS

- ▶ Variable Definition: Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
- ▶ The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable

MULTIPLE ASSIGNMENT

- ▶ Python allows you to assign a single value to several variables simultaneously.

For example –

- ▶ `a = b = c = 1`
- ▶ `a,b,c = 1,2,"john"`

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume).

Rules for Python variables:

- ▶ A variable name must start with a letter or the underscore character
- ▶ A variable name cannot start with a number
- ▶ A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- ▶ Variable names are case-sensitive (age, Age and AGE are three different variables)

IDENTIFIERS

- ▶ An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another

Rules for writing identifiers

- ▶ Identifiers can be a combination of letters in lowercase (**a to z**) or uppercase (**A to Z**) or digits (**0 to 9**) or an underscore `_`. Names like `myClass`, `var_1` and `print_this_to_screen`, all are valid example.
- ▶ An identifier cannot start with a digit. `1variable` is invalid, but `variable1` is a valid name.
- ▶ Keywords cannot be used as identifiers.

Ex. `global = 1`

Output

File "<interactive input>", line 1

`global = 1`

`^ SyntaxError: invalid syntax`

- ▶ We cannot use special symbols like `!`, `@`, `#`, `$`, `%` etc. in our identifier.

Ex. `a@ = 0`

(o/p syntax error)

- ▶ Multiple words can be separated using an underscore, like `this_is_a_long_variable`

Literals in Python

Literals is a raw data given in a variable or constants

String Literals

A string literal is a sequence of characters surrounded by quotes. We can use both single, double or triple quotes for a string.

Numeric literals are immutable i.e. unchangeable.

Numeric literals can belong to different types of number

Boolean literal- It has two values True and False

Special literal-Special literal in python is None

Constants- It is a type of variable whose value is unchanged Ex.- $\pi=3.14$

PYTHON- DATA TYPES

- 1.Numeric
- 2.String
- 3.List
- 4.Tuple
- 5.Directory

Numeric- This data type has different forms as listed below

int (Signed Integers)-can be both positive and negative

long (Long Integers)- integers of unlimited size

Float – Float number (Ex- 3.123)

Complex- Number having magnitude and imaginary value (3+j4)

Python List : List is an ordered sequence of items.

➤ e. g. `list = [10,20,30,'thirty']`

Python Tuple : Tuple is an ordered sequence of items same as list. The only difference is that tuples are immutable. Tuples once created cannot be modified

➤ e. g. `tuple = (1,2,'Red', 'green')`

Python Strings : String is collection of characters . To define the string we can use single, double, triple quote. Two operators like + and * can used with string.

Ex- `Str1= 'I love programming.'`

Python Dictionary is an unordered collection of key-value pairs.

➤ It is like an associative array or a hash table where each key stores a specific value.

➤ `dict= {1:'orange','grapes':2, 'Bananas':3, 'apple':4}`

➤ `print("Complete Dictionary:", dict)`

Other Subjects: <https://www.studymedia.in/fe/notes>

Input Operation

- Input Operation :
- In Python the input() function is used to take input from the user.
- It prompts for the user input and reads a line. After reading data, it converts it into a string and returns that. It throws an error EOFError if EOF is read.

Program to check input
type in Python

```
num = input ("Enter number :")  
print(num)  
name1 = input("Enter name : ")  
print(name1)
```

```
# Printing type of input value  
print ("type of number",  
type(num))  
print ("type of name",  
type(name1))
```

```
Enter number :123
```

```
123
```

```
Enter name : geeksforgeeks
```

```
geeksforgeeks
```

```
type of number <class 'str'>
```

```
type of name <class 'str'>
```

```
>>>
```

Comments

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python
```

```
# First comment
```

You can comment multiple lines as follows –

```
# The python code for multiplication
```

```
#Name :Rohit Patil
```

```
# Roll no. : 2021CS 001
```

Indentation

Programming languages like C, C++, Java use braces { } to define a block of code. Python uses indentation.

- A code block starts with indentation and ends with the first unindented line. The amount of indentation is up to you, but it must be consistent throughout that block.
- ▶ Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.
- ▶
- ▶ The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

Indentation

Python program showing
indentation

```
age =int(input('enter your age:'))  
  
if age < 18:  
    print('not eligible for voting')  
else:  
    print('eligible for voting')
```


RESERVED WORDS

- ▶ The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

- ▶ `>>> import keyword`

- ▶ `>>> print(keyword.kwlist)`

output: ['False', 'None', 'True', 'and', 'as', 'assert', 'async',
'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else',
'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is',
'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try',
'while', 'with', 'yield']

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Operators and Expressions

- Python provides a variety of operators described as follows :
- Arithmetic operators
- Comparison operators
- Assignment operators
- Logical operators
- Bitwise operators
- Membership operators
- Identity operators

Arithmetic Operator

Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
*	Multiplication	$x * y$
/	Division	x / y
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$

Examples of Arithmetic Operator

$a = 9$

$b = 4$

Addition of numbers

$add = a + b$

Subtraction of numbers

$sub = a - b$

Multiplication of number

$mul = a * b$

Division(float) of number

$div1 = a / b$

Division(floor) of number

$div2 = a // b$

Modulo of both number

$mod = a \% b$

Power

$p = a ** b$

Comparison Operator

Comparison operators are used to comparing the value of the two operands and returns Boolean true or false accordingly.

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

```
# Examples of Relational Operators
```

```
a = 13
```

```
b = 33
```

```
# a > b is False
```

```
print(a > b)
```

```
# a < b is True
```

```
print(a < b)
```

```
# a == b is False
```

```
print(a == b)
```

```
# a != b is True
```

```
print(a != b)
```

```
# a >= b is False
```

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

Examples of Assignment Operators

```
a = 10
```

Assign value

```
b = a
```

```
print(b)
```

Add and assign value

```
b += a
```

```
print(b)
```

Subtract and assign value

```
b -= a
```

```
print(b)
```

multiply and assign

```
b *= a
```

```
print(b)
```

LOGICAL OPERATORS

Python Logical Operators

There are following logical operators supported by Python language. Assume variable a holds 10 and variable b holds 20 then

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

```
# Examples of Logical Operator
```

```
a = True
```

```
b = False
```

```
# Print a and b is False
```

```
print(a and b)
```

```
# Print a or b is True
```

```
print(a or b)
```

```
# Print not a is False
```

```
print(not a)
```

BITWISE OPERATOR

Python Bitwise Operators

Bitwise operator works on bits and performs bit by bit operation. Assume if a = 60; and b = 13; Now in binary format they will be as follows –

a = 0011 1100

b = 0000 1101

a&b = 0000 1100 # bitwise logical ANDing

a|b = 0100 1001 # logical OR

a^b = 0011 0001 # logical XOR

~a = 1100 0011 # 1's complement

```
# Print bitwise AND operation
print(a & b)
```

```
# Print bitwise OR operation
print(a | b)
```

```
# Print bitwise NOT operation
print(~a)
```

```
# print bitwise XOR operation
print(a ^ b)
```

```
# print bitwise right shift
operation
print(a >> 2)
```

```
# print bitwise left shift operation
print(a << 2)
```


Bitwise Operators

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
Binary OR	It copies a bit if it exists in either operand.	(a b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

MEMBERSHIP OPERATORS

Python Membership Operators

Python's membership operators test for membership in a sequence, such as strings, lists, or tuples

. There are two membership operators as explained below –

[[Show Example](#)]

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not find a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Python program to illustrate

not 'in' operator

x = 24

y = 20

list = [10, 20, 30, 40, 50]

```
if (x not in list):
```

```
    print("x is NOT present in  
given list")
```

```
else:
```

```
    print("x is present in given  
list")
```

```
if (y in list):
```

```
    print("y is present in given  
list")
```

```
else:
```

```
    print("y is NOT present in  
given list")
```

IDENTITY OPERATORS

Python Identity Operators

Identity operators compare the memory locations of two objects. There are two Identity operators explained below –

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here is results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here is not results in 1 if id(x) is not equal to id(y).

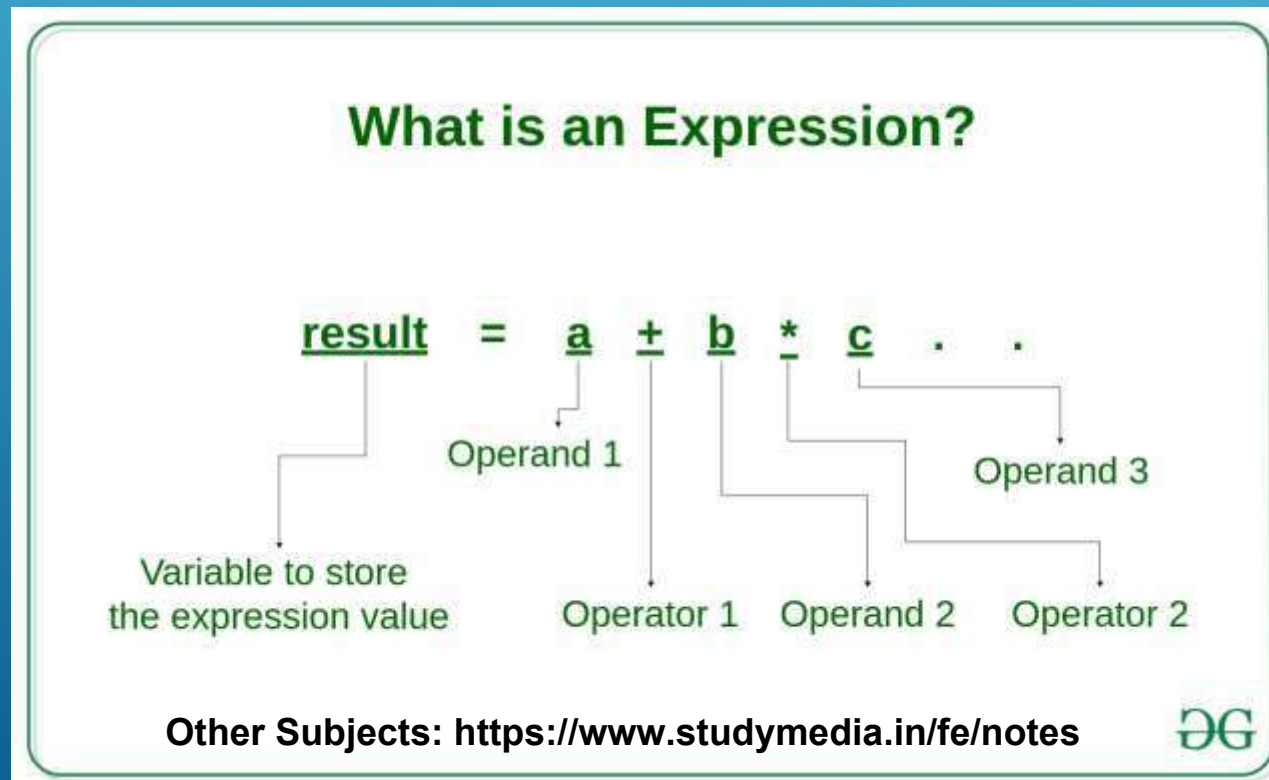
```
a = 10
b = 20
c = a
```

```
print(a is not b)
print(a is c)
```

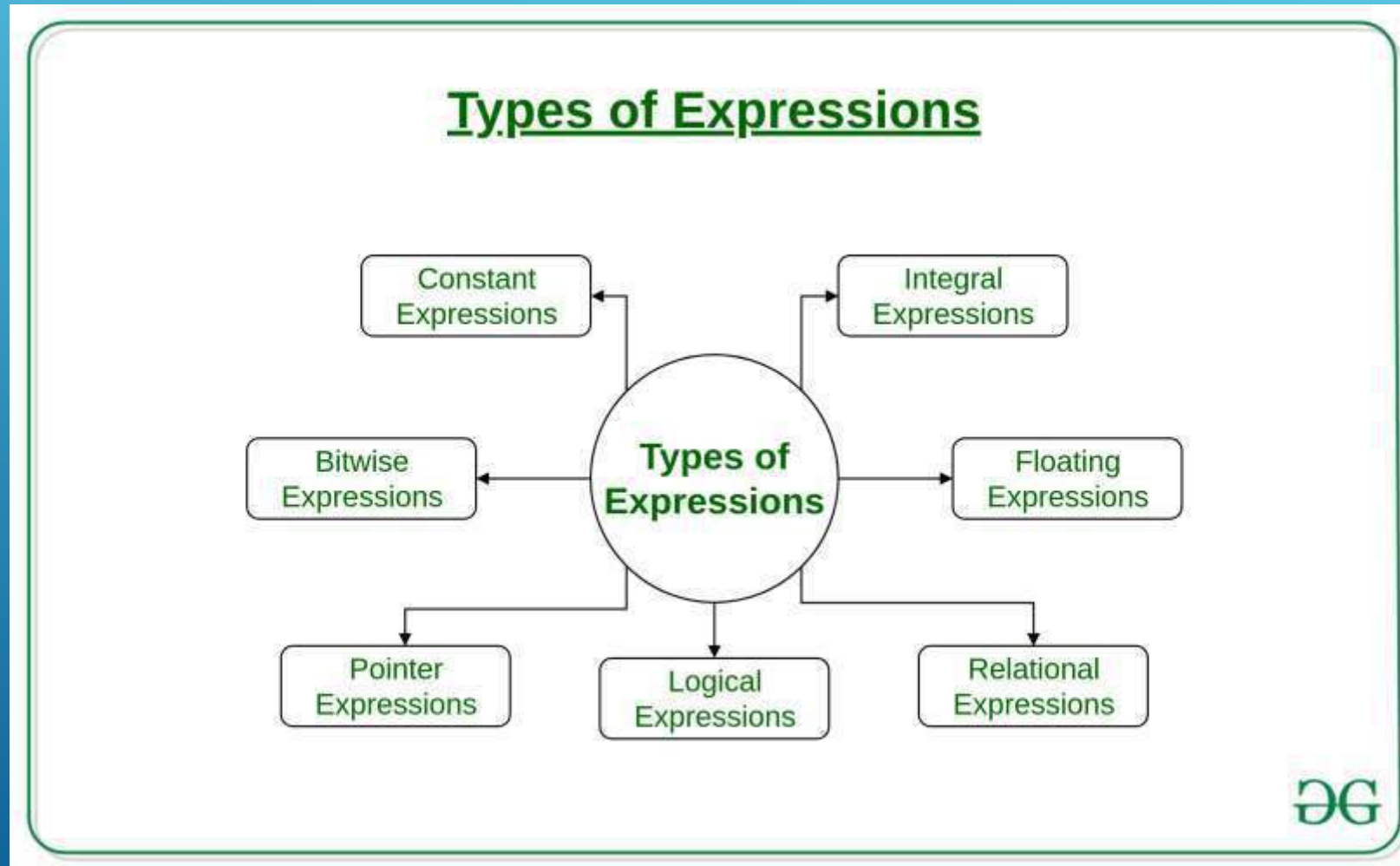
EXPRESSION IN PYTHON

An expression is a combination of operators and operands that is interpreted to produce some other value.

An expression is a combination of operators, constants and variables. An expression may consist of one or more operands, and zero or more operators to produce a value.



TYPES OF EXPRESSION



TYPES BASED ON OPERATOR POSITION

Infix

Ex- $x=y$, $x+y$

Order of execution of expression -
PEMDAS

Prefix

Ex.- $++Y$

P-parenthesis

E-exponents

Postfix

Ex- $B++$

MDAS-Multiplication, Division
Addition ,Subtraction

