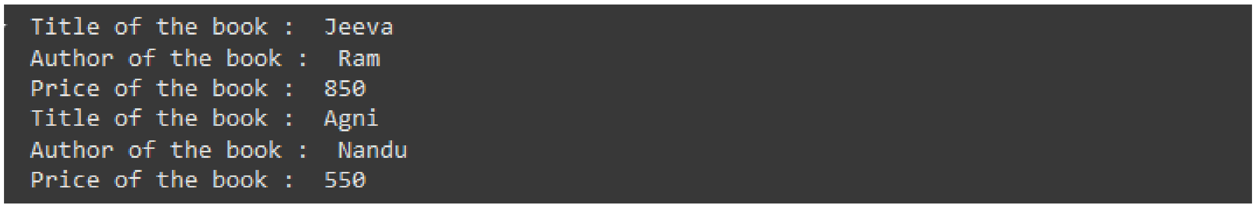
Class & Objects

1. Create a class Book with attributes for title, author, and price. Write a method to display the details of each book. Instantiate two objects of Book and print their details.

```
class Books():
    def __init__(self,title,author,price):
    self.title = title
    self.author = author
    self.price = price

    def display_books(self):
    print("Title of the book : ",self.title)
    print("Author of the book : ",self.author)
    print("Price of the book : ",self.price)

b = Books('Jeeva','Ram',850)
b1 = Books('Agni','Nandu',550)

b.display_books()
b1.display_books()
```

Output:



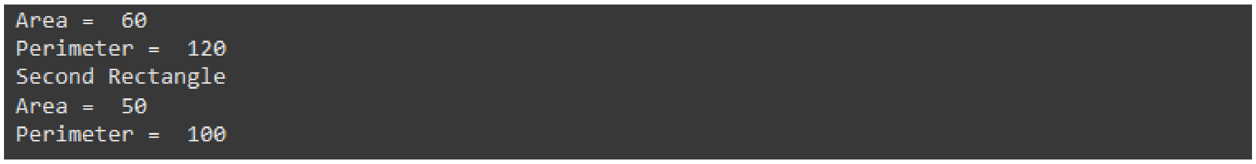
```
Title of the book :  Jeeva
Author of the book :  Ram
Price of the book :  850
Title of the book :  Agni
Author of the book :  Nandu
Price of the book :  550
```

Submitted by Nanda Krishnan V , Msc Cybersecurity

2. Write a class Rectangle with attributes length and width. Add methods to calculate and return the area and perimeter of the rectangle. Create two instances of Rectangle with different dimensions and display their area and perimeter.

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

        def area(self):
        area_rect = self.length * self.breadth
        print("Area = ",area_rect)

        def perimeter(self):
        perimeter_rect = 2 * ( self.length * self.breadth)
        print("Perimeter = ",perimeter_rect)

r1 = Rectangle(10,6)
r1.area()
r1.perimeter()

print("Second Rectangle ")
r2 = Rectangle(5,10)
r2.area()
r2.perimeter()
```

Output:

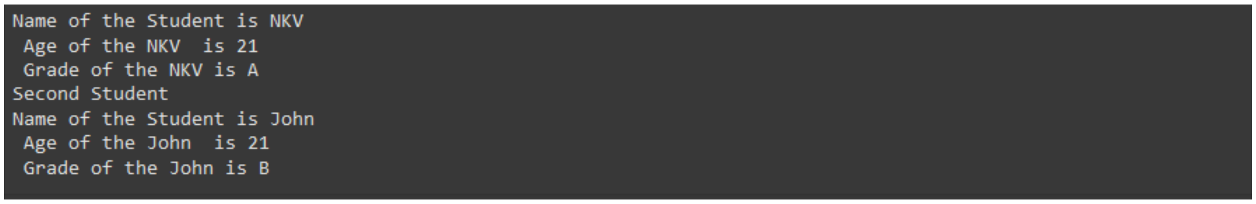


```
Area =  60
Perimeter =  120
Second Rectangle
Area =  50
Perimeter =  100
```

3. Define a class `Student` that includes attributes `name`, `age`, and `grade`. Use the `__init__` method to initialize these attributes. Write a method to display the student's information. Create multiple instances and test the display method.

```python
class Students():
        def __init__(self,name,age,grade):
        self.name = name
        self.age = age
        self.grade = grade

        def students_info(self):
        print(f"Name of the Student is {self.name}\n Age of the {self.name}  is {self.age}\n Grade of the {self.name} is {self.grade}")



s1 = Students("NKV",21,"A")
s1.students_info()

print("Second Student")

s2 = Students("John",21,"B")
s2.students_info()
```

Output:

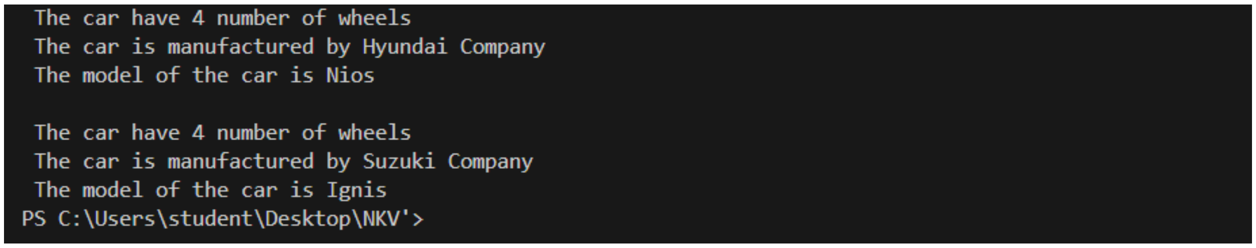


```
Name of the Student is NKV
 Age of the NKV  is 21
 Grade of the NKV is A
Second Student
Name of the Student is John
 Age of the John  is 21
 Grade of the John is B
```

4. Define a class Car with a class attribute wheels set to 4 and instance attributes make and model. Create multiple car instances and display the wheels attribute and specific make and model for each instance. Change the class attribute and observe the effect.

```
class Cars():
        wheels = 4
        def __init__(self,make,model):
        #self.wheels = 4
        self.make = make
        self.model = model

        def car_info(self):
        print(f" The car have {self.wheels} number of wheels\n The car is
manufactured by {self.make} Company \n The model of the car is {self.model}")

c1 = Cars("Hyundai","Nios")
c1.car_info()

print()

c2 = Cars("Suzuki","Ignis")
c2.car_info()
```

Output:

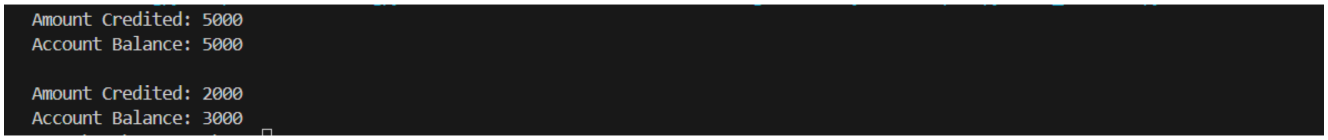


```
 The car have 4 number of wheels
 The car is manufactured by Hyundai Company
 The model of the car is Nios

 The car have 4 number of wheels
 The car is manufactured by Suzuki Company
 The model of the car is Ignis
PS C:\Users\student\Desktop\NKV'>
```

5. Implement a class BankAccount with attributes account_number and balance. Make balance a private attribute and create methods deposit() and withdraw() to modify it. Include a method to display the balance. Test the methods by creating an instance and performing a few transactions.

```
class BankAccount():
    def __init__(self,account_number):
        self.account_number = account_number
        self.__balance = 0

    def deposit(self,amount):
        self.__balance += amount
        print(f"Amount Credited: {amount}")
        print(f"Account Balance: {self.__balance}")

    def withdraw(self,amount):
        self.__balance -= amount
        print(f"Amount Credited: {amount}")
        print(f"Account Balance: {self.__balance}")

b = BankAccount(123)
b.deposit(5000)
print()
b.withdraw(2000)
```

Output:

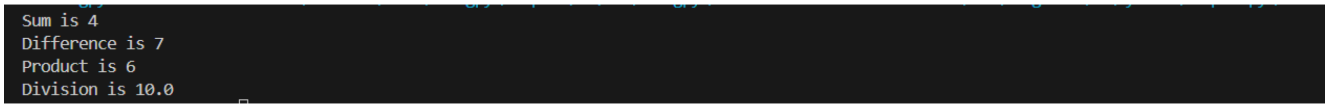


```
Amount Credited: 5000
Account Balance: 5000

Amount Credited: 2000
Account Balance: 3000
```

6. Define a class Calculator with static methods for basic operations: add, subtract, multiply, and divide. Call each method without creating an instance and verify the results.

```
class Calculator():

    @staticmethod
    def add(n1,n2):
        add = n1 + n2
        print(f"Sum is {add}")

    @staticmethod
    def sub(n1,n2):
        sub = n1 - n2
        print(f"Difference is {sub}")

    @staticmethod
    def mult(n1,n2):
        mult = n1 * n2
        print(f"Product is {mult}")

    @staticmethod
    def div(n1,n2):
        div = n1 / n2
        print(f"Division is {div}")

Calculator.add(1,3)
Calculator.sub(10,3)
Calculator.mult(2,3)
Calculator.div(30,3)
```

Output:

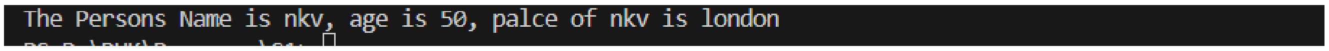


```
Sum is 4
Difference is 7
Product is 6
Division is 10.0
```

7. Create a class Person with attributes name, age, and city. Define the __str__ method to return a string representation of the object. Create an instance and print it to see the effect of __str__.

```python
class Person():
    def __init__(self,name,age,city):
        self.name = name
        self.age = age
        self.city = city

    def __str__(self):
        return f"The Persons Name is {self.name}, age is {self.age}, 
palce of {self.name} is {self.city}"

p = Person("nkv",50,"london")
print(p)
```

Output:

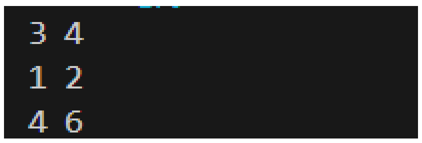


```
The Persons Name is nkv, age is 50, palce of nkv is london
```

8. Create a class Vector with x and y coordinates. Overload the + operator to add two vectors, and overload __str__ to display the vector in (x, y) format. Create two vectors and test the addition.

```python
class Vector():
    def __init__(self,x,y):
        self.x = x
        self.y = y

    def __add__(self,other):
        return Vector(self.x + other.x , self.y + other.y)
    def __str__(self):
        return f"{self.x} {self.y}"

v1 = Vector(3, 4)
v2 = Vector(1, 2)

addition = v1 + v2
print(v1)
print(v2)
print(addition)
```

Output:

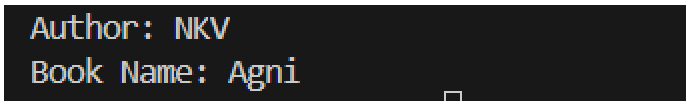


```
3 4
1 2
4 6
```

9. Create two classes Author and Book. The Book class should include an instance of the Author class as an attribute, representing the book's author. Define methods to display book details along with author details. Create instances and display their relationships.

```
class Author:
    def __init__(self, author_name):
        self.author_name = author_name

    def get_author(self):
        return self.author_name

class Book:
    def __init__(self, book_name, author):
        self.book_name = book_name
        self.author = author

    def display(self):
        print("Author:", self.author.get_author())
        print("Book Name:", self.book_name)

a = Author("NKV")
b = Book("Agni", a)

b.display()
```

Output:

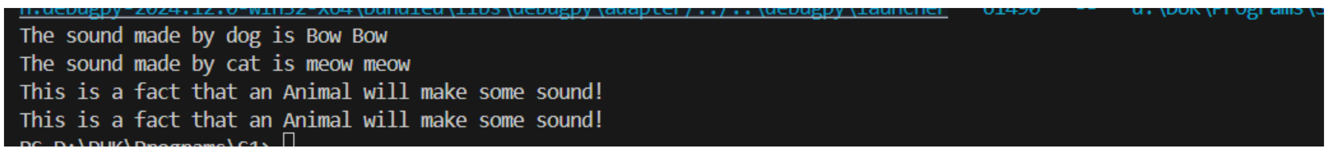


```
Author: NKV
Book Name: Agni
```

10. Create a base class Animal with attributes name and species and a method sound() that prints a generic message. Create subclasses Dog and Cat that override the sound() method to print species-specific sounds. Instantiate each subclass and call the sound() method.

```
class Animal:
    def __init__(self,name,species):
        self.name = name
        self.species = species

    def sound(self):
        print("This is a fact that an Animal will make some sound!")

class Dog(Animal):
    def __init__(self):
        pass

    def dog_sound(self):
        print("The sound made by dog is Bow Bow")

class Cat(Animal):
    def __init__(self):
        pass

    def cat_sound(self):
        print("The sound made by cat is meow meow")

#a = Animal("Jacky","Dog")
d = Dog()
c = Cat()

#a.sound()
d.dog_sound()
c.cat_sound()

d.sound()
c.sound()
```

Output:



```
n.debugpy-2024.12.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '61490' '--' 'd:\DUK\Programs\
The sound made by dog is Bow Bow
The sound made by cat is meow meow
This is a fact that an Animal will make some sound!
This is a fact that an Animal will make some sound!
PS D:\DUK\Programs\C1>
```