Experiment 8

Aim-Write python programs to implement Classes, object, Static method, constructors and Inner class

Theory:

1. Classes and Objects:

● A class is a blueprint for creating objects. Objects are instances of classes.

● Classes encapsulate data (attributes) and behaviors (methods) related to a particular concept.

Syntax:

class *MyClass*:

def \_\_init\_\_(self, *attribute1, attribute2*):

self.*attribute1* = *attribute1*

self.*attribute2* = *attribute2*

def *my\_method*(self):

# *method code here*

2. Constructors:

● A constructor (\_\_init\_\_ method) initializes the attributes of an object when it is created.

● It is called automatically when an object is instantiated.

Syntax:

class *MyClass*:

def \_\_init\_\_(self, *attribute1, attribute2*):

self.*attribute1* = *attribute1*

self.*attribute2* = *attribute2*

3. Static Method:

● A static method belongs to the class rather than an instance of the class.

● It is defined using the @staticmethod decorator.

● Static methods do not have access to instance-specific data.

Syntax:

class *MyClass*:

@staticmethod

def *my\_static\_method*():

# *static method code here*

4. Inner Class:

● An inner class is a class defined within another class.

● It is used to logically group classes that are only used in one place.

Syntax:

class *OuterClass*:

def \_\_init\_\_(self):

*# outer class constructor*

class InnerClass:

def \_\_init\_\_(self):

# *inner class constructor*

Program:

class Subject:

no\_of\_subjects = 0

def \_\_init\_\_(self, name: str, id: str, subject\_incharge: str, credits: int):

self.name = name

self.id = id

self.subject\_incharge = subject\_incharge

self.credits = credits

self.resources = []

self.prerequisite = set()

self.info = self.Info()

Subject.no\_of\_subjects += 1

class Info:

description = ""

def set\_description(self, description):

self.description = description

def display(self):

print("Description:", self.description)

@staticmethod

def get\_subject\_type():

return "Regular"

def display\_info(self):

print("Subject Name:", self.name)

print("Subject Code:", self.id)

self.info.display()

def add\_resource(self, resource):

self.resources.append(resource)

def remove\_resource(self, resource):

self.resources.remove(resource)

def get\_resources(self):

return self.resources

def add\_prerequisite(self, prerequisite):

self.prerequisite.add(prerequisite)

def remove\_prerequisite(self, prerequisite):

self.prerequisite.remove(prerequisite)

physics = Subject("Physics", "FEC102", "Neeta", 5)

physics.info.set\_description("This is where you learn laws of Nature")

maths = Subject("Maths", "ITC301", "Mariya", 5)

maths.info.set\_description("This is where you learn language of Nature")

physics.add\_resource("https://physicswallah.com")

physics.add\_resource("https://codehelp.com")

physics.display\_info()

print(physics.get\_resources())

maths.add\_resource("www.youtube.com/wth.disneshsirlivestudy=ICHIGSND")

maths.add\_resource("https://rkacademy.in")

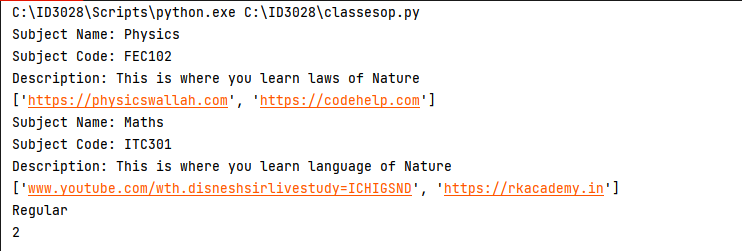
maths.display\_info()

print(maths.get\_resources())

print(Subject.get\_subject\_type())

print(Subject.no\_of\_subjects)

Output:



Conclusion: Classes and constructors and its method are executed successfully in Python