* **Python** is a high-level, general-purpose and very popular programming language. It is being used in web development, machine learning applications and software development.
* **Python Indentation:** It refers to the space at the beginning of the code. In

python, it indicates a block of code.

Example: if 3>2:

print(“Three is greater than Two.”)

* **Python Comments:** It is used to explain the code. It makes the code more readable. It starts with a #.
* # This is a comment

print(“Hello World”)

* “””

This is also a comment. But

Here we can write multiple lines.

“””

* **Python Variables:** It likes a container that stores data values.
* X = 5
* print(x)
* Variable name convention:

1. It must start with a letter and the underscore character.
2. It cannot start with any number.
3. It can contain only alpha-numeric characters and underscores.
4. It is case-sensitive.

* **Python Data Types:**
* Text Type: str
* Numerical Types: int, float, complex
* Sequence Type: list, tuple, range
* Mapping Type: dict
* Set Types: set, frozenset
* Boolean Type: bool

Example:

String : X = “Hello World”

Integer: Y = 6

Float: Z = 2.5

List: my\_list = [“apple”,”banana”,”cherry”]

Tuple: my\_tuple = (1,7,3,9,5)

Range: rang = range(5)

Dictionary: my\_dict = {1:”January”, 2:”February”, 3:”March”, 4:”April”}

Set: my\_set = {“abc”, 34, True, 40, “Male”}

Frozenset: fruit\_list = frozenset([“apple”,”banana”,”orange”])

Bool: print(bool(“Hello”)) -> Output: True

* **Python Functions**
* A function is a block of code which only runs when it is called.

we can pass data, known as parameters, into a function.

def my\_function():

print(“This is a function”)

* *Lambda Function:* A Lambda function is a small anonymous function.

It can take any number of arguments, but can only have one expression.

Example: X = lambda : a,b,c : a+b+c

print(X(2,4,5))

* *Argument of a Function:* An argument is the value that is sent to the function when it is called.
* *Types of Arguments:*

1. Default Argument: It is a parameter that assumes a default value if a value is not provided in the function call for that argument.

def my\_func(x, y=50):

print("x: ", x)

print("y: ", y)

def(10)

Here the default value for the argument y is 50.

1. Arbitrary Keyword Arguments: In python, \*args and \*\*kwargs can pass a variable number of arguments to a function using special symbols.

def myFunc(\*argv):

for arg in argv:

print(arg)

myFunc(“I” ,”am”,”a”,”student”)

* **Difference between list and tuple.**
* List is mutable(i.e, after creating a list, elements can be changed).
* Tuples are immutable(i.e, after creating a tuple, elements can not be changed.
* List is better for performing the operations, such as insertion, deletion.

Tuple is appropriate for accessing the elements.

* Tuples consume more memory as compared to the list.
* Lists have several built-in methods.

Tuples does not have many built-in methods.

* Example of list is [1,5,9,3].

Example of tuple is (5,9,2)

* **Difference between set and frozenset.**
* A set is a collection of unordered , unindexed and unique elements.

A frozenset is an unordered and unindexed collection of unique

elements.

* Sets are mutable.We can change the elements using a built-in function like add(), remove(),etc.Since the elements are mutable and not in order, they don’t have hash values.
* Frozenset is immutable and hashable. Since the elements are fixed, we can't add or remove elements from the set.
* Sets can’t be used as a dictionary key or as elements of another set. They can be used as a dictionary value.

Frozensets are hashable, we can use the elements as a dictionary key or as an element from another set

* Set is represented by curly braces like this {} or you can use set(). fruits = {"Apple", "Banana", "Cherry"}

Frozensets are represented by the built-in function which is frozenset()

fruits = frozenset(["Cherry", "Apple", "Kiwi"])

* **OOPS in Python**
* *Class*: A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created.

Class bird:

print(“This is a bird”)

* *Object*: The object is an entity that has a state and behavior associated with it. It may be any real-world object like an integer, a string and dictionaries etc.

State: It is represented by the attributes of an object

Behavior: It is represented by the methods of an object. It reflects the

response of an object to the other objects.

* *\_\_init\_\_ method*: Before executing the code, python interpreter reads source file and define few global variables.If the python interpreter is running that module as the main program, it sets special \_\_name\_\_ variable to have a value '\_\_main\_\_'.If this file is being imported from another module,\_\_name\_\_ will be set to the module's name.Module's name is available as value to \_\_name\_\_ global variable.A module is a file containing python definitions and statements.The file name is the module name with suffix.py appended.

print("Always executed")

if \_\_name\_\_ == '\_\_main\_\_':

print("Executed when invoked directly")

else:

print("Executed when imported")

* *Python Inheritance*: Inheritance is the capability of one class to derive or inherit the properties from another class. The class that derives properties is called the derived class or child class and the class from which the properties are being derived is called the base class or parent class.

Single Inheritance: Single-level inheritance enables a derived class to

inherit properties from a single-parent class.

Multilevel Inheritance: Multilevel inheritance enables a derived

class to inherit properties from an immediate parent class which in turn

inherits properties from his parent class.

Hierarchical Inheritance: Hierarchical-level inheritance enables more

than one derived class to inherit properties from a parent class.

Multiple Inheritance: Multiple-level inheritance enables one derived class

to inherit properties from more than one parent class.

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def display(self):

print(self.name, self.age)

class Student:

pass

* *Polymorphism*: It is taken from the Greek words Poly(many) and morphism(forms).In programming, it refers to methods/functions/operators with the same name that can be executed on many objects or classes.

my\_str = “I am a student”

My\_tuple = (1,2,3,4,7)

print(len(my\_str))

print(len(my\_tuple))

Here len() function is used on different objects which gives different output.

* *Encapsulation:* It is one of the fundamental concepts in object-oriented programming.It describes the idea of wrapping data and the methods that work on data within one unit.This puts restrictions on accessing the variables and methods directly and can prevent the accidental modification of data.A class is an example of encapsulation as it encapsulate all the data that is member function, variables, etc.

class Employee():

def \_\_init\_\_(self, name, project):

self.name = name

self.project = project

def work(self):

print(self.name,”is working on the”,self.project)

* *Data Abstraction:* Data abstraction is one of the most essential concepts in python OOPs which is used to hide the irrelevant information from the user and show the information that are relevant to the users.In case of television remote, for example. The buttons tell us how to increase or decrease the volume and power the television on and off, and these commands are relevant to us.

from abc import ABC class

class Car(ABC):

def mileage(self):

pass

class Tesla(Car):

def mileage(self):

print(“The mileage is 10 kmph”)

class Suzuki(Car):

def mileage(self):

print(“The mileage is 25 kmph”)

Here data abstraction is implemented using abstract classes and methods.

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