Python

Python is a high-level programming language. It is a universally interpreted interactive object-oriented programming language. Python was developed by Guido van Rossum from 1985 to 1990. Python is placed under the top 10 best programming languages.

Why Python?

As python is a high-level programming language that’s why it uses English keywords and very easy to understand due to its fewer syntactical structure then other languages. Python is a must for students and professionals if they want to become a great Software Developers. Python provides a solution to every possible problem related to programming.

Python is used for:

- Web Application development
- Software Development
- Mathematics
- System Application Development
- Artificial Intelligence
- Game Development

Propertise of Python

The properties of python a programming language is as follows.

- It supports an object-oriented, functional and structured way of coding.
- It can be used as a scripting language or a structured bytecode to create great applications.
- It offers dynamic data types at very high levels and supports dynamic type checking.
- It supports automatic waste collection.
- It can be easily integrated into C, C ++, COM, ActiveX, CORBA, and Java.
- It is processed by an interpreter at runtime. You don't need to compile your program before you can run it.
- It provides a direct way to interact and prompt with interpreter to write a program.

Advantages of Python

The advantages of python programming language are as follows.

- Python works on different operating systems like Windows, Mac, Linux, Raspberry Pi, etc.
- Python uses the English language for its coding and syntax.
- Python provides a facility to programmer to write fewer lines of code than other programming languages.
- Python is an excellent programming language for beginners as well as professionals and supports the development of several applications.

Applications of Python

The applications of python programming language is as follows.

- It is easy to learn the syntax as it uses English
- It is easy to read because of its aligned structure of code.
- It is easy to maintain.
- It provides broad standard libraries to every operating system e.g. Windows, Linux, and Macintosh.
- It allows interactive testing and debugging of snippets of code of a program.
- It can be run easily on a wide variety of hardware platforms with the same interface.
- It provides an interface to all major databases.
- It supports GUI applications that can be created by using different resources of different operating systems.
- It provides support and a better structure for complex and large codes.

Python's Hello World

To excite you about the python programming language. Below is a Hello World program.

In [3]:
```
print ("Hello World");
```

```
Hello World
```

Indentation in Python

Indentation refers to the spaces at the start of a every code line.

Python does not use braces for block indentation of class, function, and control structure.
Python use line indentation to manage the block of code.
In python, indentation is very necessary otherwise python throws an error if your indentation is not the same for every line in a particular block.
The following are examples of indentations.

In [6]:
```
if True:
    print("Welcome")
else:
    print("Not Welcome")
```

```
Welcome
```

As the above code is perfectly indented then it is easily executed.

In [ ]:
```
if True:
print("Welcome")
else
print("Not Welcome")
```

python throws error message while execution because the above code is not indented.

Flow of Control

In Python, statements are written as a list, in the way that a person would write a list of things to do. The computer starts off by following the first instruction, then the next, in the order that they appear in the program. It only stops executing the program after the last instruction is completed. We refer to the order in which the computer executes instructions as the flow of control. When the computer is executing a particular instruction, we can say that control is at that instruction.

Data Types

Data Type is the representation of data stored in memory or assigned to a variable.
Python has different standard data types that are used to represent different types of data and on which we perform a different operation.
Python has five standard data types which are as follows.

- Numeric Type
- Text Type
- Sequence Type
- Mapping Type
- Boolean Type

Numeric Type

Numeric type stores numeric values. As in Python, there is no way to describe the type of a variable. It is the value that defines the data type of a variable.

Python support four types of numeric values which are as follows.

- Integer
- Long
- Float
- Complex

The following are examples of these numeric values assigned to the variables.

In [12]:
```
a = 1 # integer it can be signed or unsigned.
b = 2112152694 # long integer it can also be signed and unsigned.
c = 5.11 # float it can also be signed and unsigned.
d = 3.14j # complex it can also be signed and unsigned.</code></pre>
```

Text Type

In python, the text is represented by the string which includes alpha or alphanumeric continuous characters enclosed in single, double, or triple quotes. This type of text is called a string data type in python.

The example of a string data type is as follows.

In [16]:
```
mystring = 'hello'
my_string = "world"
```

Sequence Type

In python, the sequence type is used to store the different or the same type of data in order. The two sequence types are as follows.

- Tuple
- List

Tuple and List are two sequence data types which store data of the same and different types in an ordered form.

Data stored in tuple cant be changed while in the list we can change the data.

The examples of sequence types are as follows.

```
In [18]: mystring = 'hello'
         mytuple = (1,2,3,4,5) # tuple with same data type.
         my_tuple = ("Hello",71,'a',3.14) # tuple with different data types.
         mylist = [1,2,3,4,5] # list with same data type.
         my_list = ["Hello",71,'a',3.14] # list with different data types.
```

Mapping Type

In python mapping type is represented by Dictionaries it is like the hash table. Dictionaries can store any type of data in it.

Dictionaries are enclosed in ({ }) curly brackets. Dictionaries have the key against every value stored in it.

The following is an example of a dictionary.

```
In [19]: mydict = {'name': 'John', 'Address': '48 Avenue', 'Age': 35}
         # In this name, Address, and Age are the keys
         # And the John, 48 Avenue, and 35 are the values.</code></pre>
```

Boolean Type

In python, a boolean type is represented by the 'bool' keyword. The bool can either be true or false.

It has only 2 possible outcomes. The following is an example of the bool type.

```
In [21]: bool(10>7)# true
         bool(5<2)# false

Out[21]: False
```

Variable

Variables are the simplest data type that stores data values. It is actually a reserved memory at the backend to store a value.

Every time a variable is created it reserves some memory to store the data values at the backend.

Python has no special command to create a variable. It can be automatically created when you assign any value to it.

In python, there is no need to declare the data type of a variable before assigning a value to it. The interpreter itself figure out the data type of a variable.

Python even facilitates you that you can change the data type after the values have been set.

The following are examples of variables.

In [24]:
```
a = 10
b = 10.5
c = 'a'
d = "Hello"
```

All the above variables are assigned with different values and legally correct.

Variable Names

A variable name must be a meaningful name that provides you information through its name. The name can be short as well if it is not meant to be meaningful. There are rules for naming the python variable which is as follows.

- A variable name in python can be started with an alphabet letter or an underscore character.
- A variable name in python cannot start with a number.
- A variable name in python can only have alphabets, number and underscore in its name.
- In python, variables are case sensitive.

The following are some examples of variable names.

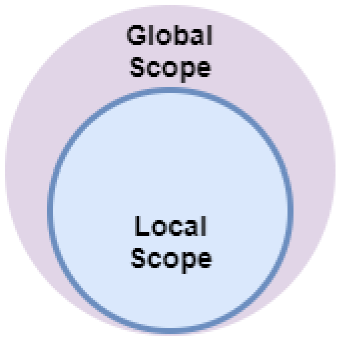
In [25]:
```
# Some legal variable name conventions in python.
thisvar = 1
this_var = 2
_thisvar = 3
Thisvar = 4
THISVAR = 5
thisvar1 = 6
this2var = 7
```

Variable Scope

The region where we can define and initialize the variable is called Scope.

In python, there are two types of scopes which are as follows.

- Local Scope
- Global Scope



Local Scope

The variable that is created inside the region of the function is called Local Scope.

The variable created in the function can be accessed in the scope of the function by any other function which is also defined in the scope of that function.

Global Scope

The variable that is created inside the region of the main body of the python code is called Global Scope.

The variable also created outside the function is global and can be used anywhere in the python code.

The variable created by using the **global** keyword inside the function is also a global variable that obeys the global scope and can be used anywhere in the python code.

User Input

Python allows us to prompt the user for input. We can take user input in our python program where it is necessary.

Python provides the 2 built-in methods for its two versions for taking inputs from the users which are as follows.

- input()
- raw_input()

Input()

This function is used by the Python 3.6 version.

The function takes input from the user and then evaluate automatically which type of data is entered by the user. Whether it is a number or an alphabetic.

If the input is not correctly provided by the user then python throws the syntax error or throws an exception.

The example of the input() function is as follows.

```
In [22]: Name = input("Please Enter Your Name: ") # taking input from user.
         print(Name) # printing the given input.
```

```
Please Enter Your Name: Abdul Mateen
Abdul Mateen
```

In Python 3.6 version when input() function is excuted. The flow of the execution is stopped until the user has given an input.

Selection

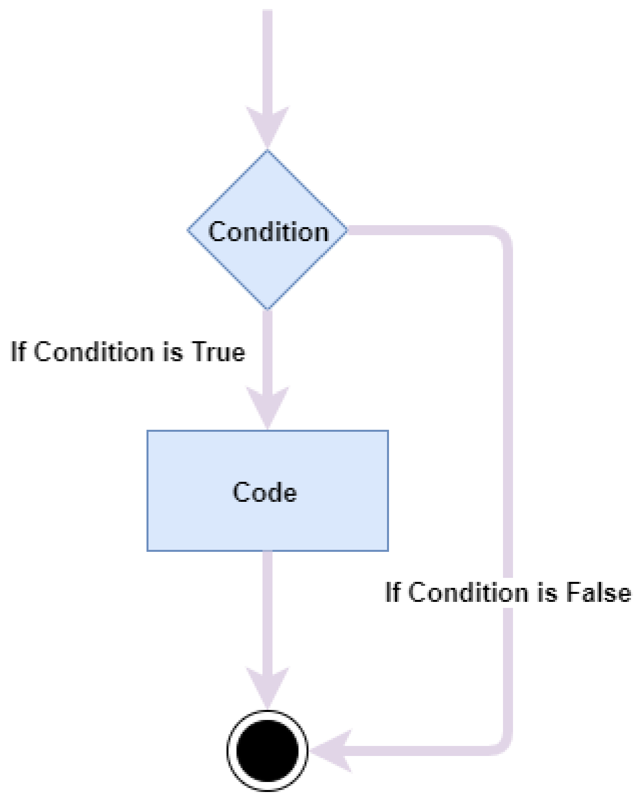
In python, selection statements are used for decision-making purposes.

These statements check the condition while execution and specify the action based on the condition.

These statements evaluate multiple expressions that produce the boolean type result which can either be true or false.

These statements are of three types which are as follows.

- If Statement
- If-Else Statement
- If-Elif-Else Statement
- Nested If Statement



If Statement

If statement consists of an expression that produces the boolean type result which is followed by the instructions which will execute when the outcome of the expression is true.

The example of the If statement is as follows.

In [26]:
```
a = 10
b = 5
if a>b:
   print("a is greater than b")
```

```
a is greater than b
```

If-Else Statement

If-Else statement consists of two clauses one is **if** clause and the second one is **else** clause.

Usually in the if-else statement, if statement is followed by the else statement which is executed when the expression is false.

The example of the If-Else statement is as follows.

In [27]:
```
a = 10
b = 5
if a>b:
   print("a is greater than b")
else:
   print("a is less than b")
```

```
a is greater than b
```

If-Elif-Else Statement

If-Elif-Else statement consists of three clauses and has two expressions.

The first clause is the **If** clause which has the first condition, the second clause is the **Elif** clause which also has one condition, and the third clause is the **Else** clause.

In the If-Elif-Else, if statement is followed by elif statement which is executed when the condition of if statement is false and elif statement is followed by the else statement which is executed when the condition of elif statement is false.

The example of the If-Elif-Else statement is as follows.

In [28]:
```
a = 10
b = 5
if a == b:
   print("a is equal to b")
elif a>b:
   print("a is greater than b")
else:
   print("a is less than b")
```

```
a is greater than b
```

Nested If Statement

Nested If statement is a statement in which there is a statement in another statement.

Python allows us that we can use If-Elif-Else , If Else, and If statement in another If statement.

The example of the Nested If statement is as follows.

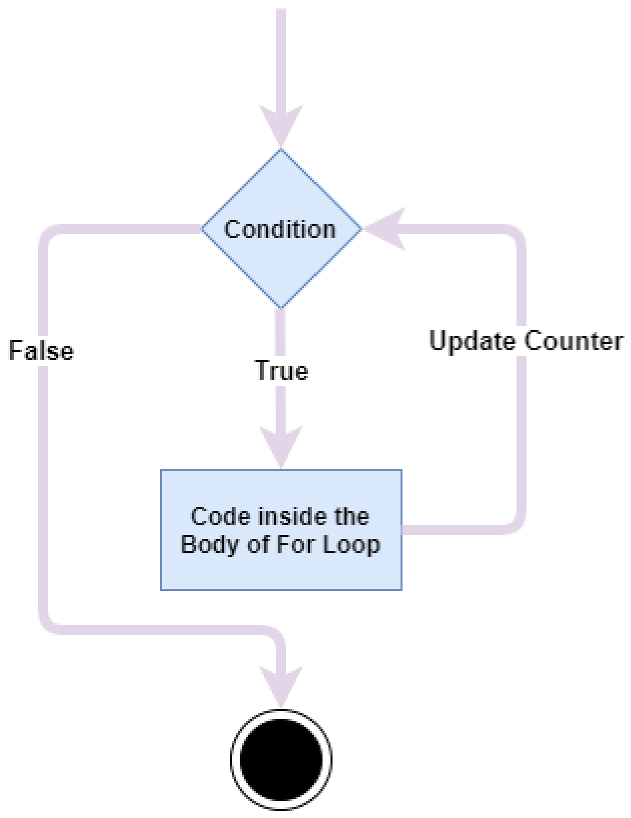
In [29]:
```
a = 10
b = 5
if a != b:
    if a > b:
        print("a is greater than b")
    else:
        print("a is less than b")
```

```
a is greater than b
```

Iteration

For Loop

Loop is a statement that allows us to execute a set of instructions multiple times.



In python, for loop is a statement that executes a set of an instruction multiple times based on the repetitions which are managed by the loop variable.

For loop is mostly used to iterate the sequence such as lists, tuples, and strings.

The example of a for loop is as follows.

In [31]:
```
days = ["Monday","Tueday","Wednesday","Thursday","Friday","Saturday","Sunday"]
for x in days:
      print(x)
```

```
Monday
Tueday
Wednesday
Thursday
Friday
Saturday
Sunday
```

Nested For Loop

Python allows us to use a loop in another loop.

So we can use multiple for loops in another for loop.

The example of nested for loop is as follows.

In [33]:
```
days = ["Monday","Tueday","Wednesday","Thursday","Friday","Saturday","Sunday"]
dayCount = [1]
i = 1
for x in days:
    for y in dayCount:
        print(i,x)
        i+=1
```

```
1 Monday
2 Tueday
3 Wednesday
4 Thursday
5 Friday
6 Saturday
7 Sunday
```

While Loop

In python, while loop is a statement that executes a set of an instruction multiple times based on the condition.

It executes the group of instructions while the condition is TRUE.

In while loop, the condition is checked before entering into the while loop.

The example of a while loop is as follows.

In [39]:
```
i = 1
while i < 6:
    print("Hello World")
    i += 1
```

```
Hello World
Hello World
Hello World
Hello World
Hello World
```

Nested While Loop

Python allows us to use a loop in another loop.

So we can use multiple while loops in another while loop.

The example of nested while loop is as follows.

In [44]:
```
i=1
j=1
while i < 6:
    while j < 6:
        print("Hello World")
        j+=1
    i+=1
```

```
Hello World
Hello World
Hello World
Hello World
Hello World
```