OOPs:

=====

-> Object Oriented Programming System

-> High-level programming languages

classified into three types:

1) Functional Programming Languages : C, Python

2) Object Based Programming Languages : VB Script

3) Object Oriented Programming Languages : C++, Python, Java, C# etc.

-> When programming languages are with functions (built-in and/or user-defined) those are called as "Functional programming languages".

-> OOPs concepts and Principles:

1) Class

2) Object

3) Method

4) Constructor

5) Destructor

6) Garbage Collector

Principles:

1) Encapsulation

2) Polymorphism

3) Inheritance

4) Abstraction

What is Class?

---------------

-> Class is a logical entity (which is not real)

-> class is one of the non-primitive datatype

class is a collection of attributes (variables/data) and/or methods (behaviors)

-> keyword: "class"

Syntax:

class Class-name:

attributes/variables/data

methods

Note:

-----

class name should be "capitalize case".

Ex: Land

Home ==> 3BHK

Plan/Design : Design Engineers

Plans on Sheet (Logic)

(Not-real object) ==> class

Home construction

on land ==> Real entity ==> Object

Example for class and Object:

----------------------------------

Person wants to join in one organization

Organization can create candidature in organization portal

Proforma:

user-name : Rajesh

age : 32

gender : Male

previous exp : 5 yrs

previous designation : Sr. Programmer

current designation : Tech Lead

current salary : 20 LPA

working domain : Banking etc.

Class is a collection/block of data and/or behaviors (method)

-> Logical entity

does not required any memory

-> to create a class,

we need to use "class" keyword.

Syntax:

class Class-Name:

data

methods

-> Class is a template

-> Class is a blueprint

When we want to access any data or method from the class, we must be required an "object".

-> Object is the reference to the class.

-> Object is a physical entity.

Requires a memory.

Syntax for the object creation:

--------------------------------

object-name = Class-name()

Object name must be in camel-case representation.

Accessing of members of the class using an object:

--------------------------------------------------

Object-name.membername

Note:

------

To access the members of the members of the class, we can use "." operator.

Class with data:

----------------

-> data ==> attribute/variable

# class with only data

class MyClass:

a = 100 # members

b = 'x'

c = 1.23

myObject = MyClass()

print("The data of a class = ")

print(myObject.a)

print(myObject.b)

print(myObject.c)

class with methods:

-------------------

Method Vs Functions:

--------------------

method always allowed to define within the class only.

Whereas the function always allowed to define in outside the class only.

method in the class can accept "self" keyword as parameter while the definition.

But the function is not.

class Student:

def profile(self):

name = "Krishna"

roll = 12

division = 10

print("Name = ",name)

print("Roll = ",roll)

print("Division = ",division)

student = Student()

student.profile()

Ways to create methods:

-----------------------

1) Methods without parameters

2) Methods with parameters

---------------------------

class Student:

def profile(self,name,roll):

print("Name of the student = ",name)

print("Roll of the student = ",roll)

stu = Student()

stu.profile("Karthik",123) # method call

stdentName = "Rahul"

studentRoll = 102

stu.profile(stdentName,studentRoll)

stu.profile(roll=studentRoll,name=stdentName)

3) methods without return type

4) methods with return type

----------------------------

class Calculator:

def addition(self):

a = 1021

b = 2031

return (a+b)

calci = Calculator()

s = calci.addition()

print("The sum = ",s)

print(calci.addition())