Name: Muhammad Tamjeed Hussain

Roll No: 333899

1. Write a Python program to create a class called "Person" with properties for

name, age and country. Include a method to display the person's details. Create two instances of the 'Person' class and display their details.

```
class Person:
    def __init__(self,name,age,nation):
        self.name=name
        self.age=age
        self.country=nation
        print("\nProfile Created")
    def Describe(self):
        print(f"\nName : {self.name}\nAge : {self.age}\nNationality :
{self.country}")

p1=Person("Seraph" ,26, "Pakistani")
p2=Person("Taha" ,21, "Pakistani")


Profile Created

Profile Created

p1. Describe()
p2. Describe()


Name : Seraph
Age : 26
Nationality : Pakistani

Name : Taha
Age : 21
Nationality : Pakistani
```

2. Write a Python program to create a class called 'Rectangle' with properties for

```
  width and height. Include two methods to calculate rectangle area
and perimeter.
  Create an instance of the 'Rectangle' class and calculate its area
and perimeter.

class Rectangle:
    def __init__(self,width,length):
        self.width=width
        self.length=length

    def Area(self):
        Area=self.width*self.length
        print(f"The Area of Rectangle is {Area}.")
    def Perimeter(self):
        Perimeter=(self.length*self.width)*2
        print(f"The Perimeter of Rectangle is {Perimeter}")

InstanceRectangle=Rectangle(45,20)

InstanceRectangle.Area()
InstanceRectangle.Perimeter()

The Area of Rectangle is 900.
The Perimeter of Rectangle is 1800
```

3. Write a Python program that creates a class called 'Vehicle' with properties for

```
make, model, and year. Include a method to display vehicle details.
Create a
subclass called 'Car' that inherits from the 'Vehicle' class and
includes an
additional property for the number of doors. Override the display
method to
include the number of doors.

class Vehicle:
    def __init__(self,make,model,year):
        self.make=make
        self.model=model
        self.year=year
    def Details(self):
        print(f"\nMake : {self.make}\nModel : {self.model}\nYear :
{self.year}")
```

```
class Car(Vehicle):
    def __init__(self,make,model,year,doors):
        super() .__init__(make,model,year)
        self.doors=doors
    def Details(self):
        print(f"\nMake : {self.make}\nModel : {self.model}\nYear :
{self.year}\nDoors : {self.doors}")

v1=Vehicle("Tesla","Model 2", 2025)
c1=Car("Honda", "Civic",2025,4)

v1. Details()
c1. Details()
```

```
Make : Tesla
Model : Model 2
Year : 2025

Make : Honda
Model : Civic
Year : 2025
Doors : 4
```

4. Write a Python program that creates a class called "BankAccount" with

```
properties for account number and balance. Include methods to deposit
and
withdraw money from the account. Create some instances of the
"BankAccount"
class, deposit some money, and withdraw a portion of it.

class BalanceException(Exception):
    pass

class BankAccount:
    def __init__(self,accname,amm):
        self.name=accname
        self.balance=amm
        print("\nAccount Created")
    def getBalance(self):
        print(f"Account '{self.name}' Balance = ${self.balance}")
    def Deposit(self,amount):
        self.balance=self.balance+amount
        print("Deposit Successful")
        self.getBalance()
```

```
    def viableTransaction(self,amount):
        if self.balance>=amount:
            return
        else:
            raise BalanceException(f"Sorry Account '{self.name}' has
balance of {self.balance}")
    def Withdraw(self,amount):
        try:
            self.viableTransaction(amount)
            self.balance=self.balance-amount
            print("Withdraw Successful")
            self.getBalance()
        except BalanceException as Error:
            print(f"Withdrawl Interupted\n{Error}")

b1=BankAccount("Raqib",500)


Account Created

b1.getBalance()

Account 'Raqib' Balance = $500

b1.Deposit(300)

Deposit Successful
Account 'Raqib' Balance = $800

b1.Withdraw(100)

Withdraw Successful
Account 'Raqib' Balance = $700

b1.Withdraw(1000)

Withdrawl Interupted
Sorry Account 'Raqib' has balance of 700
```

5. Write a Python program that creates a class called 'Shape' with a method to

```
calculate the area. Create two subclasses, 'Circle' and 'Triangle',
that inherit from
the 'Shape' class and override the area calculation method. Create an
instance of
the 'Circle' class and calculate its area. Similarly, do the same for
the 'Triangle'
class.
```

```python
import math

# Base class
class Shape:
    def area(self):
        raise NotImplementedError("This method should be overridden by
subclasses")

# Subclass for Circle
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

# Subclass for Triangle
class Triangle(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height

    def area(self):
        return 0.5 * self.base * self.height

# Create an instance of Circle
circle = Circle(5)  # Radius = 5
print(f"Area of the Circle: {circle.area():.2f}")

# Create an instance of Triangle
triangle = Triangle(10, 7)  # Base = 10, Height = 7
print(f"Area of the Triangle: {triangle.area():.2f}")
```

```
Area of the Circle: 78.54
Area of the Triangle: 35.00
```

6. Write a Python program that creates a class called 'Employee' with properties

```
for name and salary. Include a method to calculate annual salary.
Create a
subclass called 'Manager' that inherits from the 'Employee' class and
adds an
additional property for department. Override the annual salary
calculation method
to include bonuses for managers. Create two instances of the 'Manager'
```

```
class
  and calculate their annual salary.

class Employee:
    def __init__(self,name,salary):
        self.name=name
        self.salary=salary

    def AnnualSalary(self):
        print(f"\nEmployee:\nName : {self.name}\nSalary :
{self.salary*12}")

class Manager(Employee):
    def __init__(self,name,salary,dept):
        super() .__init__(name,salary)
        self.dept=dept
    def AnnualSalary(self):
        print(f"\nManager:\nName : {self.name}\nSalary :
{(self.salary*12)}\nWith Bonus : {(self.salary*12)+50000}\
nDepartment : {self.dept}")
        print("Manager Gets 50000 Bonus Annually")

e1=Employee("Taha" ,200000)
m1=Manager("Seraph" ,320000,"Supervisior")

e1.AnnualSalary()
m1.AnnualSalary()


Employee:
Name : Taha
Salary : 2400000

Manager:
Name : Seraph
Salary : 3840000
With Bonus : 3890000
Department : Supervisior
Manager Gets 50000 Bonus Annually
```

7. Write a Python program that creates a class `Book` with properties for title,

```
 author, and publication year. Include a method to display book
details. Create a
 subclass called 'Ebook' that inherits from the 'Book' class and
includes an
 additional property for book price. Override the display method to
```

```
include the
 book price. Create an instance of the 'Ebook' class and display its
details

class Book:
    def __init__(self,title,author,publicationyear):
        self.name=title
        self.writer=author
        self.year=publicationyear
    def Details(self):
        print(f"\nBook\nTitle : {self.name}\nAuthor : {self.writer}\
nYear Of Publication : {self.year}")

class Ebook(Book):
    def __init__(self,title,author,publicationyear,price):
        super() .__init__(title,author,publicationyear)
        self.price=price
    def Details(self):
        print(f"\nEbook\nTitle : {self.name}\nAuthor : {self.writer}\
nYear Of Publication : {self.year}\nPrice : {self.price}")

b1=Book("The Communist Maefesto","Karl Marx",1848)
eb1=Ebook("Mein Kampl","Adolf Hitler",1925,"$200")

b1 .Details()
eb1 .Details()


Book
Title : The Communist Maefesto
Author : Karl Marx
Year Of Publication : 1848

Ebook
Title : Mein Kampl
Author : Adolf Hitler
Year Of Publication : 1925
Price : $200
```

8. Write a Python program that creates a class called 'Animal' with properties for

```
species and sound. Include a method to make the animal's sound. Create
a
subclass called 'Dog' that inherits from the 'Animal' class and adds
an additional
property for color. Override the make sound method to include the
```

```
dog's color.
Create an instance of the 'Dog' class and make it make its sound.

class Animal:
    def __init__(self,species,sound):
        self.name=species
        self.sound=sound
    def makeSound(self):
        print(f"\nMake Sound...\n{self.sound}")

class Dog(Animal):
    def __init__(self,species,sound,color):
        super().__init__(species,sound)
        self.color=color
    def makeSound(self):
        print(f"\nMake Sound....\n{self.sound}")
        print(f"What is your color....\n{self.color}")

a1=Animal("Tiger","Roar")
d1=Dog("German Shepherd", "Bark","Black Brown")

a1.makeSound()
d1.makeSound()


Make Sound...
Roar

Make Sound....
Bark
What is your color....
Black Brown
```

9. Write a Python program that creates a class called Bank with properties for

```
bank names and branches. Include methods to add a branch, remove a
branch,
and display all branches. Create an instance of the Bank class and
perform
 operations to add and remove branches

class Bank:
    def __init__(self,name,branch):
        self.name=name
        self.branch=[branch]
        print("\nBank Registered")
    def Details(self):
```

```
        print(f"\nBank Name : {self.name}\nBranches : {self.branch}")
    def addBranch(self,branch):
        self.branch.append(branch)
        print("New Branch Have Been Contructed")
        self.Details()
    def remBranch(self,branch):
        try:
            self.branch.remove(branch)
            print("Branch Destroyed")
            self.Details()
        except:
            print("Branch Does Not Exist")

bank1=Bank("Habib Bank","Korangi")


Bank Registered

bank1.Details()


Bank Name : Habib Bank
Branches : ['Korangi']

bank1 .addBranch("Defence")

New Branch Have Been Contructed

Bank Name : Habib Bank
Branches : ['Korangi', 'Defence']

bank1.remBranch("Korangi")

Branch Destroyed

Bank Name : Habib Bank
Branches : ['Defence']

bank1.remBranch("Defence")

Branch Destroyed

Bank Name : Habib Bank
Branches : []
```

10. Write a Python program that creates a class called Product with properties for

```
product ID, name, and price. Include a method to calculate the total price by
multiplying the price by the quantity. Create a subclass called
PersonalCareProduct that inherits from the Product class and adds an additional
property for the warranty period. Override the total price calculation method to
include the warranty period. Create an instance of the PersonalCareProduct
class and calculate its total price.
```

```
class Product:
    def __init__(self,productid,name,price):
        self.name=name
        self.ID=productid
        self.price=price
    def TotalPrice(self,quantity):
        Total=self.price*quantity
        print(f"\nProduct Name : {self.name}\nProduct Price : Rs.
{self.price}\nQuantity : {quantity}\nTotal : Rs.{Total}")

class PersonalCareProduct(Product):
    def TotalPrice(self,quantity):
        print("Is Your Product Within Warranty. Press Y Or N")
        inp=input() . capitalize()
        if inp=="Y":
            print(f"\nProduct Name : {self.name}\nProduct Price : Rs.
{self.price}\nQuantity : {quantity}\nTotal Within Warranty : Rs.
{Total}")
        else:
            Total=self.price*quantity
            print(f"\nProduct Name : {self.name}\nProduct Price : Rs.
{self.price}\nQuantity : {quantity}\nTotal : Rs.{Total}")

Dishes=Product(2020,"Dishes",2)
Cream=PersonalCareProduct(2024,"Cream",250)

Dishes.TotalPrice(5)
Cream.TotalPrice(1)
```

```
Product Name : Dishes
Product Price : Rs.2
Quantity : 5
Total : Rs.10
Is Your Product Within Warranty. Press Y Or N
```

```
Product Name : Cream
Product Price : Rs.250
Quantity : 1
Total : Rs.250
```

11. Write a Python program that creates a class called BankAccount with

```
 properties for account number, account holder name, and balance.
Include
 methods to deposit, withdraw, and transfer money between accounts.
Create
 multiple instances of the BankAccount class and perform operations
such as
  depositing, withdrawing, and transferring money.

class Bank:
    def __init__(self,accountholder,accountbalance=0):
        self.name=accountholder
        self.balance=accountbalance
        self.transaction=[]
    def Balance(self):
        print(f"{self.name} Balance is {self.balance}")
    def Deposit(self,amount):
        self.balance=self.balance+amount
        self.Transactions(f"Deposit : {amount}")
        self.Balance()
        print("\nDeposit Complete")
    def Withdraw(self,amount):
        try:
            if self.balance>amount:
                self.balance=self.balance-amount
                self.Transactions(f"Withdraw : {amount}")
                self.Balance()
            else:
                print("\nBalance is Low")
        except:
            print("Withdraw Interupted")
    def Transactions(self,description):
        self.transaction.append(description)
    def Transferfund(self,amount,account):
        try:
            print("\nProcessing")
            self.Withdraw(amount)
```

```
            account.Deposit(amount)
            print("\nTransfer Completed")
        except:
            print("\nTransfer Interrupted")

Seraph=Bank("Seraph",4500)
Taha=Bank("Taha",5500)

Seraph.Deposit(1000)
```

```
Seraph Balance is 5500

Deposit Complete
```

```
Taha.Deposit(1500)
```

```
Taha Balance is 7000

Deposit Complete
```

```
Seraph.Withdraw(500)
```

```
Seraph Balance is 5000
```

```
Seraph.Transferfund(1000,Taha)


Processing
Seraph Balance is 4000
Taha Balance is 8000

Deposit Complete

Transfer Completed
```

12. Write a Python program that creates a class called University with properties

```
 for university name and departments. Include methods to add a
department,
 remove a department, and display all departments. Create an instance
of the
University class and add and remove departments.

class University:
    def __init__(self,name,departments):
        self.name=name
        self.dept=[departments]
        print("\nUniversity Registered by HEC")
```

```
    def Details(self):
        print(f"\nUniversity Name : {self.name}\nDepartments :
{self.dept}")
    def addBranch(self,branch):
        self.dept.append(branch)
        print("\nBranch Constructed")
        self.Details()

    def removeBranch(self,branch):
        try:
            self.dept.remove(branch)
            print("\nBranch Demolished.")
            self.Details()
        except:
            print("\nBranch Does Not Exist")

uni1=University("Iqra University","Malir")


University Registered by HEC

 uni1.addBranch("Lahore")


Branch Constructed

University Name : Iqra University
Departments : ['Malir', 'University Road', 'Lahore']

uni1.removeBranch("Lahore")


Branch Demolished.

University Name : Iqra University
Departments : ['Malir', 'University Road']

uni1.removeBranch("Lahore")


Branch Does Not Exist
```