Task1:

```python
class Kitten:
    pass
#Kitten object is stored in computer memoru
#But the address printed o the creen will be different
#kitt is an instance/object of the class
kitt = Kitten()
print(kitt)
```

Output:

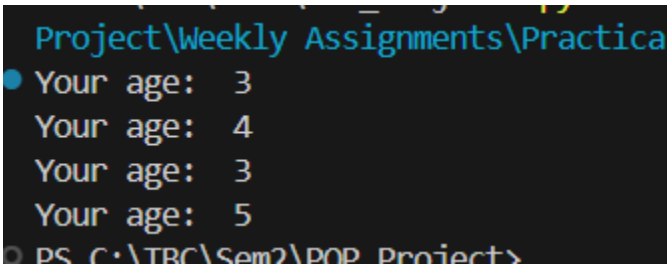
```
PS C:\TBC\Sem2\POP_Project> python -u "c:\TBC\Sem2\POP_
Project\Weekly Assignments\Practical7\kitten.py"
<__main__.Kitten object at 0x000001F701AE6A50>
PS C:\TBC\Sem2\POP_Project>
```

Kitten2 py:

```
class Kitten:
    #Constructor
    def __init__(self,value):
        #Self parameter is a regerence to the current instance/object
        # and used to access variables that belongs to the class
        #Inititalizing instance/member variable age
        #self.age =1

        self.age = value

    #instance/member method
    def set_age(self,value):
        self.age = value

    #an instance/member method
    def display_age(self):
        #print(self.age)
        #print("Age Unknown")
        print("Your age: ",self.age)

#This invokes parameterised constructor
kitt=Kitten(3)
kitt2 = Kitten(4)
#kitt3= Kitten()

#calling the instance/member method using the object kitt
kitt.display_age()
kitt2.display_age()
kitt.display_age()

#kitt3.display_age()
"""Task 1.8:
    No, the program doesnot work. We have created a parameterised constructor
    which ask value(argument) while calling the class. So, if we call the
instance/object without
    value the program doesnot work and throws error.
    Therefore, to prevent it everytime we call the Class's object we must pass
needed values.
"""

#Setting kitt 1 age 5
kitt.set_age(5)
kitt.display_age()
```

```
"""Task 1.9:
    While only initialising the member variables using a parameterised
constructor,
    we should alwaus pass argument. but by calling set_age method, if we want to
later change (modify)
    age of the kitten instead of going through the kitt class constructor we can
change it with
    the method which is more dynamic and functional as we dont need to
reconstruct the object to change age.
"""
```

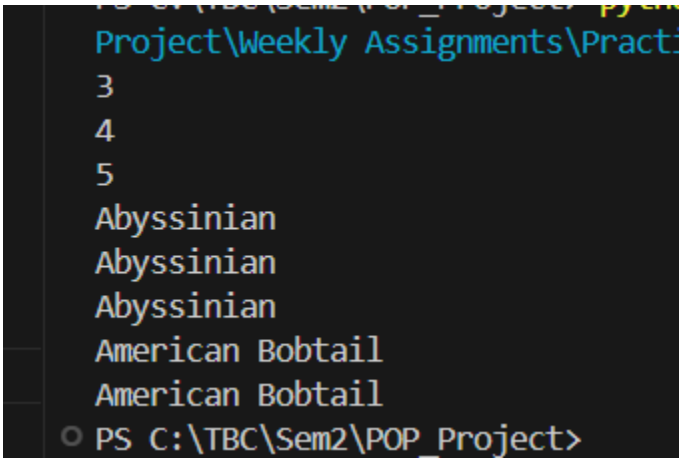
Output:



Kitten 3 py:

```
class Kitten:
    breed = "Abyssinian"

    #parameterised constructor
    def __init__(self,value=None):
        #initialising instance/member variable age
        self.age = value

    #an instance/member method
    def set_age(self,value):
        self.age = value

    def display_age(self):
        print(self.age)

kitt = Kitten(3)
kitt2 = Kitten(4)

#kitt3 = Kitten() works if during init value = None

kitt.display_age()
kitt2.display_age()

kitt.set_age(5)
kitt.display_age()

print(kitt.breed)
print(kitt2.breed)
#Breed is accessed using the class (all over the class)
print(Kitten.breed)
#instance variavle cannot be accessed via class (as its local variable for that
instance)
#print(Kitten.age)

#Chaning the breed value
Kitten.breed = "American Bobtail"
print(kitt.breed)
print(Kitten.breed)
```

Output:


Project\Weekly Assignments\Practi
3
4
5
Abyssinian
Abyssinian
Abyssinian
American Bobtail
American Bobtail
PS C:\TBC\Sem2\POP_Project>

Exercise 3:

```python
class Rectangle:

    def __init__(self,width=1.0, height=1.0):
        self.width = width
        self.height = height

    def getArea(self):
        area = self.width * self.height
        return area

    def getPerimeter(self):
        perimeter = 2*(self.width+self.height)
        return perimeter


obj1= Rectangle(width=4,height=40)
obj2= Rectangle(3.5,35.9)
area2 = obj2.getArea()
perimeter2 = obj2.getPerimeter()
print("Object 1: ")
print(f"Width of obj1:{obj1.width}\nHeight of obj1: {obj1.height}")
print(f"Area of Obj1: {obj1.getArea():.2f}")
print(f"Perimeter of Obj1: {obj1.getPerimeter():.2f}\n")

print("Object 2: ")
print(f"Width of obj1:{obj2.width}\nHeight of obj2: {obj2.height}")
print(f"Area of Obj2: {area2:.2f}")
print(f"Perimeter of Obj2: {perimeter2:.2f}\n")
```

Output:

```
"
Object 1:
Width of obj1:4
Height of obj1: 40
Area of Obj1: 160.00
Perimeter of Obj1: 88.00

Object 2:
Width of obj1:3.5
Height of obj2: 35.9
Area of Obj2: 125.65
Perimeter of Obj2: 78.80

PS C:\TBC\Sem2\POP_Project>
```

Exercise 4:

```
class Car:
    #attribute of car
    brand = "ToyCar"
    color = "White"

    #Car's constructor
    def __init__(self,door=5,price=20000):
        self.door=door
        self.price=price


    #Car's instance/ member method
    def startCar(self):
        print("Car has stated")

    def stopCar(self):
        print("Car has stopped")

    def setNumberofDoors(self,door):
        self.door = door

    def getNumberOfDoors(self):
        print(f"Number of doors: {self.door}")

    def setPrice(self,price):
        self.price=price

    def getPrice(self):
        print(f"Car's Price: {self.price}")

car1 = Car()
print("Car 1:")
#printing car1 details
car1.getNumberOfDoors()
car1.getPrice()
print("Car Color:",car1.color)
print(f"Car Brand: {car1.brand}")

print("\nCar 2:")
car2= Car()
#Changing cars door num price,color,brand
car2.setNumberofDoors(10)
car2.setPrice(50000)
```

```
car2.brand = "CarToy"
car2.color = "Maroon"
#printing car details
car2.getNumberOfDoors()
car2.getPrice()
print("Car Color:",car2.color)
print(f"Car Brand: {car2.brand}")
```

Output:

```
PS C:\TBC\Sem2\POP_Project> python -u "c:\TBC\
Project\Weekly Assignments\Practical7\carClass
Car 1:
Number of doors: 5
Car's Price: 20000
Car Color: White
Car Brand: ToyCar

Car 2:
Number of doors: 10
Car's Price: 50000
Car Color: Maroon
Car Brand: CarToy
PS C:\TBC\Sem2\POP_Project>
```