Practical No 10

Aim: To implement classes and objects in Python, including instance methods, constructors, and destructors.

Theory:

In Python, classes, objects, constructors, and destructors are fundamental concepts of Object-Oriented Programming (OOP).

Class

A class is a blueprint for creating objects.
It defines the properties (variables) and behaviors (functions or methods) that the objects will have.

Syntax:
```
class ClassName:

    # class definition
```

Objects

- An object is called an instance of a class.
- Suppose Bike is a class then we can create objects like bike1, bike2, etc from the class.

Syntax:
```
objectName = ClassName()
```

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.

Example
```
class Person:
  pass
```

Constructor:

- The _ _init_ _ method is similar to **constructors** in c++ and Java.
- Constructors are used to initialize the object's state.
- The task of constructors is to initialize(assign values) to the data members of the class when an object of class is created.

Synatx:

```
class ClassName:
  def _init_( self , variables...):
    ##body
```

Destructor:

- The __del___method is similar to destructor in c++ and Java.
- Destructors are used to destroying the object's state.

Syntax:

```
class ClassName:
  def _del_( self ,):
  ##body
```

- The destructor was called **after the program ended.**
- It can be called Automatically as well as manually.
- Object will be deleted at the end of the program.

Example:

```
class Student:
    # Constructor: runs automatically when an object is created
    def __init__(self, name, roll_no):
        self.name = name
        self.roll_no = roll_no
        print(f"Constructor called: Student {self.name} (Roll No: {self.roll_no}) created.")

    # Instance method
    def show(self):
        print(f"Name: {self.name}, Roll No: {self.roll_no}")

    # Destructor: runs automatically when an object is deleted
    def __del__(self):
        print(f"Destructor called: Student {self.name} (Roll No: {self.roll_no}) deleted.")

# ---- Creating Object ----
s1 = Student("Prachi", 101)
s1.show()

# ---- Deleting Object ----
del s1
```

Python does not require to pre-declare variables inside a class like C++ or Java.
Variables can directly create and assign inside the **constructor (`__init__`)** or anywhere in the class.

Sometimes, variables that are **shared by all objects** these are called **class variables**.

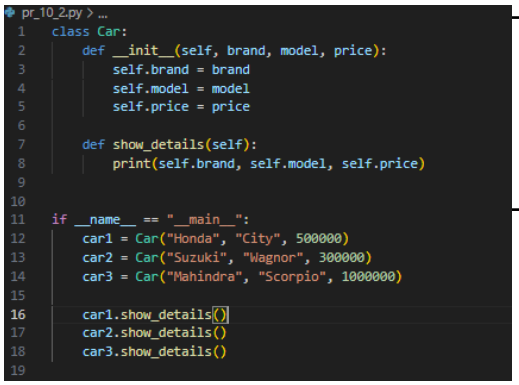
```
class Student:
    college_name = "YCCE"   # Class variable (common to all objects)

    def __init__(self, name, roll_no):
        self.name = name    # Instance variable (unique to each object)
        self.roll_no = roll_no

s1 = Student("Akshay", 101)
s2 = Student("Priya", 102)

print(s1.college_name)
print(s2.college_name)
```

Program:

1. Design a class Laptop with the attributes brand, RAM, and processor. Initialize these attributes using a constructor, and display the laptop details using an instance method. Include a destructor that prints a message such as "Laptop configuration deleted."

```
pr10_1.py > ...
class Laptop:

    def __init__(self, brand, RAM, processor):
        self.brand = brand
        self.RAM = RAM
        self.processor = processor

    def showDetails(self):
        print(f"Brand: {self.brand}, RAM: {self.RAM} and processor: {self.processor}\n")

    def __del__(self):
        print("Laptop configuration deleted")


if __name__ == "__main__":
    laptop1 = Laptop("lenovo", "8gb", "i5")
    laptop1.showDetails()
```

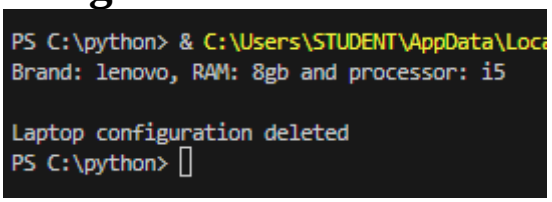
2. Create a class `Car` with data members `brand`, `model`, and `price`. Use a **constructor** to initialize them and a method `show_details()` to print car information. Demonstrate how multiple **objects** can be created from the same class.



```
pr_10_2.py > ...
class Car:
    def __init__(self, brand, model, price):
        self.brand = brand
        self.model = model
        self.price = price

    def show_details(self):
        print(self.brand, self.model, self.price)


if __name__ == "__main__":
    car1 = Car("Honda", "City", 500000)
    car2 = Car("Suzuki", "Wagnor", 300000)
    car3 = Car("Mahindra", "Scorpio", 1000000)

    car1.show_details()
    car2.show_details()
    car3.show_details()
```

Output:

Program 1-



Program 2-