# Python Introduction

**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 History and Versions

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

<https://www.python.org/doc/versions/>

# Python Features

Python provides lots of features that are listed below.

### 1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high-level programming language.

### 2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

### 3) 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.

### 4) 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.

### 5) Free and Open Source

Python language is freely available at address. The source-code is also available. Therefore, it is open source.

### 6) Object-Oriented Language

Python supports object-oriented language and concepts of classes and objects come into existence.

### 7) 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.

### 8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

### 9) GUI Programming Support

Graphical user interfaces can be developed using Python.

### 10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

# Python Applications

### 1) 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, beautiful Soup, Feed parser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develops web-based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

### 2) 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.

### 3) 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.

### 4) 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.

### 5) Business Applications

Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high-level application platform.

### 6) Console Based Application

We can use Python to develop console-based applications. For example: **IPython**.

### 7) 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.

### 8) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

### 9) 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.

### 10) Applications for Images

Using Python several applications 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. 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 bound 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.

a=10;

name=” Debasish”;

marks=50.50;

Print a;

Print name;

Print marks;

## Multiple Assignment

Python allows us to assign a value to multiple variables in a single statement which is also known as multiple assignment.

We can apply multiple assignments in two ways either by assigning a single value to multiple variables or assigning multiple values to multiple variables.

There is Two Way we can Assign Value in Python

* Assigning single value to multiple variables
* Assigning multiple values to multiple variables

Python Numbers

There are three numeric types in Python:

* int
* float
* complex

x = 1    # int  
y = 2.8  # float  
z = 1j   # complex

To verify the type of any object in Python, use the type () function

Ex- Number.py

## Int

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

## Float

Float, or "floating point number" is a number, positive or negative, containing one or more decimals.Float can also be scientific numbers with an "e" to indicate the power of 10.

## Complex

Complex numbers are written with a "j" as the imaginary part

# Python Casting

Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

* int() - constructs an integer number from an integer literal, a float literal (by rounding down to the previous whole number), or a string literal (providing the string represents a whole number)
* float() - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
* str() - constructs a string from a wide variety of data types, including strings, integer literals and float literals

### **Python String**

The string can be defined as the sequence of characters represented in the quotation marks. In python, we can use single, double, or triple quotes to define a string.

String handling in python is a straightforward task since there are various inbuilt functions and operators provided.

In the case of string handling, the operator + is used to concatenate two strings as the operation "hello"+" python"returns "hello python".

The operator \* is known as repetition operator as the operation "Python " \*2 returns "Python Python ".

String.py

## Command-line String Input

Python allows for command line input.

That means we are able to ask the user for input.

The following example asks for the user's name, then, by using the input() method, the program prints the name to the screen:

CLIINPUTSTRING.py

# **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.

* 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 results 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.If you imagine a room where 3 is on the ceiling and 2 is on the floor. 2.5 would fit in the middle. Floor division means the "//" will always take the floor or the lower number. |

## assignment operators

The assignment operators are used to assign the value of the right expression to the left operand. The assignment operators are described in the following table.

|  |  |
| --- | --- |
| **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. |

## Comparison operator

Comparison operators are used to comparing the value of the two operands and returns boolean true or false accordingly. The comparison operators are described in the following table.

|  |  |
| --- | --- |
| 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 value of two operands is not equal, 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. |

|  |  |
| --- | --- |
| **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 value of two operands is not equal, 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. |

## Bitwise operator

The bitwise operators perform bit by bit operation on the values of the two operands.

|  |  |
| --- | --- |
| **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

The logical operators are used primarily in the expression evaluation to make a decision. Python supports the following 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

Python membership operators are used to check the membership of value inside a data structure. If the value is present in the data structure, then the resulting value is true otherwise it returns false.

|  |  |
| --- | --- |
| **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

The precedence of the operators is important to find out since it enables us to know which operator should be evaluated first. The precedence table of the operators in python is given below.

|  |  |
| --- | --- |
| **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 then, less then equal to, greater then, greater then equal to). |
| <> == != | Equality operators. |
| = %= /= //= -= += \*= \*\*= | Assignment operators |
| is is not | Identity operators |
| in not in | Membership operators |
| not or and | Logical operators |

## Python Collections (Arrays)

There are four collection data types in the Python programming language:

* **List** is a collection which is ordered and changeable. Allows duplicate members.
* **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
* **Set** is a collection which is unordered and unindexed. No duplicate members.
* **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

### **List**

Lists are similar to arrays in C. However; the list can contain data of different types. The items stored in the list are separated with a comma (,) and enclosed within square brackets [].

We can use slice [:] operators to access the data of the list. The concatenation operator (+) and repetition operator (\*) works with the list in the same way as they were working with the strings.

list.py

### **Tuple**

A tuple is similar to the list in many ways. Like lists, tuples also contain the collection of the items of different data types. The items of the tuple are separated with a comma (,) and enclosed in parentheses ().

A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.A tuple is a collection which is ordered and **unchangeable**. In Python tuples are written with round brackets.

tupal.py

# Set

A set is a collection which is unordered and unindexed. In Python sets are written with curly brackets

### **Dictionary**

Dictionary is an ordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. Key can hold any primitive data type whereas value is an arbitrary Python object.

The items in the dictionary are separated with the comma and enclosed in the curly braces {}.

# **Python Keywords**

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |

|  |  |
| --- | --- |
| 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. |

Indentation in Python

For the ease of programming and to achieve simplicity, python doesn't allow the use of parentheses for the block level code. In Python, indentation is used to declare a block. If two statements are at the same indentation level, then they are the part of the same block.

Generally, four spaces are given to indent the statements which are a typical amount of indentation in python.

Indentation is the most used part of the python language since it declares the block of code. All the statements of one block are intended at the same level indentation. We will see how the actual indentation takes place in decision making and other stuff in python.

# **Python Functions**

Functions are the most important aspect of an application. A function can be defined as the organized block of reusable code which can be called whenever required.

Python allows us to divide a large program into the basic building blocks known as function. The function contains the set of programming statements enclosed by {}. A function can be called multiple times to provide reusability and modularity to the python program.

In other words, we can say that the collection of functions creates a program. The function is also known as procedure or subroutine in other programming languages.

Python provide us various inbuilt functions like range() or print(). Although, the user can create its functions which can be called user-defined functions.

Call by reference in Python

In python, all the functions are called by reference, i.e., all the changes made to the reference inside the function revert back to the original value referred by the reference.

However, there is an exception in the case of mutable objects since the changes made to the mutable objects like string do not revert to the original string rather, a new string object is made, and therefore the two different objects are printed.

Types of arguments

There may be several types of arguments which can be passed at the time of function calling.

1. Required arguments
2. Keyword arguments
3. Default arguments
4. Variable-length arguments

Required Arguments

Till now, we have learned about function calling in python. However, we can provide the arguments at the time of function calling. As far as the required arguments are concerned, these are the arguments which are required to be passed at the time of function calling with the exact match of their positions in the function call and function definition. If either of the arguments is not provided in the function call, or the position of the arguments is changed, then the python interpreter will show the error.

Keyword arguments

Python allows us to call the function with the keyword arguments. This kind of function call will enable us to pass the arguments in the random order.

The name of the arguments is treated as the keywords and matched in the function calling and definition. If the same match is found, the values of the arguments are copied in the function definition.

Default Arguments

Python allows us to initialize the arguments at the function definition. If the value of any of the argument is not provided at the time of function call, then that argument can be initialized with the value given in the definition even if the argument is not specified at the function call.

Variable length Arguments

In the large projects, sometimes we may not know the number of arguments to be passed in advance. In such cases, Python provides us the flexibility to provide the comma separated values which are internally treated as tuples at the function call.

However, at the function definition, we have to define the variable with \* (star) as \*<variable - name >.

Scope of variables

The scopes of the variables depend upon the location where the variable is being declared. The variable declared in one part of the program may not be accessible to the other parts.

In python, the variables are defined with the two types of scopes.

1. Global variables
2. Local variables

The variable defined outside any function is known to have a global scope whereas the variable defined inside a function is known to have a local scope.

# **Python Lambda Functions**

Python allows us to not declare the function in the standard manner, i.e., by using the def keyword. Rather, the anonymous functions are declared by using lambda keyword. However, Lambda functions can accept any number of arguments, but they can return only one value in the form of expression.

The anonymous function contains a small piece of code. It simulates inline functions of C and C++, but it is not exactly an inline function.

Why use lambda functions?

The main role of the lambda function is better described in the scenarios when we use them anonymously inside another function. In python, the lambda function can be used as an argument to the higher order functions as arguments.

# **Python File Handling**

Till now, we were taking the input from the console and writing it back to the console to interact with the user.

Sometimes, it is not enough to only display the data on the console. The data to be displayed may be very large, and only a limited amount of data can be displayed on the console, and since the memory is volatile, it is impossible to recover the programmatically generated data again and again.

However, if we need to do so, we may store it onto the local file system which is volatile and can be accessed every time. Here, comes the need of file handling.

In this section of the tutorial, we will learn all about file handling in python including, creating a file, opening a file, closing a file, writing and appending the file, etc.

|  |  |  |
| --- | --- | --- |
| SN | Access mode | Description |
| 1 | r | It opens the file to read-only. The file pointer exists at the beginning. The file is by default open in this mode if no access mode is passed. |
| 2 | rb | It opens the file to read only in binary format. The file pointer exists at the beginning of the file. |
| 3 | r+ | It opens the file to read and write both. The file pointer exists at the beginning of the file. |
| 4 | rb+ | It opens the file to read and write both in binary format. The file pointer exists at the beginning of the file. |
| 5 | w | It opens the file to write only. It overwrites the file if previously exists or creates a new one if no file exists with the same name. The file pointer exists at the beginning of the file. |
| 6 | wb | It opens the file to write only in binary format. It overwrites the file if it exists previously or creates a new one if no file exists with the same name. The file pointer exists at the beginning of the file. |
| 7 | w+ | It opens the file to write and read both. It is different from r+ in the sense that it overwrites the previous file if one exists whereas r+ doesn?t overwrite the previously written file. It creates a new file if no file exists. The file pointer exists at the beginning of the file. |
| 8 | wb+ | It opens the file to write and read both in binary format. The file pointer exists at the beginning of the file. |
| 9 | a | It opens the file in the append mode. The file pointer exists at the end of the previously written file if exists any. It creates a new file if no file exists with the same name. |
| 10 | ab | It opens the file in the append mode in binary format. The pointer exists at the end of the previously written file. It creates a new file in binary format if no file exists with the same name. |
| 11 | a+ | It opens a file to append and read both. The file pointer remains at the end of the file if a file exists. It creates a new file if no file exists with the same name. |
| 12 | ab+ | It opens a file to append and read both in binary format. The file pointer remains at the end of the file. |

# **Python Modules**

A python module can be defined as a python program file which contains a python code including python functions, class, or variables. In other words, we can say that our python code file saved with the extension (.py) is treated as the module. We may have a runnable code inside the python module.

Modules in Python provides us the flexibility to organize the code in a logical way.

Python packages

The packages in python facilitate the developer with the application development environment by providing a hierarchical directory structure where a package contains sub-packages, modules, and sub-modules. The packages are used to categorize the application level code efficiently.

# **Python Exceptions**

An exception can be defined as an abnormal condition in a program resulting in the disruption in the flow of the program.

Whenever an exception occurs, the program halts the execution, and thus the further code is not executed. Therefore, an exception is the error which python script is unable to tackle with.

Python provides us with the way to handle the Exception so that the other part of the code can be executed without any disruption. However, if we do not handle the exception, the interpreter doesn't execute all the code that exists after the that.

## Common Exceptions

A list of common exceptions that can be thrown from a normal python program is given below.

1. **ZeroDivisionError:** Occurs when a number is divided by zero.
2. **NameError:** It occurs when a name is not found. It may be local or global.
3. **IndentationError:** If incorrect indentation is given.
4. **IOError:** It occurs when Input Output operation fails.
5. **EOFError:** It occurs when the end of the file is reached, and yet operations are being performed.

Exception handling in python

If the python program contains suspicious code that may throw the exception, we must place that code in the try block. The try block must be followed with the except statement which contains a block of code that will be executed if there is some exception in the try block.

We can also use the else statement with the try-except statement in which, we can place the code which will be executed in the scenario if no exception occurs in the try block.

# **Python OOPs Concepts**

Like other general purpose languages, python is also an object-oriented language since its beginning. Python is an object-oriented programming language. It allows us to develop applications using an Object Oriented approach. In Python, we can easily create and use classes and objects.

Major principles of object-oriented programming system are given below.

* Object
* Class
* Method
* Inheritance
* Polymorphism
* Data Abstraction
* Encapsulation

Object

The object is an entity that has state and behavior. It may be any real-world object like the mouse, keyboard, chair, table, pen, etc.

Everything in Python is an object, and almost everything has attributes and methods. All functions have a built-in attribute \_\_doc\_\_, which returns the doc string defined in the function source code.

Class

The class can be defined as a collection of objects. It is a logical entity that has some specific attributes and methods. For example: if you have an employee class then it should contain an attribute and method, i.e. an email id, name, age, salary, etc.

Method

The method is a function that is associated with an object. In Python, a method is not unique to class instances. Any object type can have methods.

Inheritance

Inheritance is the most important aspect of object-oriented programming which simulates the real world concept of inheritance. It specifies that the child object acquires all the properties and behaviors of the parent object.

By using inheritance, we can create a class which uses all the properties and behavior of another class. The new class is known as a derived class or child class, and the one whose properties are acquired is known as a base class or parent class.

It provides re-usability of the code.

Polymorphism

Polymorphism contains two words "poly" and "morphs". Poly means many and Morphs means form, shape. By polymorphism, we understand that one task can be performed in different ways. For example You have a class animal, and all animals speak. But they speak differently. Here, the "speak" behavior is polymorphic in the sense and depends on the animal. So, the abstract "animal" concept does not actually "speak", but specific animals (like dogs and cats) have a concrete implementation of the action "speak".

Encapsulation

Encapsulation is also an important aspect of object-oriented programming. It is used to restrict access to methods and variables. In encapsulation, code and data are wrapped together within a single unit from being modified by accident.

Data Abstraction

Data abstraction and encapsulation both are often used as synonyms. Both are nearly synonym because data abstraction is achieved through encapsulation.

Abstraction is used to hide internal details and show only functionalities. Abstracting something means to give names to things so that the name captures the core of what a function or a whole program does.

# **Python Constructor**

A constructor is a special type of method (function) which is used to initialize the instance members of the class.

Constructors can be of two types.

1. Parameterized Constructor
2. Non-parameterized Constructor

Constructor definition is executed when we create the object of this class. Constructors also verify that there are enough resources for the object to perform any start-up task.

Creating the constructor in python

In python, the method \_\_**init**\_\_ simulates the constructor of the class. This method is called when the class is instantiated. We can pass any number of arguments at the time of creating the class object, depending upon \_\_**init**\_\_ definition. It is mostly used to initialize the class attributes. Every class must have a constructor, even if it simply relies on the default constructor.

Python In-built class functions

|  |  |  |
| --- | --- | --- |
| SN | Function | Description |
| 1 | getattr(obj,name,default) | It is used to access the attribute of the object. |
| 2 | setattr(obj, name,value) | It is used to set a particular value to the specific attribute of an object. |
| 3 | delattr(obj, name) | It is used to delete a specific attribute. |
| 4 | hasattr(obj, name) | It returns true if the object contains some specific attribute. |

Built-in class attributes

|  |  |  |
| --- | --- | --- |
| SN | Attribute | Description |
| 1 | \_\_dict\_\_ | It provides the dictionary containing the information about the class namespace. |
| 2 | \_\_doc\_\_ | It contains a string which has the class documentation |
| 3 | \_\_name\_\_ | It is used to access the class name. |
| 4 | \_\_module\_\_ | It is used to access the module in which, this class is defined. |
| 5 | \_\_bases\_\_ | It contains a tuple including all base classes. |

# **Python Inheritance**

Inheritance is an important aspect of the object-oriented paradigm. Inheritance provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch.

In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class. A child class can also provide its specific implementation to the functions of the parent class. In this section of the tutorial, we will discuss inheritance in detail.

In python, a derived class can inherit base class by just mentioning the base in the bracket after the derived class name. Consider the following syntax to inherit a base class into the derived class.

BASE CLASS

DERIVE CLASS

Python Multi-Level inheritance

Multi-Level inheritance is possible in python like other object-oriented languages. Multi-level inheritance is archived when a derived class inherits another derived class. There is no limit on the number of levels up to which, the multi-level inheritance is archived in python.

Class1

Class2

classN

Python Multiple inheritance

CLASS1 CLASS2 CLASS3 CLASSN

DERIVED CLASS

The issubclass(sub,sup) method

The issubclass(sub, sup) method is used to check the relationships between the specified classes. It returns true if the first class is the subclass of the second class, and false otherwise.

The isinstance (obj, class) method

The isinstance() method is used to check the relationship between the objects and classes. It returns true if the first parameter, i.e., obj is the instance of the second parameter, i.e., class.

Method Overriding

We can provide some specific implementation of the parent class method in our child class. When the parent class method is defined in the child class with some specific implementation, then the concept is called method overriding. We may need to perform method overriding in the scenario where the different definition of a parent class method is needed in the child class.

Data abstraction in python

Abstraction is an important aspect of object-oriented programming. In python, we can also perform data hiding by adding the double underscore (\_\_\_) as a prefix to the attribute which is to be hidden. After this, the attribute will not be visible outside of the class through the object.

# **Python Tkinter**

Python provides the standard library for creating the graphical user interface for desktop based applications.

Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

## Tkinter widgets

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

|  |  |  |
| --- | --- | --- |
| **SN** | **Widget** | **Description** |
| 1 | Button | The Button is used to add various kinds of buttons to the python application. |
| 2 | Canvas | The canvas widget is used to draw the canvas on the window. |
| 3 | Check button | The Checkbutton is used to display the CheckButton on the window. |
| 4 | Entry | The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values. |
| 5 | Frame | It can be defined as a container to which, another widget can be added and organized. |
| 6 | Label | A label is a text used to display some message or information about the other widgets. |
| 7 | List Box | The ListBox widget is used to display a list of options to the user. |
| 8 | Menu button | The Menubutton is used to display the menu items to the user. |
| 9 | Menu | It is used to add menu items to the user. |
| 10 | Message | The Message widget is used to display the message-box to the user. |
| 11 | Radio button | The Radiobutton is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them. |
| 12 | Scale | It is used to provide the slider to the user. |
| 13 | Scrollbar | It provides the scrollbar to the user so that the user can scroll the window up and down. |
| 14 | Text | It is different from Entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it. |
| 14 | Top level | It is used to create a separate window container. |
| 15 | Spin box | It is an entry widget used to select from options of values. |
| 16 | Paned Window | It is like a container widget that contains horizontal or vertical panes. |
| 17 | Label Frame | A LabelFrame is a container widget that acts as the container |
| 18 | Message Box | This module is used to display the message-box in the desktop based applications. |

# Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

The pack() method

The grid() method

The place() method

### **Python Tkinter pack() method**

The pack() widget is used to organize widget in the block. The positions widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call.

However, the controls are less and widgets are generally added in the less organized manner.