**Python Programming**

**Practical Questions - Database Management**

# Session 2 - Part 2

**CLASSES, OBJECTS and METHODS (Functions)**

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

# Create a Class

To create a class, use the keyword

class

:

class

MyClass

:

x

=

5

# Create Object

Now we can use the class named MyClass (in example above) to create objects:

#Create an object named p1, and print the value of x:

#To comment, use the hashtag symbol (#)

p1

=

MyClass

()

print

(

p1

.

x

)

An object consists of :

**State:** It is represented by the attributes of an object. It also reflects the properties of an object.

**Behaviour:** It is represented by the methods of an object. It also reflects the response of an object to other objects.

**Identity:** It gives a unique name to an object and enables one object to interact with other objects.

## Declaring Objects (Also called instantiating a class)

When an object of a class is created, the class is said to be instantiated. All the instances share the attributes and the behavior of the class. But the values of those attributes, i.e. the state are unique for each object. A single class may have any number of instances.

# Python3 program to

# demonstrate instantiating

# a class

class

Dog

:

# A simple class

# attribute

attr1

=

"mammal"

attr2

=

"dog"

# A sample method

def

fun

(

self

):

print

(

"I'm a"

,

self

.

attr1

)

print

(

"I'm a"

,

self

.

attr2

)

# Driver code

# Object instantiation

Rodger

=

Dog

()

# Accessing class attributes

# and method through objects

print

(

Rodger

.

attr1

)

Rodger

.

fun

()

# init method

The **init** method is similar to constructors in C++ and Java. Constructors are used to initialize the object’s state. Like methods, a constructor also contains a collection of statements(i.e. instructions) that are executed at the time of Object creation. It runs as soon as an object of a class is instantiated. The method is useful to do any initialization you want to do with your object.

# Sample class with init method

class

Person

:

# init method or constructor

def

\_\_init\_\_

(

self

,

name

):

self

.

name

=

name

# Sample Method

def

say\_hi

(

self

):

print

(

'Hello, my name is'

,

self

.

name

)

p

=

Person

(

'Nikhil'

)

p

.

say\_hi

()

Output

Hello

,

my

name

is

Nikhil

# Class and Instance Variables

Instance variables are for data, unique to each instance and class variables are for attributes and methods shared by all instances of the class. Instance variables are variables whose value is assigned inside a constructor or method with self whereas class variables are variables whose value is assigned in the class.

**Defining instance variables using a constructor.**

# Python3 program to show that the variables with a value

# assigned in the class declaration, are class variables and

# variables inside methods and constructors are instance

# variables.

# Class for Dog

class

Dog

:

# Class Variable

animal

=

'dog'

# The init method or constructor

def

\_\_init\_\_

(

self

,

breed

,

color

):

# Instance Variable

self

.

breed

=

breed

self

.

color

=

color

# Objects of Dog class

Rodger

=

Dog

(

"Pug"

,

"brown"

)

Buzo

=

Dog

(

"Bulldog"

,

"black"

)

print

(

'Rodger details:'

)

print

(

'Rodger is a'

,

Rodger

.

animal

)

print

(

'Breed: '

,

Rodger

.

breed

)

print

(

'Color: '

,

Rodger

.

color

)

print

(

'\nBuzo details:'

)

print

(

'Buzo is a'

,

Buzo

.

animal

)

print

(

'Breed: '

,

Buzo

.

breed

)

print

(

'Color: '

,

Buzo

.

color

)

# Class variables can be accessed using class

# name also

print

(

"\nAccessing class variable using class name"

)

print

(

Dog

.

animal

)

**Output**

#OUTPUT

Rodger

details

:

Rodger

is

a

dog

Breed

:

Pug

Color

:

brown

Buzo

details

:

Buzo

is

a

dog

Breed

:

Bulldog

Color

:

black

Accessing

class

variable

using

class

name

dog

**Defining instance variables using the normal method.**

# Python3 program to show that we can create

# instance variables inside methods

# Class for Dog

class

Dog

:

# Class Variable

animal

=

'dog'

# The init method or constructor

def

\_\_init\_\_

(

self

,

breed

):

# Instance Variable

self

.

breed

=

breed

# Adds an instance variable

def

setColor

(

self

,

color

):

self

.

color

=

color

# Retrieves instance variable

def

getColor

(

self

):

return

self

.

color

# Driver Code

Rodger

=

Dog

(

"pug"

)

Rodger

.

setColor

(

"brown"

)

print

(

Rodger

.

getColor

())

**Output**

brown

# The init() Function

The examples above are classes and objects in their simplest form, and are not really useful in real life applications.

To understand the meaning of classes we have to understand the built-in **init**() function.

All classes have a function called **init**(), which is always executed when the class is being initiated.

Use the **init**() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

**Note:**

The

\_\_init\_\_()

function is called automatically every time the class is being used to

create a new object.

#Create a class named Person, use the \_\_init\_\_() function to assign values for name

and age:

class

Person

:

def

\_\_init\_\_

(

self

,

name

,

age

):

self

.

name

=

name

self

.

age

=

age

p1

=

Person

(

"John"

,

36

)

print

(

p1

.

name

)

print

(

p1

.

age

)

# The str() Function

The **str**() function controls what should be returned when the class object is represented as a string.

If the **str**() function is not set, the string representation of the object is returned:

#The string representation of an object WITHOUT the \_\_str\_\_() function:

class

Person

:

def

\_\_init\_\_

(

self

,

name

,

age

):

self

.

name

=

name

self

.

age

=

age

p1

=

Person

(

"John"

,

36

)

print

(

p1

)

# Object Methods

Objects can also contain methods. Methods in objects are functions that belong to the object.

Let us create a method in the Person class:

**Note:**

The

self

parameter is a reference to the current instance of the class, and is used to

access variables that belong to the class.

Insert

a

function

that

prints

a

greeting

,

and

execute

it

on

the

p1

object

:

class

Person

:

def

\_\_init\_\_

(

self

,

name

,

age

):

self

.

name

=

name

self

.

age

=

age

def

myfunc

(

self

):

print

(

"Hello my name is "

+

self

.

name

)

p1

=

Person

(

"John"

,

36

)

p1

.

myfunc

()

# The self Parameter

The self parameter is a reference to the current instance of the class, and is used to access variables that belongs to the class.

It does not have to be named self , you can call it whatever you like, but it has to be the first parameter of any function in the class:

#Use the words mysillyobject and abc instead of self:

class

Person

:

def

\_\_init\_\_

(

mysillyobject

,

name

,

age

):

mysillyobject

.

name

=

name

mysillyobject

.

age

=

age

def

myfunc

(

abc

):

print

(

"Hello my name is "

+

abc

.

name

)

p1

=

Person

(

"John"

,

36

)

p1

.

myfunc

()

# Modify Object Properties

You can modify properties on objects like this:

#Set the age of p1 to 40:

p1

.

age

=

40

# Delete Object Properties

You can delete properties on objects by using the

del

keyword:

W

#Delete the age property from the p1 object:

del

p1

.

age

# Delete Objects

You can delete objects by using the

del

keyword:

del

p1

# The pass Statement

class definitions cannot be empty, but if you for some reason have a class definition with no

content, put in the pass statement to avoid getting an error.

class

Person

:

pass

**Activity**

Q1. Create a class called Cat

Create a class variable named Cat

Create a constructor (init method)

Create instance variables for breed and color

Create objects (Rasco , Snobs) \* Remember, objects have names and properties

Print output