Hands-on_ObjectsAndClasses

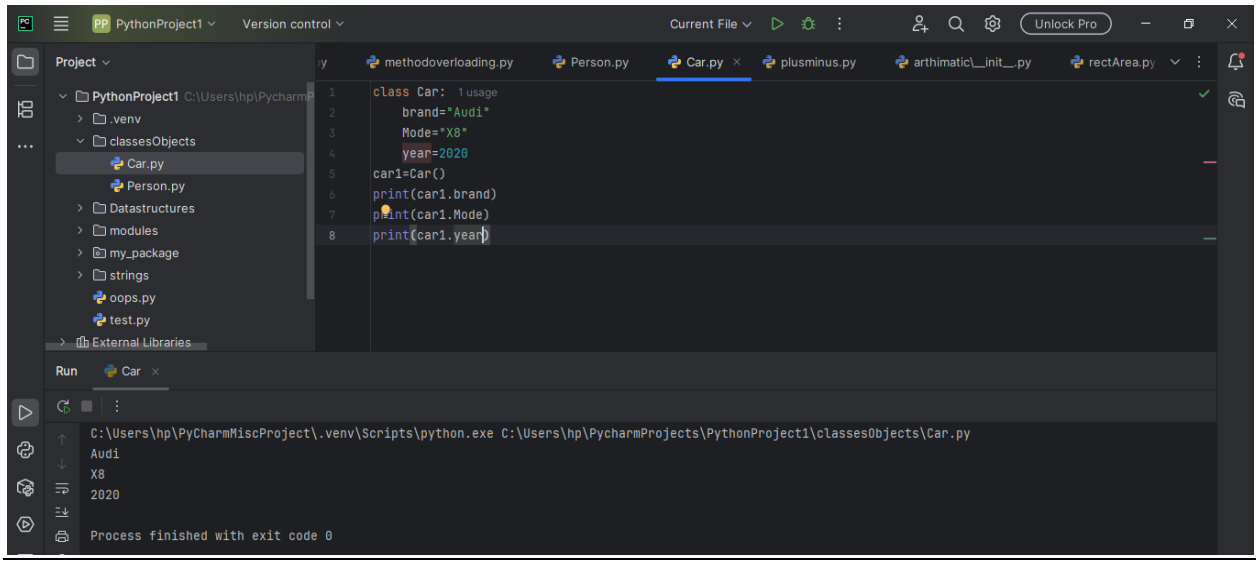
Exercise 1: Create a class `Person` with attributes `name` and `age`

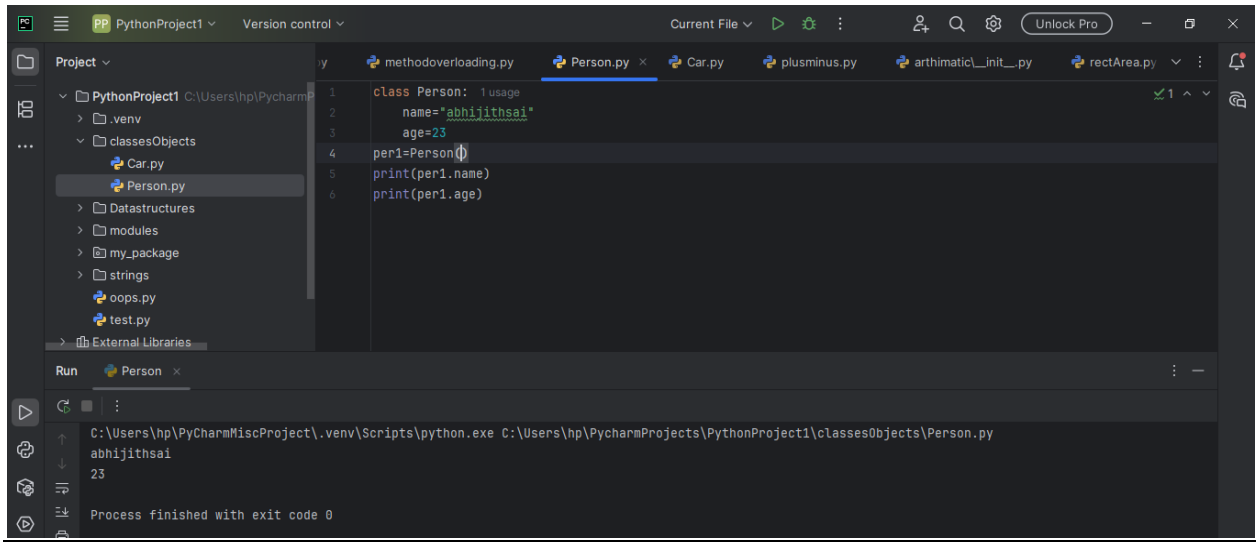
➢ Instantiate an object of the class and print its attributes

Exercise 2: Create a class `Car` with attributes `brand`, `model`, and `year`

➢ Instantiate an object of the class and print the car's details

Code





Exercise 3: Create a class `Book` with a constructor that accepts `title` and `author`

➢ Instantiate the class and print the book's title and author

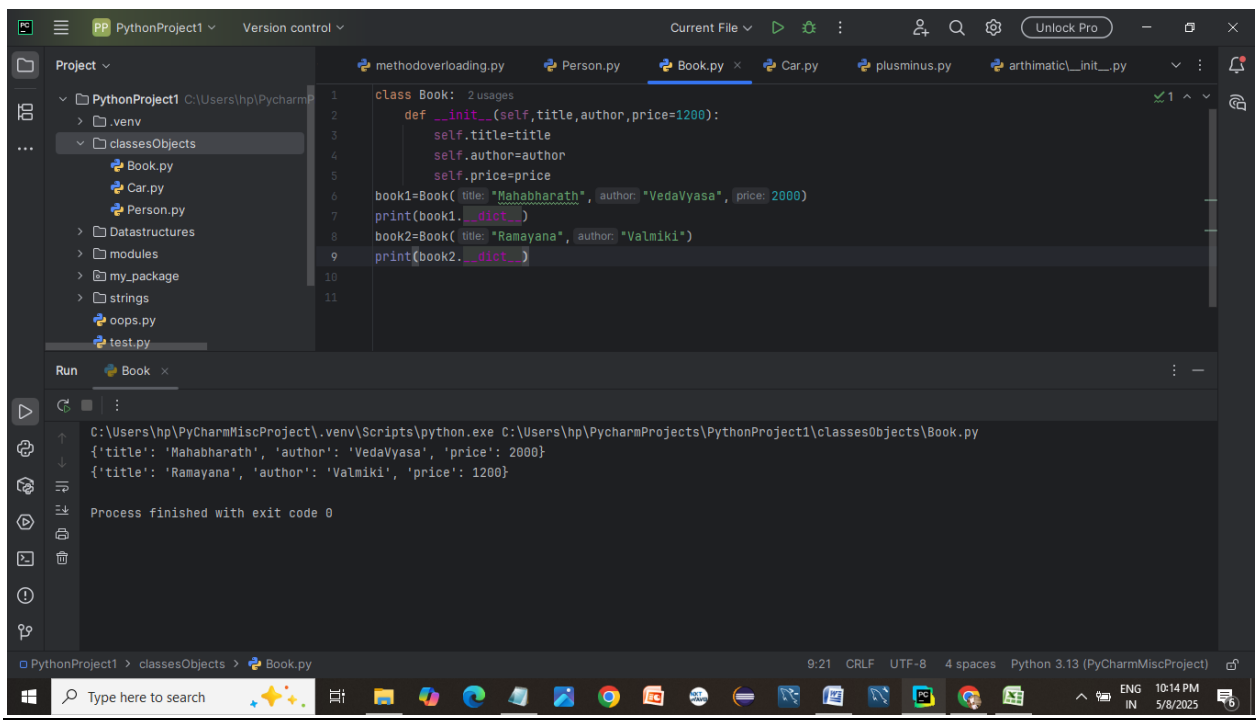
Exercise 4: Add an additional attribute `price` to the `Book` class and set a default price

➢ Print the price along with the title and author

Code

```
class Book:
    def __init__(self,title,author,price=1200):
        self.title=title
        self.author=author
        self.price=price
book1=Book("Mahabharath","VedaVyasa",2000)
print(book1.__dict__)
book2=Book("Ramayana","Valmiki")
print(book2.__dict__)
```

Output



Exercise 5: Create a class `Rectangle` with a method `area` to calculate the area of the rectangle

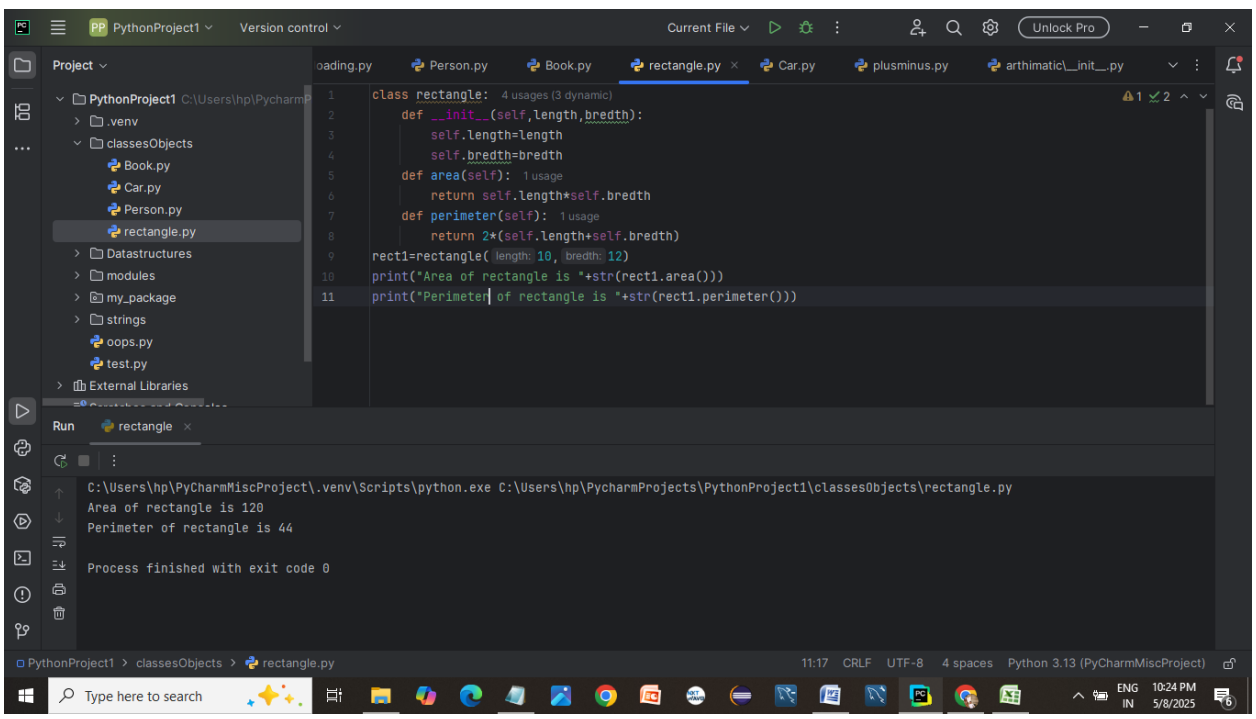
➢ Instantiate a `Rectangle` object and call the `area` method

Exercise 6: Add a method `perimeter` to the `Rectangle` class to calculate the perimeter of the rectangle

➢ Instantiate the object and call the `perimeter` method

```
class rectangle:
    def __init__(self,length,bredth):
        self.length=length
        self.bredth=bredth
    def area(self):
        return self.length*self.bredth
    def perimeter(self):
        return 2*(self.length+self.bredth)
rect1=rectangle(10,12)
print("Area of rectangle is "+str(rect1.area()))
print("Perimeter of rectangle is "+str(rect1.perimeter()))
```

Output



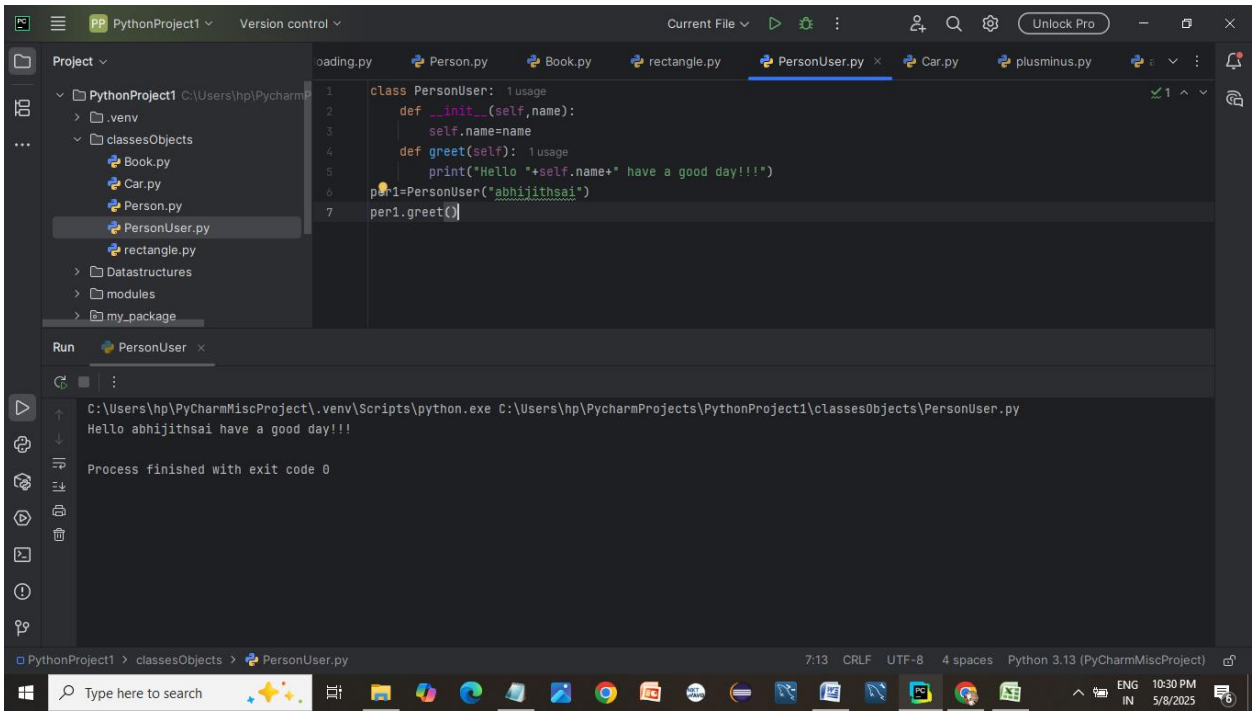
Exercise 7: Create a class `Person` with a method `greet` that prints a greeting with the person's name

➢ Call the `greet` method on an instance of `Person` to display a greeting

<u>Code</u>

```
class PersonUser:
    def   init  (self,name):
        self.name=name
    def greet(self):
        print("Hello "+self.name+" have a good day!!!")
per1=PersonUser("abhijithsai")
per1.greet()
```

Output



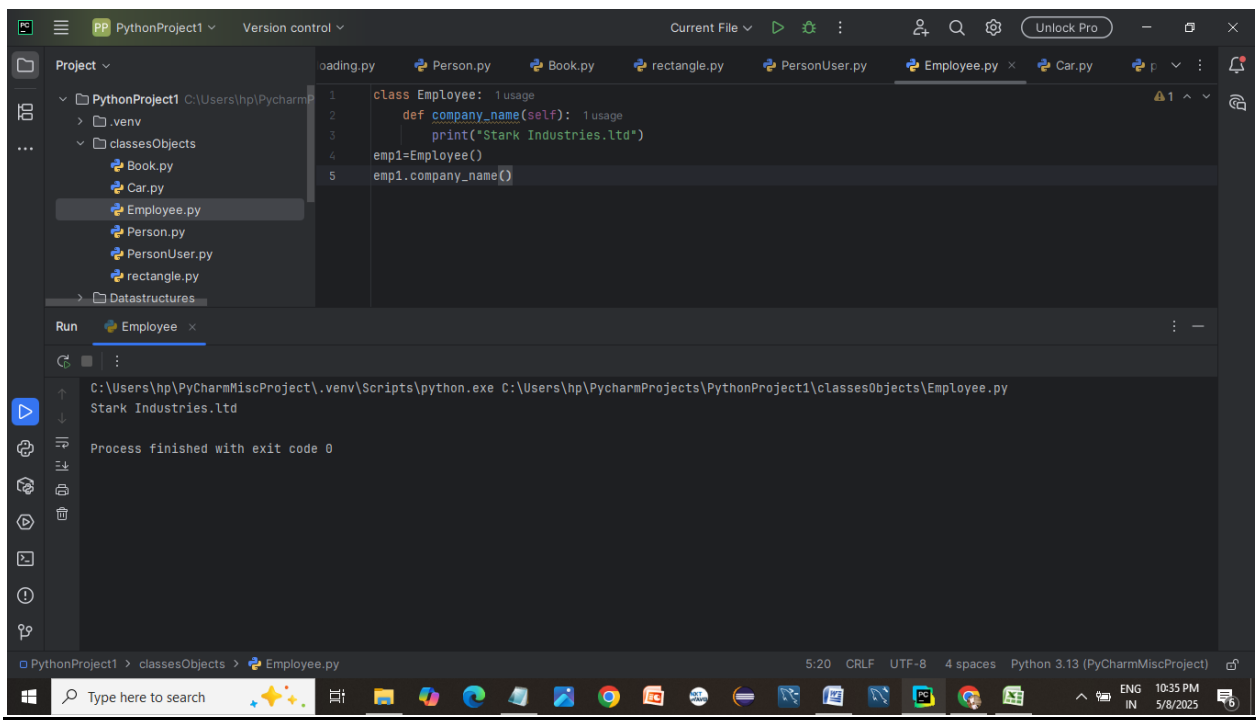
Exercise 8: Create a class `Employee` with a class method `company_name` that returns the company's name

➢ Call the class method directly using the class name

Code)

```
class Employee:
    def company_name(self):
        print("Stark Industries.ltd")
emp1=Employee()
emp1.company_name()
```

Output

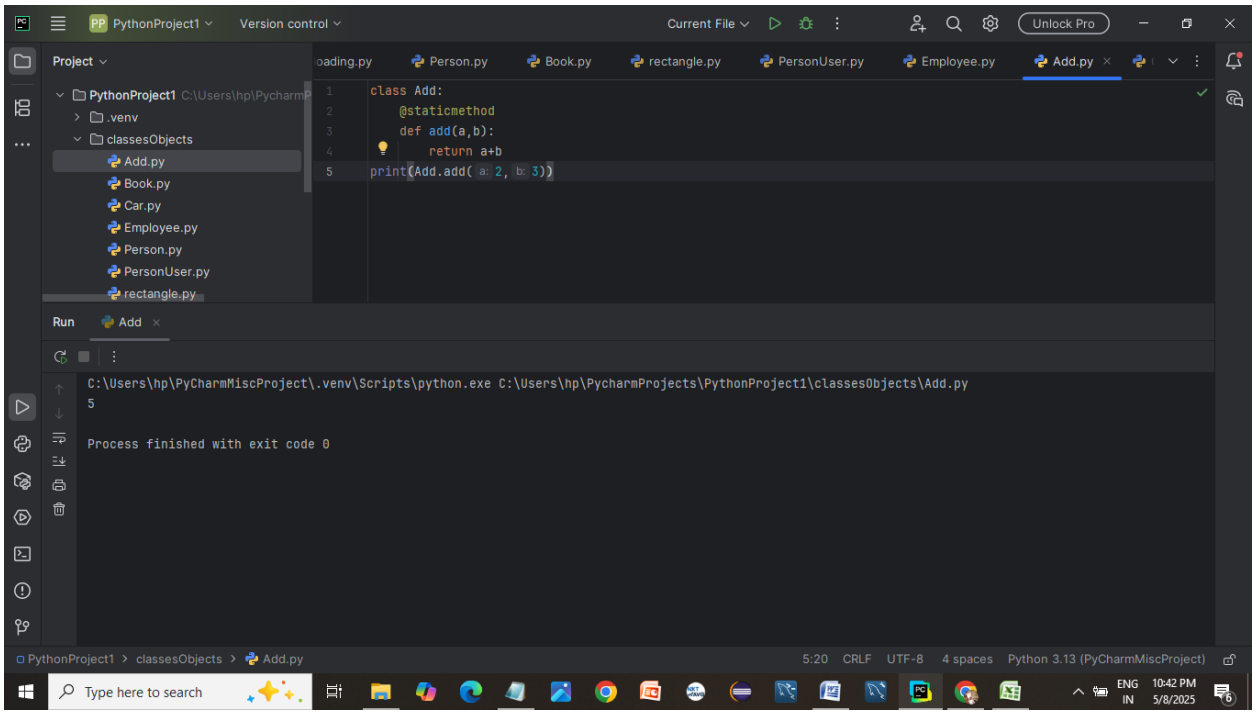


Exercise 9: Create a class `MathOperations` with a static method `add` that adds two numbers

➢ Call the static method without creating an instance

```
class Add:
    @staticmethod
    def add(a,b):
        return a+b
print(Add.add(2,3))
```

Output



Exercise 10: Create a class `Animal` with a method `speak` that prints "Animal speaks"

➢ Create a subclass `Dog` that inherits from `Animal` and override the `speak` method to print "Bark"

```
class Animal:
    def speak(self):
        print("Animal speak")
class Dog(Animal):
    def speak(self):
        print("DogsBark")
obj1=Animal()
obj1.speak()
obj2=Dog()
obj2.speak()
```

Output)