# Unit-1:: Introduction to Python Programming

**Course Learning Objectives:**

- ❖ Students will come to know about What is python Programming Language and it's Classifications
- ❖ Students can understand about the Interpreter and Compiler
- ❖ Students will cover the history, Application and Features of Python Programming.
- ❖ Students will learn to Install and Run Python in Windows and Ubuntu
- ❖ Students will be able to know about Types of modes and IDE tools and sample program in Python.
- ❖ Students will know about basic concepts of Reserved key words, Identifiers, Variables and Constants,
- ❖ Students will learn about Statements & Comments, Operators and Operands in Python

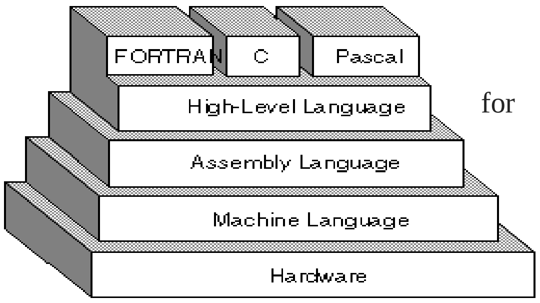
# Module 1 - What is Python Programming Language and its Classifications

## 1.1 What is a Language?

- Language is the method of human communication, either spoken or written, It consisting of the use of words in a structured and conventional way.
- The Language is nothing but set of instructions we are used to communicate.
- To communicate a particular person, we are passing instruction using a particular language like English, Telugu, and Hindi…. etc.
- But while using a language we need to fallow some of instructions, some rules are already they have given
- What are the rules? If I want to speak in English, to form a sentence, first we should be good at grammatically, are else we cannot form a sentence, to speak in English language.
- Similarly, computer language is also for communication sake only

## 1.2 What is Programming Language?

- Programming Language is also like an English, Telugu…etc.
- It should contain vocabulary and set of grammatical rules for instructing a computer to perform specific tasks.
- Each language has a unique set of keywords and a special syntax for organizing program instructions
- Programming languages are classified as:
- Machine language, Assembly language and High-level language
- The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, FORTRAN, Ada, and Pascal



## 1.3 Types of Programming Languages

**1.3.1 Machine Language:** The language of 0s and 1s is called as machine language. This is the only language which can be understood computers directly.

**Merits:**

- It is directly understood by the processor so has faster execution time since the programs written in this language need not to be translated.
- It doesn't need larger memory.

**Demerits:**

- It is very difficult to program using Machine Language since all the instructions are to be represented by 0s and 1s.
- Use of this language makes programming time consuming.
- It is difficult to find error and to debug.
- It can be used by experts only.

**1.3.2 Assembly Languages:** It is low level programming language in which the sequence of 0's and 1's are replaced by mnemonic (ni-monic) codes. Typical instruction for addition and subtraction.

**Example:** ADD for addition, SUB for subtraction etc.
Since our system only understand the language of 0s and 1s . therefore, a system program is known as assembler. Which is designed to translate an assembly language program into the machine language program.

**Merits:**

- It is makes programming easier than Machine Language since it uses mnemonics code for programming. Eg: ADD for addition, SUB for subtraction, DIV for division, etc.
- It makes programming process faster.
- Error can be identified much easily compared to Machine Language
- It is easier to debug than machine language.

**Demerits:**

- Programs written in this language is not directly understandable by computer so translators should be used.
- It is hardware dependent language so programmers are forced to think in terms of computer's architecture rather than to the problem being solved.
- Being machine dependent language, programs written in this language are very less or not portable.
- Programmers must know its mnemonics codes to perform any task.

**1.3.3 High Level Language:** High level languages are English like statements and programs Written in these languages are needed to be translated into machine language before execution using a system software such as compiler. Program written in high level languages are much easier to maintain and modified.

- High level language program is also called source code.
- Machine language program is also called object code

**Merits:**

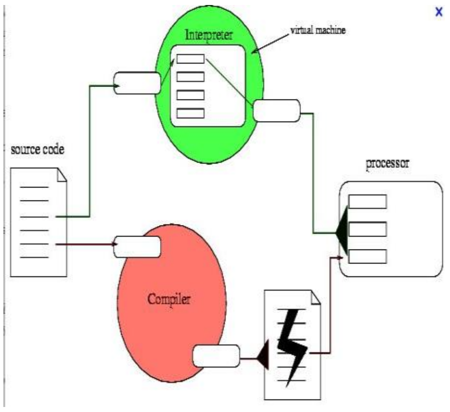
- Since it is most flexible, variety of problems could be solved easily
- Programmer does not need to think in term of computer architecture which makes them focused on the problem.
- Programs written in this language are portable.

**Demerits:**

- It is easier but needs higher processor and larger memory.
- It needs to be translated therefore its execution time is more.

## 1.4 Interpreter and Compiler

- We generally write a computer program using a high-level language. A high-level language is one which is understandable by humans.
- But a computer does not understand high-level language. It only understands program written in 0's and 1's in binary, called the machine code or object code.
- A program written in high-level language is called a source code. We need to convert the source code into machine code and this is accomplished(actioned) by compilers and interpreters.



| Interpreter | Compiler |
|---|---|
| **Translates program one statement at a time.** | Scans the entire program and translates it into machine code. |
| **It takes less amount of time to analyze the source code but the overall execution time is slower.** | It takes large amount of time to analyze the source code but the overall execution time is comparatively faster. |
| **No intermediate object code is generated, hence are memory efficient.** | Generates intermediate object code which further requires linking, hence requires more memory. |
| **Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.** | It generates the error message only after scanning the whole program. Hence debugging is comparatively |

|  |  |
|---|---|
|  | hard. |
| **Programming language like Python, Ruby use interpreters.** | Programming language like C, C++ use compilers. |

## Exercises

1. What is a programming language?
2. Explain different types of programming languages?
3. What is a compiler?
4. What is an interpreter?
5. How is compiled or interpreted code different from source code?
6. Difference between Interpreter and Compiler?

# Module 2 – Introduction to Python Programming

## 1.5 Introduction

- Python is an interpreter, object-oriented, high-level programming language with dynamic semantics (substance).
- Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.
- Python supports modules and packages, which encourages program modularity and code reuse
- Its consists of high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.
- Python is an open-source programming language made to both look good and be easy to read.

## 1.6 History of Python

- Python is an old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.
- Python is influenced by following programming languages:
    - ABC language.
    - Modula-3

## 1.7 Why Python was created?

- In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group.
- He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to a design of new language which was later named Python.
- Python drew inspiration from other programming languages like C, C++, Java, Perl, and Lisp.

## 1.8 Why the name was Python?

- No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series Monty Python's Flying Circus in late seventies.
- The name "Python" was adopted from the same series "Monty Python's Flying Circus".

### Release Dates of Different Versions

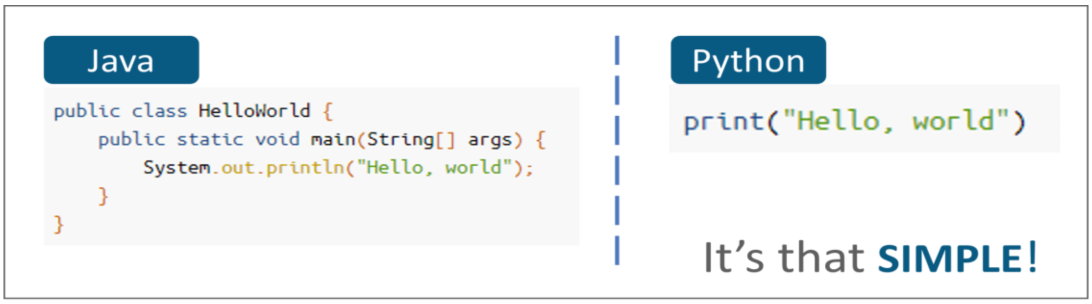
| Version | Release Data |
|---|---|
| **Python 1.0 (first standard release)** | January 1994 |
| **Python 1.6 (Last minor version)** | September 5, 2000 |
| **Python 2.0 (Introduced list comprehensions)** | October 16, 2000 |

|  |  |
|---|---|
| **Python 2.7 (Last minor version)** | July 3, 2010 |
| **Python 3.0 (Emphasis on removing duplicative constructs and module)**<br>**Python 3.5** | December 3, 2008<br>September 13, 2015 |
| **Python 3.7** | June 27, 2018 |
| **Python 3.8.0 (Last updated version)** | Oct. 14, 2019 |

## 1.9 Features of Python Programming

**A simple language which is easier to learn**

Python has a very simple and elegant (graceful) syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#.



```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world");
    }
}
```

```python
print("Hello, world")
```

**Free and open-source**

- You can even make changes to the Python's source code and update.

**Portability**

- You can move Python programs from one platform to another, and run it without any changes. It runs smoothly on almost all platforms including Windows, Mac OS X and Linux

**Extensible and Embeddable**

- Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code and other languages may not provide out of the box

**A high-level, interpreted language**

- Unlike C/C++, you don't have to worry about daunting (Cause to lose courage) tasks like memory management, garbage collection and so on, likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations

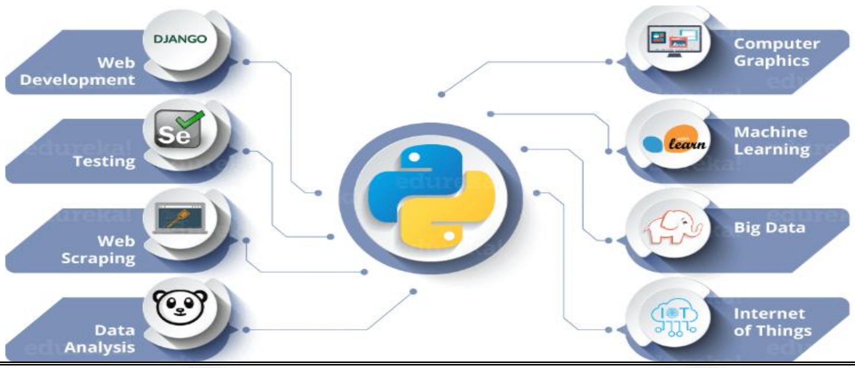
**Large standard libraries to solve common tasks**

- Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself
- For example: Need to connect **MySQL** database on a Web server? You can use **MySQLdb** library using **import MySQLdb**

**Object-oriented**

- Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively.
- With OOP, you are able to divide these complex problems into smaller sets by creating objects.

  **Images For Applications of Python**



**Web Applications**

You can create scalable Web Apps using frameworks and CMS (Content Management System) that are built on Python. Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS. Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

**Scientific and Numeric Computing**

There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: **SciPy** and **NumPy** that are used in general purpose computing. And, there are specific libraries like: **EarthPy** for earth science, **AstroPy** for Astronomy and so on. Also, the language is heavily used in machine learning, data mining and deep learning.

## 1.10 Why Python is very easy to learn?

One big change with Python is the use of whitespace to define code: spaces or tabs are used to organize code by the amount of spaces or tabs. This means at the end of each line; a **semicolon** is not needed and curly braces ({}) are not used to group the code. Which are both common in C. The combined effect makes Python a very easy to read language.

### Four Reason to Choose Python as First Language

### 1.10.1 Simple Elegant (Graceful) Syntax

- It's easier to understand and write Python code.
- Why? Syntax feels Naturals with Example Code

```
A=12
B=23
sum=A+B
print(sum)
```

- Even if you have never programmed before, you can easily guess that this program adds two numbers and prints it.

### 1.10.2 Not overly strict

- You don't need to define the type of a variable in Python. Also, it's not necessary to add semicolon at the end of the statement.
- Python enforces you to follow good practices (like proper indentation). These small things can make learning much easier for beginners.

### 1.10.3 Expressiveness of the language

- Python allows you to write programs having greater functionality with fewer(less) lines of code.
- We can build game **(Tic-tac-toe)** with Graphical interface in less than 500 lines of code
- This is just an example. You will be amazed how much you can do with Python once you learn the basics

### 1. 10. 4 Great Community and Support

- Python has a large supporting community.
- There are numerous active forums online which can be handy if you are stuck

## 1.11 Install and Run Python

### Ubuntu

1. Install the following dependencies:
   $ sudo apt-get install **build-essential** checkinstall
   $sudo apt-get install **libsqlite3-dev**
   $ sudo apt-get install **libbz2-dev**
   **(libreadline-gplv2-dev** libncursesw5-dev **libssl-dev** tk-dev **libgdbm-dev** libc6-dev)

   Video Link: - https://www.youtube.com/watch?v=sKiDjO_0dCQ
2. Go to Download Python page on the official site and click **Download Python 3.4.3**
3. In the terminal, go to the directory where the file is downloaded and run the command:

i. $ tar -xvf  Python-3.4.3.tgz
ii. This will extract your zipped file.

Note: The filename will be different if  you've downloaded a different version. Use the appropriate filename

4. Go to the extracted directory.
   $ cd Python-3.4.3
5. Issue the following commands to compile Python source code on your Operating system
   $./configure
   $ make
   $ make install
6. Go to Terminal and type   the following to run sample 'hello world ' Program

```
ubuntu@~$ python3.4.3
>>>print('Hellow World')
Hello World
>>>
```

**Windows**

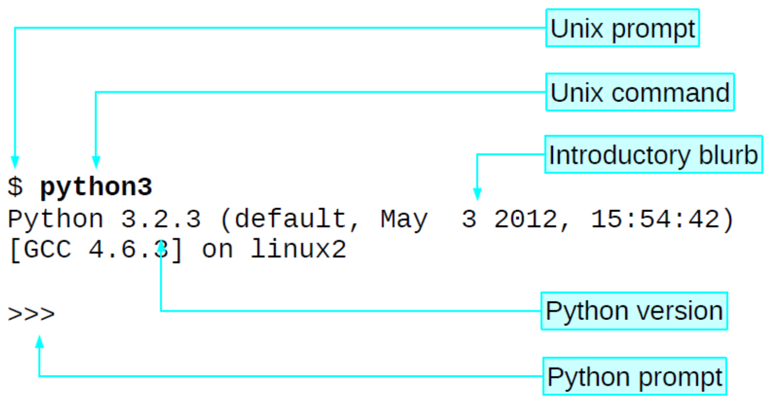
1. Go to Download Python page on the official site and click Download Python 3.4.3
2. When the download is completed, double-click the file and follow the instructions to install it
3. When Python is installed, a program called IDLE is also installed along with it. It provides graphical user interface to work with Python
4. Open IDLE, copy the following code below and press enter
   print ("Hello, World ")
5. To create a file in IDLE, go to File > New Window (Shortcut: Ctrl+N).
6. Write Python code (you can copy the code below for now) and save (Shortcut: Ctrl+S) with .py file extension like: hello.py or your-first-program.py
   print ("Hello, World ")
7. Go to Run > Run module (Shortcut: F5) and you can see the output. Congratulations, you've successfully run your first Python program

## 1.12 Modes of running

There are various ways to start Python

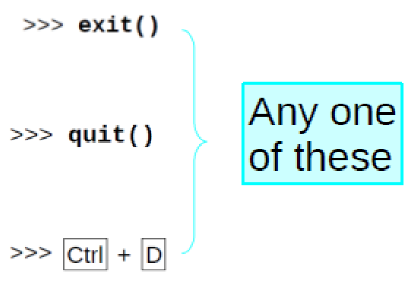
### Immediate Mode or Interactive Mode

- Typing python in the command line will invoke the interpreter in immediate mode. We can directly type in Python expressions and press enter to get the output **(>>>)**
- Is the python prompt and it tells us interpreter is ready for input



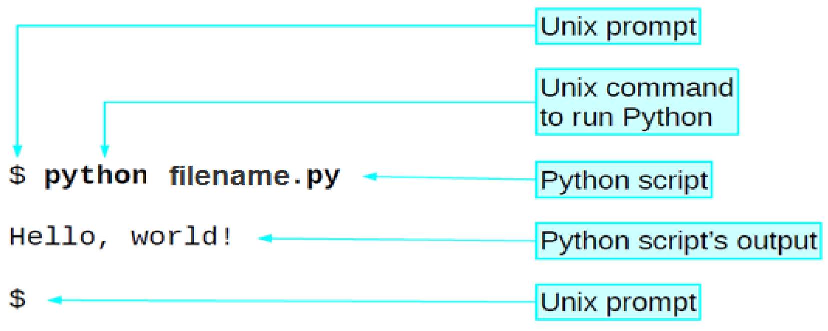
```
$ python3
Python 3.2.3 (default, May  3 2012, 15:54:42)
[GCC 4.6.3] on linux2

>>>
```

**Quitting Python**



### Script Mode

- This mode is used to execute Python program written in a file. Such a file is called a script. Scripts can be saved to disk for future use. Python scripts should have the extension .py, it means that the filename ends with **.py**.
- For example: **helloWorld.py**
- To execute this file in script mode we simply write **python3 helloWorld.py** at the command prompt.



```
$ python filename.py
Hello, world!
$
```

### Integrated Development Environment (IDE)

- We can use any text editing software to write a Python script file. Like Notepad, Editplus, sublime…etc

## 1.13 Python Program to Print Hello world!

- Type the following code in any text editor or an IDE and Save it as **helloworld.py**
- Now at the command window, go to the location of this file, use **cd** command to **change directory**
- To run the script, type **python3 helloworld.py** in the command window then we get output like this: **Hello World**
- In this program we have used the built-in function **print(),** to print out a string to the screen
- String is the value inside the quotation marks **i.e. Hello World**

Let us execute programs in different modes of programming.

**Interactive Mode Programming:**
Invoking the interpreter without passing a script file as a parameter brings up the following prompt:

```
$ python3

Python 3.7 (r27:82525, Jul  4 2010, 09:01:59) [MSC v.1500 32 bit (Intel)] on win32

Type "copyright", "credits" or "license()" for more information.

>>>
```

Type the following text at the Python prompt and press the Enter:

```
>>> print "Hello, IIIT RK Valley, RGUKT-AP!";
```

If you are running new version of Python, then you need to use print statement with parenthesis as in print ("Hello, IIIT RK Valley");. However in Python version 2.7, this produces the following result:

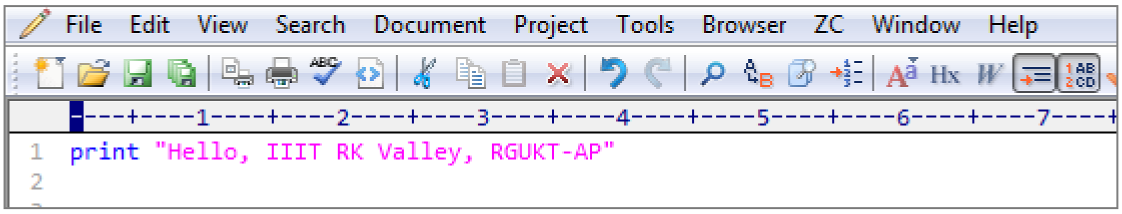
```
Hello, IIIT RK Valley, RGUKT-AP!
```

**Script Mode Programming:**

Python programs must be written with a structure. The syntax must be correct, or the interpreter will generate error messages and not execute the program. This section introduces Python by providing a simple example program.

To write Python programming in script mode we have to use editors. Let us write a simple Python program in a script mode using editors. Python files have extension .py. Type the following source code in a simple.py file.

Program 2.1 (simple.py) is one of the simplest Python programs that does something



```
print "Hello, IIIT RK Valley, RGUKT-AP"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows:

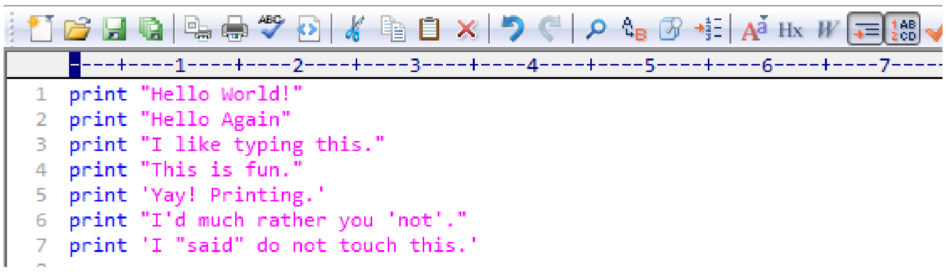
```
$ python3 simple.py
```

This produces the following result:

```
Hello, IIIT RK Valley, RGUKT-AP!
```

## Exercise

1. What is Python Programming Language?
2. How many ways can run the Python Programming?
3. What is print statement
4. What is prompt
5. Print "Welcome to Python Programming Language" from interactive Python.
6. Write above statement in exercise1.py to print the same text.
7. Run the modified exercise1.py script.
8. Write the below statements in exercise2.py to print the same below



```
print "Hello World!"
print "Hello Again"
print "I like typing this."
print "This is fun."
print 'Yay! Printing.'
print "I'd much rather you 'not'."
print 'I "said" do not touch this.'
```

# Module-3 Reserved key words, Identifiers, Variables and Constant

## 1.14 Keywords

- Keywords are the reserved words in Python and we cannot use a keyword as variable name, function name or any other identifier.
- They are used to define the syntax and structure of the Python language.
- In Python, keywords are case sensitive.
- There are 35 keywords in Python 3.7.3 This number keep on growing with the new features coming in python
- All the keywords except True, False and None are in lowercase and they must be written as it is.
- The list of all the keywords is given below

We can get the complete list of keywords using python interpreter help utility.

```
$ python3
>>> help()
help> keywords
```

| **False** | **class** | **from** | **or** |
|---|---|---|---|
| **None** | continue | global | pass |
| **True** | def | if | raise |
| **and** | del | import | return |
| **as** | elif | in | try |
| **assert** | else | is | while |
| **async** | except | lambda | with |
| **await** | finally | nonlocal | yield |
| **break** | for | not | |

## 1.15 Identifiers

Identifier is the name given to entities like class, functions, variables etc. in Python. It helps differentiating 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 perfectly fine.
- Keywords cannot be used as identifiers.

```
>>> global = 1  File "<interactive input>", line 1
global = 1
^SyntaxError: invalid syntax
```

- We cannot use special symbols like !, @, #, $, % etc. in our identifier. >>> a@ = 0

```
File "<interactive input>", line 1a@ = 0^
SyntaxError: invalid syntax
```

- Identifier can be of any length.

## 1.16 Things to care about

- Python is a case-sensitive language. This means, **Variable** and **variable** are not the same. Always give a valid name to identifiers so that it makes sense.
- While, c = 10 is valid. Writing count = 10 would make more sense and it would be easier to figure out what it does even when you look at your code after a long gap.
- Multiple words can be separated using an underscore, this_is_a_long_variable.
- We can also use camel-case style of writing,
- i.e., capitalize every first letter of the word except the initial word without any spaces.

- For example: camelCase Example.

## 1.17 Variable

- A variable is a location in memory used to store some data.
- Variables are nothing but reserved memory locations to store values, this means that when we create a variable, we reserved some space in memory.
- They are given unique names to differentiate between different memory locations.
- The rules for writing a variable name are same as the rules for writing identifiers in Python.
- We don't need to declare a variable before using it.
- In Python, we simply assign a value to a variable and it will exist.
- We don't even have to declare the type of the variable. This is handled internally according to the type of value we assign to the variable.

**Variable assignment:** We use the assignment operator (=) to assign values to a variable. Any type of value can be assigned to any valid variable.

```
a = 5
b = 3.2
c = "Hello"
```

Here, we have three assignment statements. 5 is an integer assigned to the variable a.

Similarly, 3.2 is a floating-point number and "Hello" is a string (sequence of characters) assigned to the variables b and c respectively.

**Multiple assignments:**

- In Python, multiple assignments can be made in a single statement as follows: a, b, c = 5, 3.2, "Hello"
- If we want to assign the same value to multiple variables at once, we can do this as x = y = z = "same"
- This assigns the "same" string to all the three variables.

**Constants:** A constant is a type of variable whose value cannot be changed. It is helpful to think of constants as containers that hold information which cannot be changed later. You can think of constants as a bag to store some books which cannot be replaced once placed inside the bag.

**Assigning value to constant in Python** In Python, constants are usually declared and assigned in a module. Here, the module is a new file containing variables, functions, etc which is imported to the main file. Inside the module, constants are written in all capital letters and underscores separating the words.

**Example 1: Declaring and assigning value to a constant**

Create a **constant.py**:

```
PI = 3.14
GRAVITY = 9.8
```

Create a **main.py**:

```
import constant
print(constant.PI)
print(constant.GRAVITY)
```

**Output**

```
3.14
9.8
```

In the above program, we create a **constant.py** module file. Then, we assign the constant value to *PI* and *GRAVITY*. After that, we create a **main.py** file and import the constant module. Finally, we print the constant value.

**Note**: In reality, we don't use constants in Python. Naming them in all capital letters is a convention to separate them from variables, however, it does not actually prevent reassignment.

## 1.18 Statements & Comments

Instructions that a Python interpreter can execute are called statements.
For example,
a = 1 is an assignment statement. if statement, for statement, while statement etc.

### 2.18.1 Multi-line statement

- ❖ In Python, end of a statement is marked by a newline character. But we can make a statement extend over multiple lines with the line continuation character (\).
  For example:

  ```
  a = 1 + 2 + 3 + \
  4 + 5 + 6 + \
  7 + 8 + 9
  ```

  This is explicit line continuation.
- ❖ This is explicit line continuation. In Python, line continuation is implied inside parentheses ( ), brackets [ ] and braces { }.
- ❖ For instance, we can implement the above multi-line statement as

  ```
  a = (1 + 2 + 3 +
  4 + 5 + 6 +
  7 + 8 + 9)
  ```

- ❖ Here, the surrounding parentheses ( ) do the line continuation implicitly. Same is the case with [ ] and { }.

  ```
  For example: colors = ['red',
                         'blue',
                         'green']
  ```

- ❖ We could also put multiple statements in a single line using semicolons, as follows a = 1; b = 2; c = 3

### 1.18.2 Comments

- Comments are very important while writing a program. It describes what's going on inside a program so that a person looking at the source code does not have a hard time figuring it out.
- You might forget the key details of the program you just wrote in a month's time. So taking time to explain these concepts in form of comments is always fruitful.
- In Python, we use the hash (#) symbol to start writing a comment. It extends up to the newline character.
- Comments are for programmers for better understanding of a program. Python Interpreter ignores comment.

  ```
  #This is a comment
  #print out Hello
  print('Hello')
  ```

#### Multi-line comments

- If we have comments that extend multiple lines, one way of doing it is to use hash (#) in the beginning of each line. For example:

  ```
  #This is a long comment
  #and it extends
  #to multiple lines
  ```

- Another way of doing this is to use triple quotes, either ''' or """.
- These triple quotes are generally used for multi-line strings. But they can be used as multi-line comment as well. Unless they are not docstrings, they do not generate any extra code.

  ```
  """This is also a
  perfect example of multi-line comments"""
  ```

## Exercise

1. What is a variable?
2. What is value? How can you define the type of value?
3. What is assignment statement?
4. What is comment?
5. What is expression?

6. Write a program that uses raw_input to prompt a user for their Name and then welcomes them.

```
Enter your name: Chuck

Hello Chuck
```

7. Write a program to prompt the user for hours and rate per hour to compute gross pay.

```
Enter Hours: 35

Enter Rate: 2.75

Pay: 96.25
```

8. Assume that we execute the following assignment statements: width = 17, height = 12.0
   For each of the following expressions, write the value of the expression and the type (of the value of the expression).

a. width/2
b. width/2.0
c. height/3
d. 4. 1 + 2 * 5

   Write program and also Use the Python interpreter to check your answers.

9. Write a program which prompts the user for a Celsius temperature, convert the temperature to Fahrenheit, and print out the converted temperature.
10. Will the following lines of code print the same thing? Explain why or why not.

```
x = 6

print(6)

print("6")
```

11. Will the following lines of code print the same thing? Explain why or why not.

```
x = 7

print(x)

print("x")
```

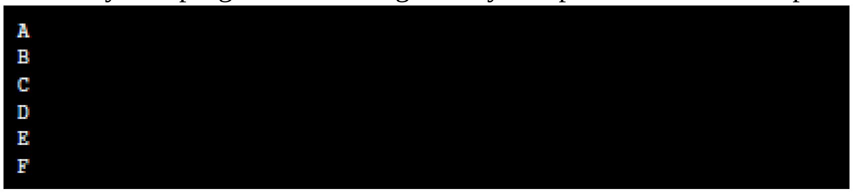
12. What happens if you attempt to use a variable within a program, and that variable has not been assigned a value?
13. What is wrong with the following statement that attempts to assign the value ten to variable x?

```
10 = x
```

14. In Python can you assign more than one variable in a single statement?
15. Classify each of the following as either a legal or illegal Python identifier:

| | | | |
|---|---|---|---|
| a. | Flag | h. | sumtotal |
| b. | if | i. | While |
| c. | 2x | j. | x2 |
| d. | -4 | k. | $16 |
| e. | sum_total | l. | _static |
| f. | sum-total | m. | wilma’s |
| g. | sum total | | |

16. What can you do if a variable name you would like to use is the same as a reserved word?
17. What is the difference between the following two strings? ’n’ and ’\n’?
18. Write a Python program containing exactly one print statement that produces the following output



```
A
B
C
D
E
F
```

# Module-4 Python Operators

## 1.19 Get Started with Python Operators

- Operators are special symbols in Python that carry out arithmetic or logical computation. The value that the operator operates on is called the operand.
  For example: >>> 2+3 = 5
- Here, + is the operator that performs addition. 2 and 3 are the operands and 5 is the output of the operation.
- Python has a number of operators which are classified below.
  - Arithmetic operators
  - Assignment operators
  - Comparison (Relational) operators
  - Logical (Boolean) operators
  - Bitwise operators
  - Assignment operators
  - Special operators

### 1.19.1 Arithmetic Operators

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

| Arithmetic operators in Python | | |
|---|---|---|
| **Operator** | **Meaning** | **Example** |
| **+** | Add two operands or unary plus | x + y +2 |
| **-** | Subtract right operand from the left or unary minus | x – y -2 |
| ***** | Multiply two operands | x * y |
| **/** | Divide left operand by the right one (always results into float) | x / y |
| **%** | Modulus - remainder of the division of left operand by the right | x % y (remainder of 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 (x to the power y) |

Example Program for Arithmetic operators

```
x = 25
y = 15
print('x + y = ',x+y)
print('x - y = ',x-y)
print('x * y = ',x*y)
print('x / y = ',x/y)
print('x // y = ',x//y)
print('x ** y = ',x**y)
```

### 1.19.2 Comparison (Relational) Operators

Comparison operators are used to compare values. It either returns True or False according to the condition

| Operator | Meaning | Example |
|---|---|---|
| > | Greater than - True if left operand is greater than the right | x > y |
| < | Less than - True if left operand is less than the right | x < y |
| == | Equal to - True if both operands are equal | x == y |
| != | Not equal to - True if operands are not equal | x != y |
| >= | Greater than or equal to - True if left operand is greater than or equal to the right | x >= y |
| <= | Less than or equal to - True if left operand is less than or equal to the right | x <= y |

Example: x = 10; y = 12

```
print('x > y  is', x>y)
```

### 1.19.3 Logical (Boolean) Operators

Logical operators are the and, or, not operators

| 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 |

Here is an example.

```
x = True  y = False

print('x and y is',x and y)

print('x or y is',x or y)

print('not x is',not x)
```

### 1.19.4 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) |

### 1.19.5 Assignment Operators

Assignment operators are used in Python to assign values to variables.
a = 5 is a simple assignment operator that assigns the value 5 on the right to the variable *a* on the left.

There are various compound operators in Python
Example:

a += 5 that adds to the variable and later assigns the same
It is equivalent to a = a + 5.

| Operator | Example | Equivatent to |
|---|---|---|
| = | x = 5 | x = 5 |
| += | x += 5 | x = x + 5 |
| -= | x -= 5 | x = x - 5 |
| *= | x *= 5 | x = x * 5 |
| /= | x /= 5 | x = x / 5 |
| %= | x %= 5 | x = x % 5 |
| //= | x //= 5 | x = x // 5 |
| **= | x **= 5 | x = x ** 5 |
| &= | x &= 5 | x = x & 5 |
| \|= | x \|= 5 | x = x \| 5 |
| ^= | x ^= 5 | x = x ^ 5 |
| >>= | x >>= 5 | x = x >> 5 |
| <<= | x <<= 5 | x = x << 5 |

### 1.19.6 Membership Operators

**in** and **not in** are the membership operators in Python. They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

In a dictionary we can only test for presence of key, not the value.

| Operator | Meaning | Example |
|---|---|---|
| **in** | True if value/variable is found in the sequence | 5 in x |
| **not in** | True if value/variable is not found in the sequence | 5 not in x |

Here is an example.

```
x = 'Hello world'

y = {1:'a',2:'b'}

print('H' in x)

print('hello' not in x)

print(1 in y)

print('a' in y)
```

Here, 'H' is in x but 'hello' is not present in x (remember, Python is case sensitive). Similarly, 1 is key and 'a' is the value in dictionary y. Hence, 'a' in y returns False.

## 1.20 Rules for solving equations in Python

Order of Operations Worksheets - BEDMAS or PEMDAS

Step 1: First, **perform** the operations within the brackets or parenthesis.
Step 2: Second, evaluate the exponents.
Step 3: Third, **perform** multiplication and division from left to right.
Step 4: Fourth, **perform** addition and subtraction from left to right.

**Example**: >>>(40+20)*30/10

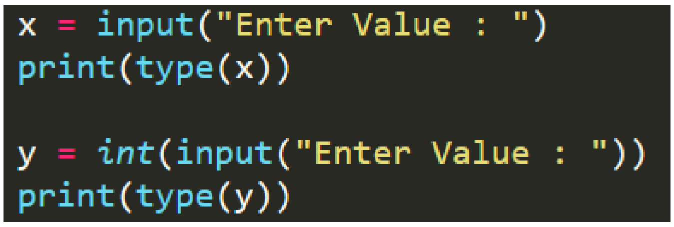
**Output:** 180

## 1.21 Reading Input value from the User

The print() function enables a Python program to display textual information to the user. Python provides built-in functions to get input from the user. The function is input().

Generally input() function is used to retrieve string values from the user.

**Program 3.6 (sampleinput.py) shows the type of user enter values**



```
x = input("Enter Value : ")
print(type(x))

y = int(input("Enter Value : "))
print(type(y))
```

**Program 3.6 (sampleinput.py) produce result as**

```
Enter Value : 4
<class 'str'>
Enter Value : 4
<class 'int'>
```

We can use the above mentioned functions in *python 2.x*, but not on *python 3.x*. input() in python 3.x always return a string. Moreover, raw_input() has been deleted from python 3

python2.x python3.x

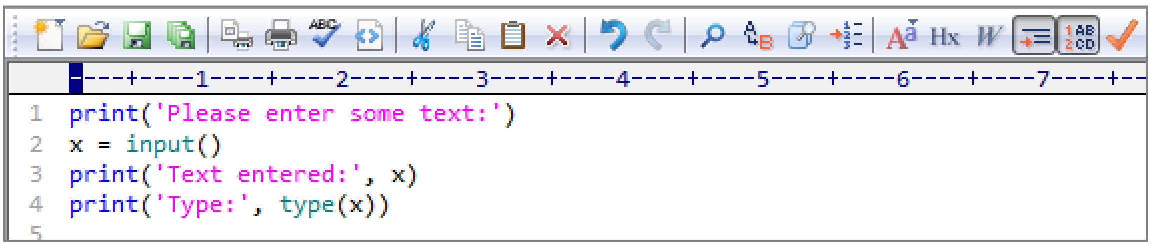
raw_input() -------------- input()
input() ------------------- eval(input())

We can simply say in *python 3.x* only use of the input () function to read value from the user and it assigns a string to a variable.

x = input()

The parentheses are empty because, the input function does not require any information to do its job.

**Program 3.7 (usinginput.py) demonstrates that the input function produces a string value.**



```
print('Please enter some text:')
x = input()
print('Text entered:', x)
print('Type:', type(x))
```

The following shows a sample run of Listing 2.11 (usinginput.py):

```
>>>
Please enter some text:
Hi RGUKT
('Text entered:', 'Hi RGUKT')
('Type:', <type 'str'>)
```

The second line shown in the output is entered by the user, and the program prints the first, third, and fourth lines. After the program prints the message *Please enter some text:*, the program's execution stops and waits for the user to type some text using the keyboard. The user can type, backspace to make changes, and type some more. The text the user types is not committed until the user presses the Enter (or return) key. In Python 3.X, input() function produces only strings, by using conversion functions we can change the *type* of the input value. Example as int(), str() and float().

## Exercise

1. Explain type of operators
2. Find whether these expressions will evaluate to **True** or **False**. Then try them in interactive mode and script mode
   a. 4 > 5
   b. 12 != 20
   c. 4 <= 6
   d. 4 => 1
   e. 'sparrow' > 'eagle'
   f. 'dog' > 'Cat' or 45 % 3 == 0
   g. 42+12 < 34//2
   h. 60 - 45 / 5 + 10 == 1 and 32*2 < 12
3. Write a python program to find type of the value of below expressions
   a. 4 > 23
   b. 5+=21
   c. 2**=3
4. What is compound assignment operator? Explain with examples?
5. Difference between '=' and '=='
6. Difference between '/' and '//'
7. Write a python program to prompts the user for x and y operands and find result for all arithmetic operators.

## Multiple Choice Questions

1. Python is -------
   a) Objected Oriented Programming Language
   b) Powerful Programming Language
   c) Powerful scripting Language
   d) All the above
2. Python is developed by ------------
   a) Guido van Rossum.
   b) Balaguruswami
   c) James Gosling
   d) None of the above
3. Which of the following is not a keyword?
   a) eval
   b) assert
   c) nonlocal
   d) pass
4. All keywords in Python are in _________
   a) lower case
   b) UPPER CASE
   c) Capitalized
   d) None of the mentioned
5. Which of the following is true for variable names in Python?
   a) unlimited length
   b) all private members must have leading and trailing underscores
   c) underscore and ampersand are the only two special characters allowed
   d) none of the mentioned
6. Which of the following translates and executes program code line by line rather than the whole program in one step?
   a) Interpreter
   b) Translator
   c) Assembler
   d) Compiler
7. Which of the following languages uses codes such as MOVE, ADD and SUB?
   a) assembly language
   b) Java
   c) Fortarn
   d) Machine language
8. What is the output of the following assignment operator?

   ```
   y = 10
   x = y += 2
   print(x)
   ```

   a) 12
   b) 10
   c) Syntax error
9. What is assignment operator?
   a) = =
   b) =
   c) +=
   d) - =
10. What is a correct Python expression for checking to see if a number stored in a variable x is between 0 and 5?
    a) x>0 and <5
    b) x>0 or x<5
    c) x>0 and x<5

## Descriptive Questions:

1. Explain about Applications and Features of Python Programming Language?
2. Write short notes on types of operators in python with appropriate examples?
3. Explain about interpreter and compiler?
4. Explain about identifiers
5. Explain about types of modes
6. Explain briefly about:
    - Constant and variables
    - keywords and statement

## Solved Problems:

1. Evaluate the value of z=x+y*z/4%2-1

```
x=input(“Enter the value of x= ”)
y=input(“Enter the value of y=  “)
z=input(“Enter the value of z=   “)
a=y*z
b=a/4
c=b%2
t=x+c-1
print(“the value of z=x+y*z/4%2-1 is”,t)
```

**input**: Enter the value of x=2

```
Enter the value of y=3
Enter the value of z=4
```

**Output**: the value of z=x+y*z/4%2-1 is 2

Note: while solving equations follow the BEDMAS or PADMAS Rule

2. Python program to show bitwise operators

```
a = 10
b = 4
# Print bitwise AND operation
print("a & b =", a & b)
# Print bitwise OR operation
print("a | b =", a | b)
# Print bitwise NOT operation
print("~a =", ~a)
# print bitwise XOR operation
print("a ^ b =", a ^ b)
```

**Output:**

```
a & b = 0
a | b = 14
~a = -11
a ^ b = 14
```

## Unsolved Problems:

1. Write a program to find out the value of 9+3*9/3?
2. Write a program to find out the value of (50-5*6)/4?
3. Write a program to find out the value of 3*3.75/1.5?
4. 45-20+(10?5)%5*3=25 what is the operator in the place of “?” mark?
5. 3*6/22%2+7/2*(3.2-0.7)*2 in this problem which one is evaluated first? finally what is the answer?
6. Write a program to display **Arithmetic** operations for any two numbers?
7. Python Program to find the average of 3 numbers?
8. Write a python program to find simple and compound interest?

**References:**

**Book: Python programming (Using Problem Solving Approach) – REEMA THARAJA**