

Hands-on ObjectsAndClasses

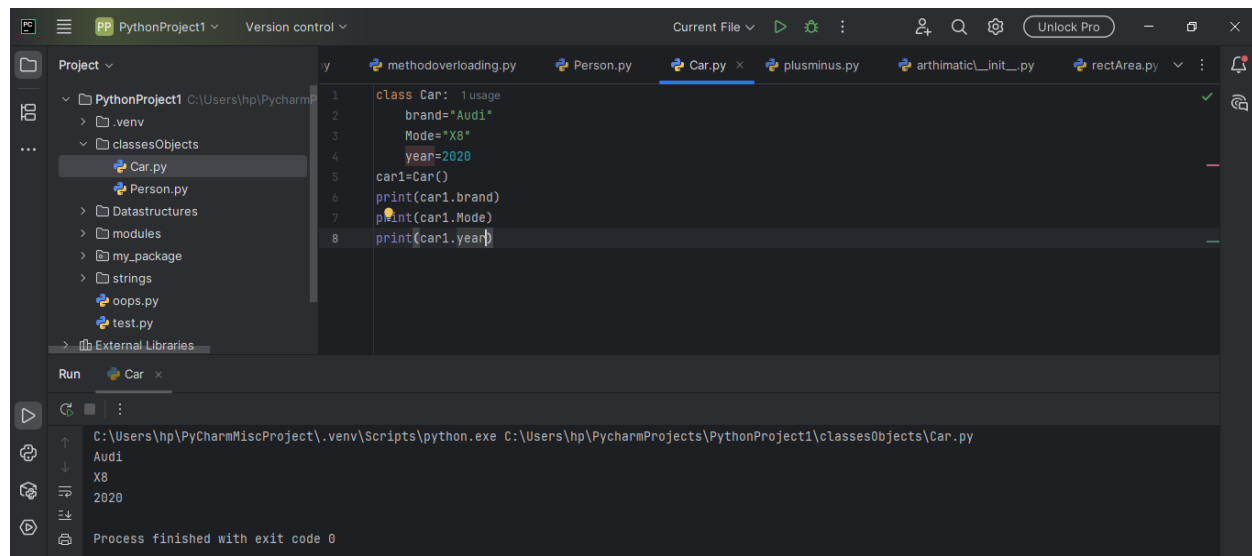
Exercise 1: Create a class `Person` with attributes `name` and `age`

➤ 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



The screenshot shows the PyCharm IDE with the file `Car.py` open. The code defines a `Car` class with attributes `brand`, `model`, and `year`. An instance `car1` is created, and its attributes are printed. The Run window shows the output: `Audi`, `X8`, and `2020`.

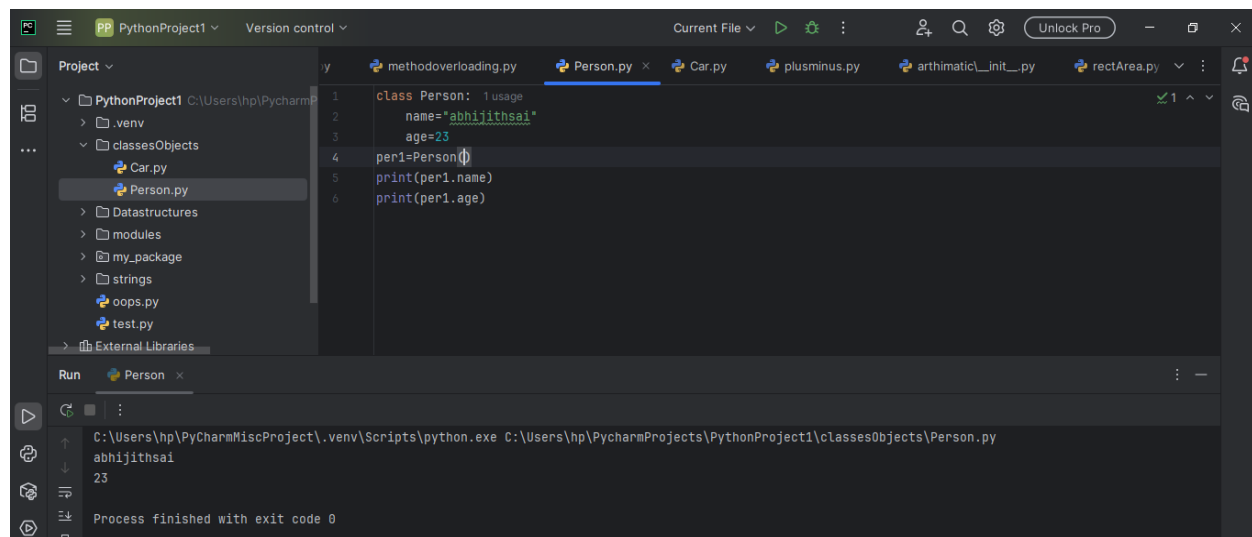
```
1 class Car:
2     brand="Audi"
3     model="X8"
4     year=2020
5     car1=Car()
6     print(car1.brand)
7     print(car1.model)
8     print(car1.year)
```

Run: Car

C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\classesObjects\Car.py

Audi
X8
2020

Process finished with exit code 0



The screenshot shows the PyCharm IDE with the file `Person.py` open. The code defines a `Person` class with attributes `name` and `age`. An instance `per1` is created, and its attributes are printed. The Run window shows the output: `abhiijithsai` and `23`.

```
1 class Person:
2     name="abhiijithsai"
3     age=23
4     per1=Person()
5     print(per1.name)
6     print(per1.age)
```

Run: Person

C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\classesObjects\Person.py

abhiijithsai
23

Process finished with exit code 0

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

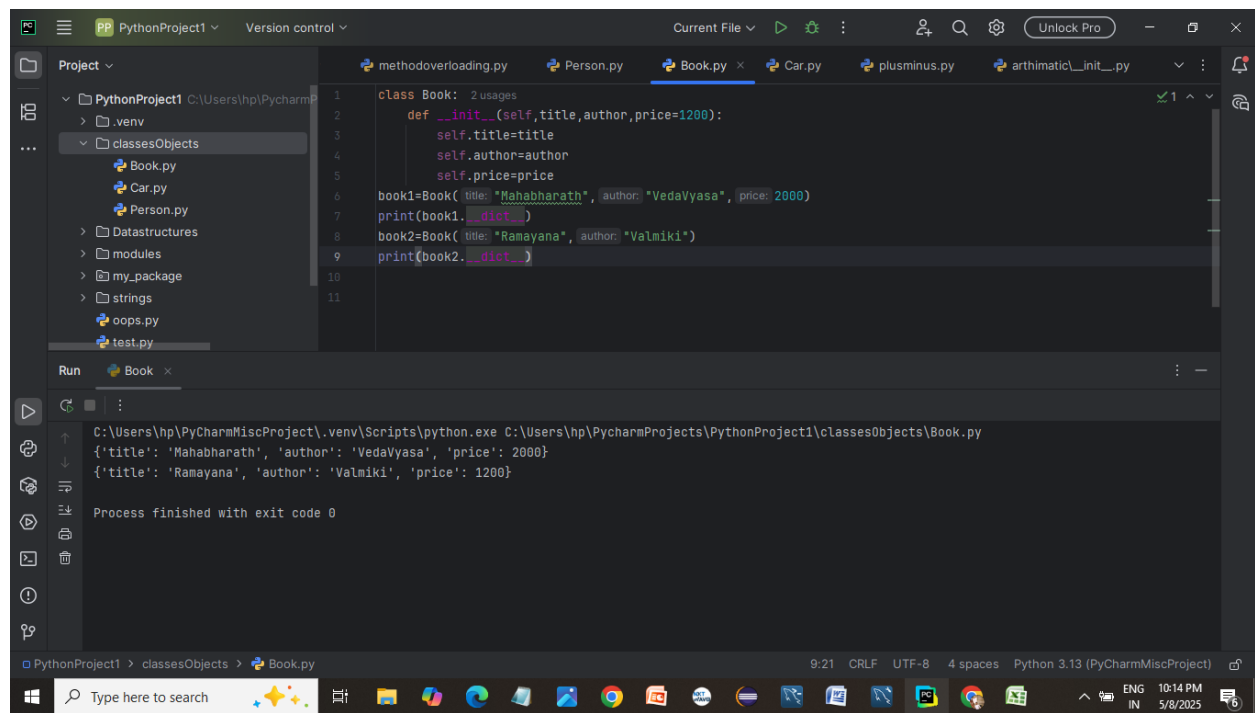
Exercise 4: Add an additional attribute `price` to the `Book` class and set a default price

➤ 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



The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for file operations, running, and debugging, along with a 'Unlock Pro' button. The left sidebar displays the project structure for 'PythonProject1', showing folders like 'venv' and 'classesObjects', and files like 'Book.py', 'Car.py', 'Person.py', 'Datastructures', 'modules', 'my_package', 'strings', 'oops.py', and 'test.py'. The main editor window shows the code from the previous block, with line numbers 1 through 11. The bottom pane is the 'Run' window, which shows the command executed: 'C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\classesObjects\Book.py'. The output of the program is displayed below the command:
{'title': 'Mahabharath', 'author': 'VedaVyasa', 'price': 2000}
{'title': 'Ramayana', 'author': 'Valmiki', 'price': 1200}
The process finished with exit code 0. The status bar at the bottom indicates the current file is 'Book.py' in the 'classesObjects' directory, with a timestamp of 9:21, encoding of CRLF, UTF-8, 4 spaces, and Python 3.13 (PyCharmMiscProject).

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

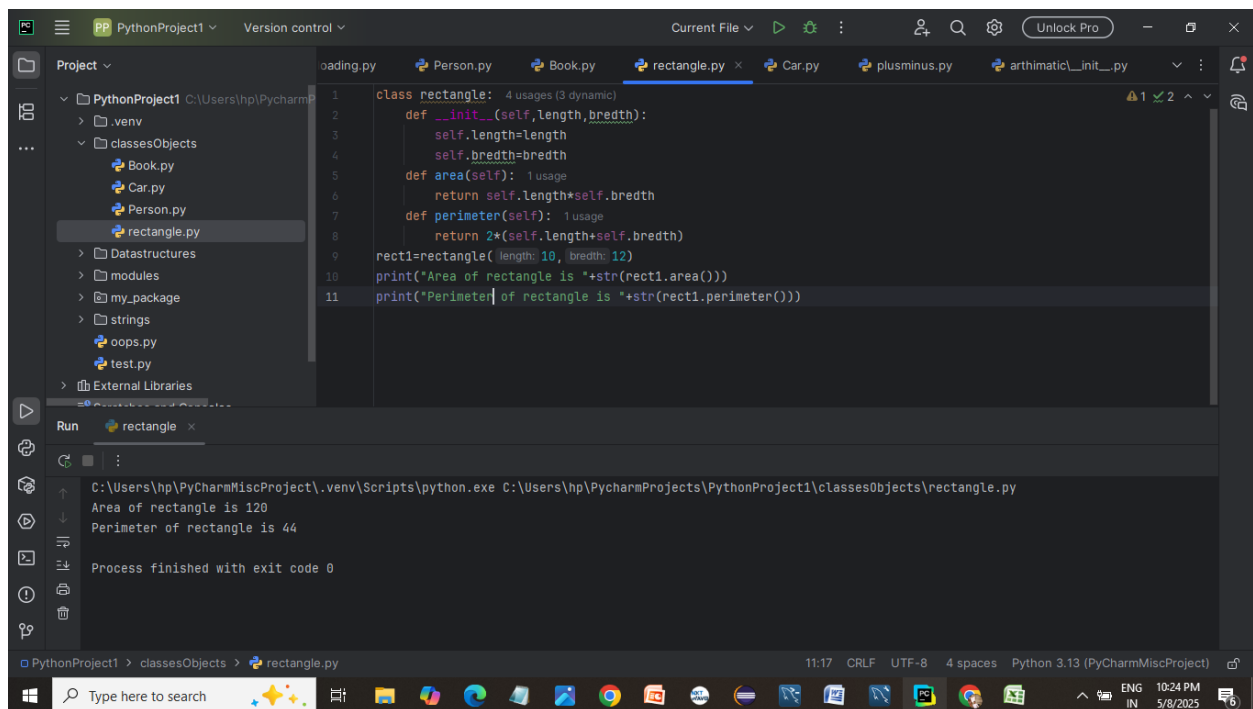
➤ Instantiate a `Rectangle` object and call the `area` method

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



The screenshot shows the PyCharm IDE interface. The main editor window displays the Python code for the `rectangle` class, including the `__init__`, `area`, and `perimeter` methods, and the instantiation of a `rect1` object with dimensions 10 and 12. The code uses `length` and `bredth` (sic) as attributes. The Run window at the bottom shows the output of the program: "Area of rectangle is 120" and "Perimeter of rectangle is 44". The status bar at the bottom indicates the file is `rectangle.py` in the `classesObjects` directory, using Python 3.13.

```
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()))
```

Run output:

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\classesObjects\rectangle.py
Area of rectangle is 120
Perimeter of rectangle is 44
Process finished with exit code 0
```

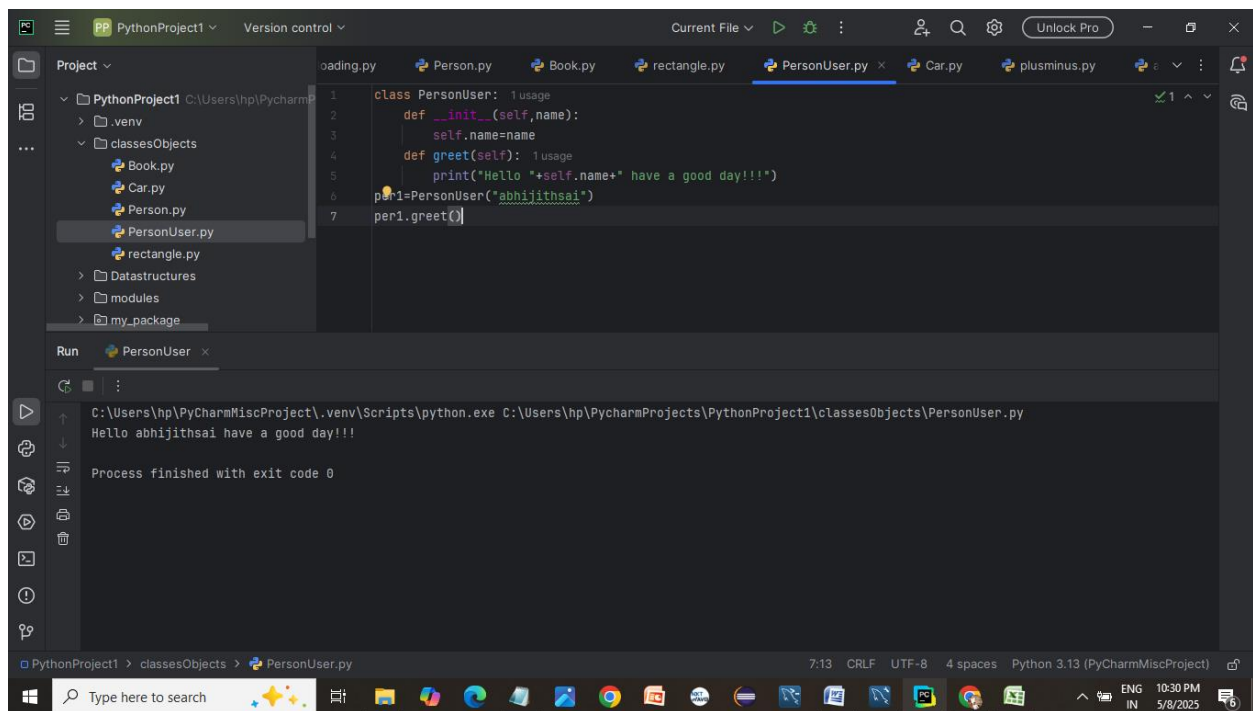
Exercise 7: Create a class `Person` with a method `greet` that prints a greeting with the person's name

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

Code

```
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



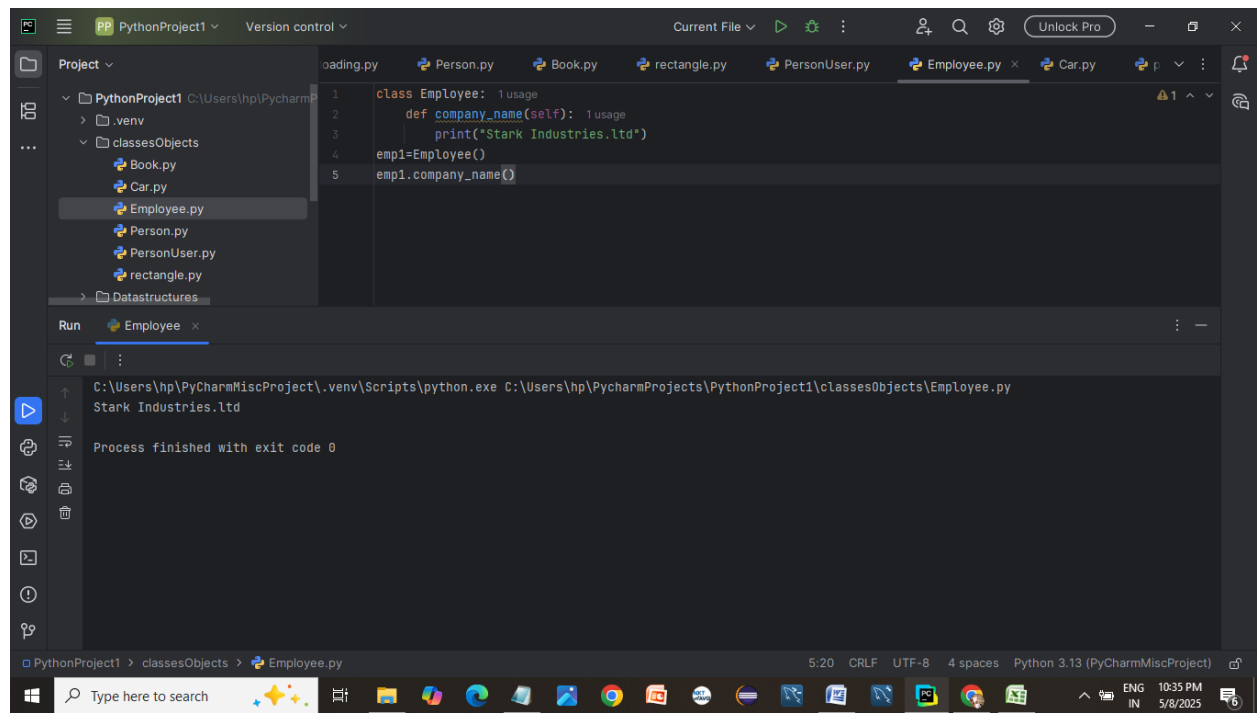
Exercise 8: Create a class `Employee` with a class method `company_name` that returns the company's name

➤ 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

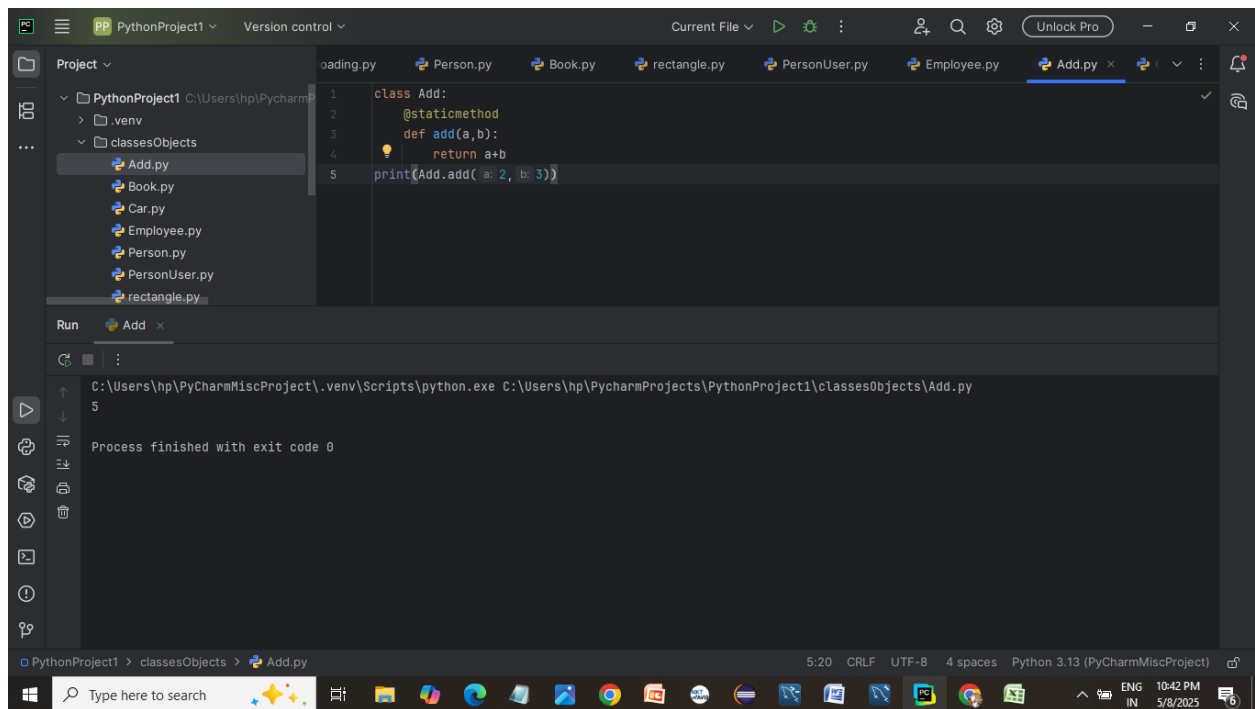


Exercise 9: Create a class `MathOperations` with a static method `add` that adds two numbers

➤ Call the static method without creating an instance

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

Output



The screenshot shows the PyCharm IDE interface. The top toolbar includes buttons for running and debugging. The left sidebar shows the project structure with a folder named 'classesObjects' containing several Python files, including 'Add.py'. The main editor window displays the code for 'Add.py', which defines a class 'Add' with a static method 'add' and a call to 'Add.add(2, 3)'. The bottom pane shows the output of the program, which is the number '5'. The status bar at the bottom indicates the file encoding (UTF-8), indentation (4 spaces), and the Python version (3.13).

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

Run

C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\classesObjects\Add.py

5

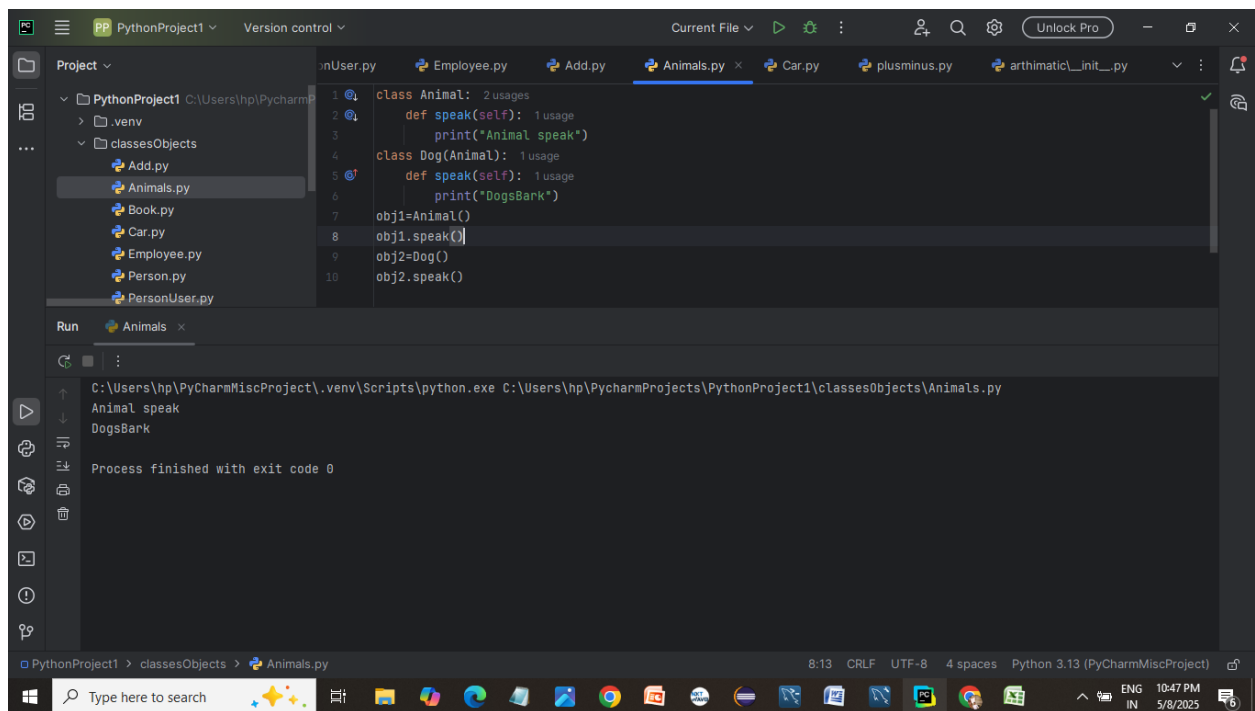
Process finished with exit code 0

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)



The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for file operations, search, and a 'Unlock Pro' button. The left sidebar displays the project structure for 'PythonProject1', showing a 'classesObjects' directory containing several Python files, including 'Animals.py'. The main editor window displays the code from the previous block, with line numbers 1 through 10. The 'Run' button (a green play icon) is visible below the code editor. The bottom pane shows the output of the program: 'Animal speak' followed by 'DogsBark' on a new line. Below the output, it states 'Process finished with exit code 0'. The status bar at the very bottom indicates the file path, encoding (UTF-8), and the Python version (3.13).

```
1 class Animal:
2     def speak(self):
3         print("Animal speak")
4 class Dog(Animal):
5     def speak(self):
6         print("DogsBark")
7 obj1=Animal()
8 obj1.speak()
9 obj2=Dog()
10 obj2.speak()
```

Run Animals

C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\classesObjects\Animals.py

Animal speak
DogsBark

Process finished with exit code 0