* Python is a simple, general purpose, high level, and object-oriented programming language.
* Python is an interpreted scripting language also. ***Guido Van Rossum*** is known as the founder of Python programming.

**What is Python?**

* Python is a **general purpose**, **dynamic**, **high-level**, and **interpreted programming language**. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.
* Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development.
* Python's syntax and dynamic typing with its interpreted nature make it an ideal language for **scripting** and **rapid application development**.
* Python supports **multiple programming patterns**, including object-oriented, imperative, and functional or procedural programming styles.
* Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD etc.
* Python makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

**Python 2 vs. Python 3 -**

A list of differences between Python 2 and Python 3 are given below:

1. Python 2 uses print as a statement and used as **print "something"** to print some string on the console. On the other hand, Python 3 uses print as a function and used as **print("something")** to print something on the console.
2. Python 2 uses the function **raw\_input()** to accept the user's input. It returns the string representing the value, which is typed by the user. To convert it into the integer, we need to use the int() function in Python. On the other hand, Python 3 uses **input()** function which automatically interpreted the type of input entered by the user. However, we can cast this value to any type by using primitive functions (int(), str(), etc.).
3. In Python 2, the implicit string type is **ASCII**, whereas, in Python 3, the implicit string type is **Unicode**.
4. Python 3 doesn't contain the **xrange()** function of Python 2. The xrange() is the variant of range() function which returns a xrange object that works similar to Java iterator. The **range()** returns a list for example the function range(0,3) contains 0, 1, 2.
5. There is also a small change made in Exception handling in Python 3. It defines a keyword as which is necessary to be used.

**Python Features –**

Python provides lots of features that are listed below.

* **Easy to Learn and Use:** Python is easy to learn and use. It is developer-friendly and high level programming language.
* **Expressive Language:** Python language is more expressive means that it is more understandable and readable.
* **Interpreted Language:** Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.
* **Cross-platform Language:** Python can run equally on different platforms such as Windows, Linux, UNIX and Macintosh etc. So, we can say that Python is a portable language.
* **Free and Open Source:** Python language is freely available at official web address.The source-code is also available. Therefore, it is open source.
* **Object-Oriented Language:** Python supports object oriented language and concepts of classes and objects come into existence.
* **Extensible:** It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.
* **Large Standard Library:** Python has a large and broad library and provides rich set of module and functions for rapid application development.
* **GUI Programming Support:** Graphical user interfaces can be developed using Python.
* **Integrated:** It can be easily integrated with languages like C, C++ and JAVA etc.

**Python History and Versions –**

* Python laid its foundation in the late 1980s.
* The implementation of Python was started in the December 1989 by Guido Van Rossum at CWI in Netherland.
* In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
* In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
* Python 2.0 added new features like: list comprehensions, garbage collection system.
* On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
* ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
* Python is influenced by following programming languages-
  + **ABC language**
  + **Modula-3**

**Python Version List -**

Python programming language is being updated regularly with new features and supports. There are lots of updations in python versions, started from 1994 to current release.A list of python versions with its released date is given below.

|  |  |
| --- | --- |
| **Python Version** | **Released Date** |
| Python 1.0 | January 1994 |
| Python 1.5 | December 31, 1997 |
| Python 1.6 | September 5, 2000 |
| Python 2.0 | October 16, 2000 |
| Python 2.1 | April 17, 2001 |
| Python 2.2 | December 21, 2001 |
| Python 2.3 | July 29, 2003 |
| Python 2.4 | November 30, 2004 |
| Python 2.5 | September 19, 2006 |
| Python 2.6 | October 1, 2008 |
| Python 2.7 | July 3, 2008 |
| Python 3.0 | December 3, 2008 |
| Python 3.1 | June 27, 2009 |
| Python 3.2 | February 20, 2011 |
| Python 3.3 | September 29, 2012 |
| Python 3.4 | March 16, 2014 |
| Python 3.5 | September 13, 2015 |
| Python 3.6 | December 23, 2016 |
| Python 3.7 | June 27, 2018 |

**Python Applications -**

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.Here, we are specifying applications areas where python can be applied.

* **Web Applications -**We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

#### Desktop GUI Applications - Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

* **Software Development -** Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.
* **Scientific and Numeric -** Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.
* **Business Applications -** Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high level application platform.
* **Console Based Application -** We can use Python to develop console based applications. For example: IPython.
* **Audio or Video based Applications -** Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.
* **3D CAD Applications -** To create CAD application Fandango is a real application which provides full features of CAD.
* **Enterprise Applications -** Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.
* **Applications for Images -** Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

**Python Variables –**

* Variable is a name which is used to refer memory location. Variable also known as identifier and used to hold value.
* In Python, we don't need to specify the type of variable because Python is a type infer language and smart enough to get variable type.
* Variable names can be a group of both letters and digits, but they have to begin with a letter or an underscore.
* It is recommended to use lowercase letters for variable name. e.g., Rahul and rahul both are two different variables.

**Identifier Naming -**

Variables are the example of identifiers. An Identifier is used to identify the literals used in the program. The rules to name an identifier are given below –

* The first character of the variable must be an alphabet or underscore ( \_ ).
* All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore or digit (0-9).
* Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, \*).
* Identifier name must not be similar to any keyword defined in the language.
* Identifier names are case sensitive for example my name, and MyName is not the same.

**Examples of valid identifiers:** a123, \_n, n\_9, etc.

**Examples of invalid identifiers:** 1a, n%4, n 9, etc.

**Declaring Variable and Assigning Values –**

Python does not bind us to declare variable before using in the application. It allows us to create variable at required time.We don't need to declare explicitly variable in Python. When we assign any value to the variable that variable is declared automatically.**The equal (=) operator is used to assign value to a variable.**

**Multiple Assignments –**

Python allows us to assign a value to multiple variables in a single statement which is also known as multiple assignments. We can apply multiple assignments in two ways either by assigning a single value to multiple variables or assigning multiple values to multiple variables.

* **Assigning single value to multiple variables –**

**x=y=z=50**

**print (x)**

**print (y)**

**print (z)**

* **Assigning multiple values to multiple variables –**

**a,b,c=5,10,15**

**print (a)**

**print (b)**

**print (c)**

**Basic Fundamentals –**

This section contains the basic fundamentals of Python like –

1. **Tokens and their types**
2. **Comments**

**Tokens -**

Tokens can be defined as a punctuator mark, reserved words and each individual word in a statement.Token is the smallest unit inside the given program.

There are following tokens in Python:

* **Keywords**
* **Identifiers**
* **Literals**
* **Operators**

**Python Data Types –**

Variables can hold values of different data types. **Python is a dynamically typed language** hence we need not define the type of the variable while declaring it. The interpreter implicitly binds the value with its type.Python enables us to check the type of the variable used in the program. **Python provides us the *type()* function which returns the type of the variable passed.**

**e.g,**

**A=10**

**b="Hi Python"**

**print(type(a));**

**print(type(b));**

**Standard data types -**

A variable can hold different types of values. Python provides various standard data types that define the storage method on each of them. The data types defined in Python are given below.

1. Numbers
2. String
3. List
4. Tuple
5. Dictionary

**Python Keywords –**

Python Keywords are special reserved words which convey a special meaning to the compiler/interpreter. Each keyword has a special meaning and a specific operation. These keywords can't be used as variable.

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

**Python Literals -**

Literals can be defined as a data that is given in a variable or constant.Python support the following literals –

* **String literals:** String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes for a String.

e.g. "Aman" , '12345'

* + **Types of Strings:**There are two types of Strings supported in Python -
    - **Single line String-** Strings that are terminated within a single line are known as Single line Strings.

e**.**g. **text1='hello'**

* + - **Multi line String -** A piece of text that is spread along multiple lines is known as multiple line String.There are two ways to create Multiline Strings –
* **Adding black slash at the end of each line –**

**text1='hello\**

**user'**

* **Using triple quotation marks -**

**c = '''Welcome**

**to GGI'''**

* **Numeric Literals –** Numeric Literals are **immutable**. Numeric literals can belong to following four different numerical types.
  + **Int (Signed Integers) –** Numbers(can be both positive and negative) with no fractional part. e.g.: 100
  + **Long (Long Integers) -**Integers of unlimited size followed by lowercase or uppercase L e.g.: 87032845L
  + **Float (Floating Point) -**Real numbers with both integer and fractional part e.g.: 26.2
  + **Complex (complex) -**In the form of a+bj where a forms the real part and b forms the imaginary part of complex number. e.g.: 3.14j
* **Boolean Literals -**A Boolean literal can have any of the two values: True or False.
* **Special Literals -** Python contains one special literal i.e., **None**. None is used to specify to that field that is not created. It is also used for end of lists in Python.

**g = 10**

**h = None**

**print (g)**

**print (h)**

* **Literal Collections -** Collections such as tuples, lists and Dictionary are used in Python.

**Python Operators –**

The operator can be defined as a symbol which is responsible for a particular operation between two operands. Operators are the pillars of a program on which the logic is built in a particular programming language.Python provides a variety of operators described as follows.

* **Arithmetic operators**
* **Comparison operators**
* **Assignment Operators**
* **Logical Operators**
* **Bitwise Operators**
* **Membership Operators**
* **Identity Operators**

**Arithmetic Operators –**

|  |  |
| --- | --- |
| Operator | Description |
| + (Addition) | It is used to add two operands. For example, if a = 20, b = 10 => a+b = 30 |
| - (Subtraction) | It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value result negative. For example, if a = 20, b = 10 => a - b = 10 |
| / (divide) | It returns the quotient after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a/b = 2 |
| \* (Multiplication) | It is used to multiply one operand with the other. For example, if a = 20, b = 10 => a \* b = 200 |
| % (reminder) | It returns the reminder after dividing the first operand by the second operand. For example, if a = 20, b = 10 => a%b = 0 |
| \*\* (Exponent) | It is an exponent operator represented as it calculates the first operand power to second operand. |
| // (Floor division) | It gives the floor value of the quotient produced by dividing the two operands. |

**Comparison Operators –**

|  |  |
| --- | --- |
| Operator | Description |
| == | If the value of two operands is equal, then the condition becomes true. |
| != | If the value of two operands is not equal then the condition becomes true. |
| <= | If the first operand is less than or equal to the second operand, then the condition becomes true. |
| >= | If the first operand is greater than or equal to the second operand, then the condition becomes true. |
| > | If the first operand is greater than the second operand, then the condition becomes true. |
| < | If the first operand is less than the second operand, then the condition becomes true. |

**Assignment Operators -**

|  |  |
| --- | --- |
| Operator | Description |
| = | It assigns the the value of the right expression to the left operand. |
| += | It increases the value of the left operand by the value of the right operand and assign the modified value back to left operand. For example, if a = 10, b = 20 => a+ = b will be equal to a = a+ b and therefore, a = 30. |
| -= | It decreases the value of the left operand by the value of the right operand and assign the modified value back to left operand. For example, if a = 20, b = 10 => a- = b will be equal to a = a- b and therefore, a = 10. |
| \*= | It multiplies the value of the left operand by the value of the right operand and assign the modified value back to left operand. For example, if a = 10, b = 20 => a\* = b will be equal to a = a\* b and therefore, a = 200. |
| %= | It divides the value of the left operand by the value of the right operand and assign the reminder back to left operand. For example, if a = 20, b = 10 => a % = b will be equal to a = a % b and therefore, a = 0. |
| \*\*= | a\*\*=b will be equal to a=a\*\*b, for example, if a = 4, b =2, a\*\*=b will assign 4\*\*2 = 16 to a. |
| //= | A//=b will be equal to a = a// b, for example, if a = 4, b = 3, a//=b will assign 4//3 = 1 to a. |

**Bitwise Operator –**

|  |  |
| --- | --- |
| Operator | Description |
| & (binary and) | If both the bits at the same place in two operands are 1, then 1 is copied to the result. Otherwise, 0 is copied. |
| | (binary or) | The resulting bit will be 0 if both the bits are zero otherwise the resulting bit will be 1. |
| ^ (binary xor) | The resulting bit will be 1 if both the bits are different otherwise the resulting bit will be 0. |
| ~ (negation) | It calculates the negation of each bit of the operand, i.e., if the bit is 0, the resulting bit will be 1 and vice versa. |
| << (left shift) | The left operand value is moved left by the number of bits present in the right operand. |
| >> (right shift) | The left operand is moved right by the number of bits present in the right operand. |

**Logical Operators –**

|  |  |
| --- | --- |
| Operator | Description |
| and | If both the expression are true, then the condition will be true. If a and b are the two expressions, a → true, b → true => a and b → true. |
| or | If one of the expressions is true, then the condition will be true. If a and b are the two expressions, a → true, b → false => a or b → true. |
| not | If an expression **a** is true then not (a) will be false and vice versa. |

**Membership Operators –**

|  |  |
| --- | --- |
| Operator | Description |
| In | It is evaluated to be true if the first operand is found in the second operand (list, tuple, or dictionary). |
| not in | It is evaluated to be true if the first operand is not found in the second operand (list, tuple, or dictionary). |

**Identity Operators –**

|  |  |
| --- | --- |
| Operator | Description |
| is | It is evaluated to be true if the reference present at both sides point to the same object. |
| is not | It is evaluated to be true if the reference present at both side do not point to the same object. |

**Operator Precedence –**

|  |  |
| --- | --- |
| Operator | Description |
| \*\* | The exponent operator is given priority over all the others used in the expression. |
| ~ + - | The negation, unary plus, and minus. |
| \* / % // | The multiplication, divide, modules, reminder, and floor division. |
| + - | Binary plus, and minus |
| >><< | Left shift. and right shift |
| & | Binary and. |
| ^ | | Binary xor, and or |
| <= <>>= | Comparison operators (less than, less than equal to, greater than, greater then equal to). |
| <> == != | Equality operators. |
| = %= /= //= -= += \*= \*\*= | Assignment operators |
| is is not | Identity operators |
| in not in | Membership operators |
| not or and | Logical operators |

**Python Comments –**

* Comments in Python can be used to explain any program code. It can also be used to hide the code as well.
* Comments are the most helpful stuff of any program. It enables us to understand the way, a program works. In python, any statement written along with # symbol is known as a comment. The interpreter does not interpret the comment.
* Comment is not a part of the program, but it enhances the interactivity of the program and makes the program readable.

Python supports two types of comments –

1. **Single Line Comment –**

**# This is single line comment.**

**print "Hello Python"**

1. **Multi Line Comment –**

**''' This**

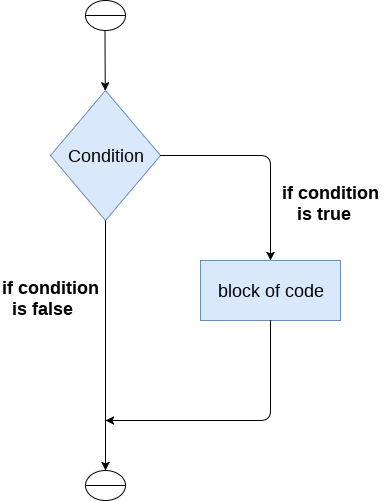
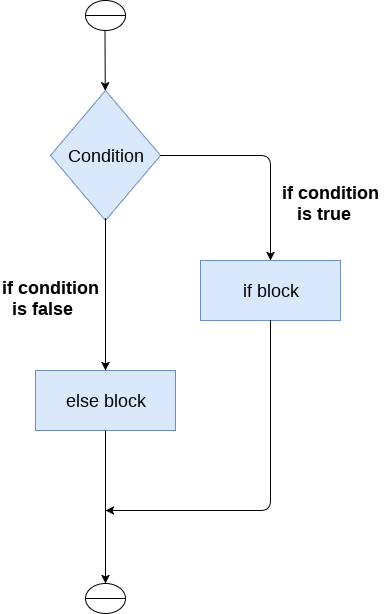
**Is**

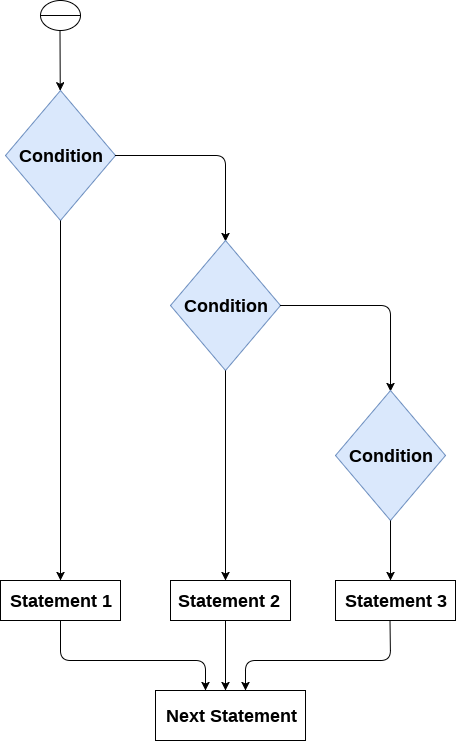
**Multipline comment'''**

**Decision making–**

Decision making is the most important aspect of almost all the programming languages. As the name implies, decision making allows us to run a particular block of code for a particular decision. Here, the decisions are made on the validity of the particular conditions. Condition checking is the backbone of decision making.

|  |  |
| --- | --- |
| Statement | Description |
|  |  |
| If Statement | The if statement is used to test a specific condition. If the condition is true, a block of code (if-block) will be executed. |
| If - else Statement | The if-else statement is similar to if statement except the fact that, it also provides the block of the code for the false case of the condition to be checked. If the condition provided in the if statement is false, then the else statement will be executed. |
| Nested if Statement | Nested if statements enable us to use if ? else statement inside an outer if statement. |



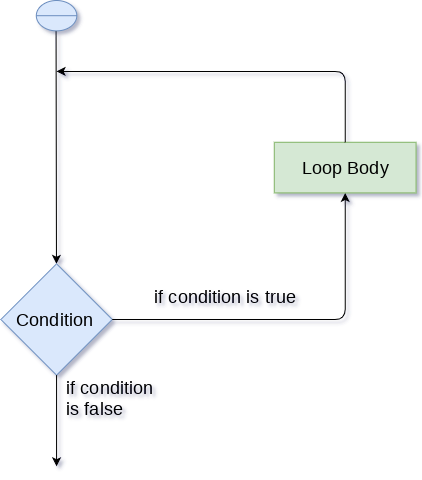


**Loops -**The flow of the programs written in any programming language is sequential by default. Sometimes we may need to alter the flow of the program. The execution of a specific code may need to be repeated several numbers of times.

**Advantages of loops -**There are the following advantages of loops in Python -

* It provides code re-usability.
* Using loops, we do not need to write the same code again and again.
* Using loops, we can traverse over the elements of data structures (array or linked lists)

|  |  |
| --- | --- |
| Loop Statement | Description |
| for loop | * The for loop is used in the case where we need to execute some part of the code until the given condition is satisfied. * The for loop is also called as a per-tested loop. It is better to use for loop if the number of iteration is known in advance. |
| while loop | * The while loop is to be used in the scenario where we don't know the number of iterations in advance. * The block of statements is executed in the while loop until the condition specified in the while loop is satisfied. It is also called a **pre-tested loop**. |
| do-while loop | * The do-while loop continues until a given condition satisfies. It is also called post tested loop. * Python does not have any do-while loop |



**For Loops -** The for loop in Python is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like list, tuple, or dictionary.

**Syntax - for iterating\_var in sequence:**

**statement(s)**

**For Loop using range() function –**

The range() function is used to generate the sequence of the numbers. If we pass the range(10), it will generate the numbers from 0 to 9. The syntax of the range() function is given below.

**Syntax -** **range(start,stop,step size)**

* The start represents the beginning of the iteration.
* The stop represents that the loop will iterate till stop-1. The **range(1,5)** will generate numbers 1 to 4 iterations. It is optional.
* The step size is used to skip the specific numbers from the iteration. It is optional to use. By default, the step size is 1. It is optional.

**Nested for loop in python -**Python allows us to nest any number of for loops inside a **for** loop. The inner loop is executed n number of times for every iteration of the outer loop.

**Syntax -** **for iterating\_var1 in sequence: #outer loop**

**for iterating\_var2 in sequence: #inner loop**

**#block of statements**

**#other statements**

**While Loop -**The Python while loop allows a part of the code to be executed until the given condition returns false. It is also known as a pre-tested loop.

**Syntax - while expression:**

**Statements**

**Infinite Loop -**If the condition is given in loop that never becomes false, then the loop will never terminate, and it turns into the infinite loop.

**Syntax -while (1):**

**print("Hi! we are inside the infinite while loop")**

**Using else with For Loop & While Loop -**Python allows us to use the else statement with the for loop and while loop which can be executed only when all the iterations are exhausted.

* **For Loop –**

**for i in range(0,5):**

**print(i)**

**else:**

**print("for loop completely exhausted...")**

* **While Loop -**

**i=1**

**while(i<=5):**

**print(i)**

**i=i+1**

**else:**

**print("The while loop exhausted")**

**Break Statement -**The break is a keyword in python which is used to bring the program control out of the loop. The break statement breaks the loops one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops.

**Syntax - #loop statements**

**break;**

**Example -    list =[1,2,3,4]**

**count = 1;**

**for i in list:**

**if i == 4:**

**print("item matched")**

**count = count + 1;**

**break;**

**print("found at",count,"location");**

**Example -*break statement with while loop***

**i = 0;**

**while 1:**

**print(i," ",end=""),**

**i=i+1;**

**if i == 10:**

**break;**

**print("came out of while loop");**

**Continue -**The continue statement in Python is used to bring the program control to the beginning of the loop. The continue statement skips the remaining lines of code inside the loop and start with the next iteration. It is mainly used for a particular condition inside the loop so that we can skip some specific code for a particular condition.The continue statement in Python is used to bring the program control to the beginning of the loop.

**Syntax –**

**#loop statements**

**continue**

**#the code to be skipped**

**Example –**

**i = 0**

**while(i < 10):**

**i = i+1**

**if(i == 5):**

**continue**

**print(i)**

**Pass Statement –** The pass statement is a null operation since nothing happens when it is executed. It is used in the cases where a statement is syntactically needed but we don't want to use any executable statement at its place.Pass is also used where the code will be written somewhere but not yet written in the program file.

**list = [1,2,3,4,5]**

**flag = 0**

**for i in list:**

**print("Current element:",i,end=" ");**

**if i==3:**

**pass**

**print("\nWe are inside pass block\n");**

**flag = 1**

**if flag==1:**

**print("\nCame out of pass\n");**

**flag=0**