

ASSIGNMENT 6 – OOPS

1. Build a program to manage a university's course catalog. You want to define a base class `Course` that has the following properties:

- `course_code`: a string representing the course code (e.g., "CS101")
- `course_name`: a string representing the course name (e.g., "Introduction to Computer Science")
- `credit_hours`: an integer representing the credit hours for the course (e.g., 3)

You also want to define two subclasses `CoreCourse` and `ElectiveCourse`, which inherit from the `Course` class. `CoreCourse` should have an additional property `required_for_major` which is a boolean representing whether the course is required for a particular major. `ElectiveCourse` should have an additional property `elective_type` which is a string representing the type of elective (e.g., "general", "technical", "liberal arts").

```
class Course:
```

```
    def __init__(self, course_code, course_name, credit_hours):  
        self.course_code = course_code  
        self.course_name = course_name  
        self.credit_hours = credit_hours
```

```
class CoreCourse(Course):
```

```
    def __init__(self, course_code, course_name, credit_hours, required_for_major):  
        Course.__init__(self, course_code, course_name, credit_hours)  
        self.required_for_major = required_for_major
```

```
class ElectiveCourse(Course):
```

```
    def __init__(self, course_code, course_name, credit_hours, elective_type):
```

```
        Course.__init__(self, course_code, course_name, credit_hours)
```

```
        self.elective_type = elective_type
```

```
core_course1 = CoreCourse("CS101", "Introduction to Computer Science", 3, True)
```

```
print(f"\nCore Course: {core_course1.course_code} - {core_course1.course_name}")
```

```
print(f"Credit Hours: {core_course1.credit_hours}")
```

```
print(f"Required for Major: {'Yes' if core_course1.required_for_major else 'No'}")
```

```
elective_course1 = ElectiveCourse("MATH201", "Calculus", 4, "technical")
```

```
print(f"\nElective Course: {elective_course1.course_code} -  
{elective_course1.course_name}")
```

```
print(f"Credit Hours: {elective_course1.credit_hours}")
```

```
print(f"Elective Type: {elective_course1.elective_type}")
```

The screenshot shows the PyCharm IDE interface. The main editor window displays a Python script named `ASSIGNMENT_6_OOPS.py`. The script defines a `class ElectiveCourse(Course):` with an `__init__` method that calls `Course.__init__` and sets `self.elective_type`. It then creates two objects: `core_course1` (a `CoreCourse` object) and `elective_course1` (an `ElectiveCourse` object). The script prints details for both courses. The left sidebar shows the project structure with various files and folders. The bottom panel shows the output of the script, which matches the printed output in the code.

```
class ElectiveCourse(Course):
    def __init__(self, course_code, course_name, credit_hours, elective_type):
        Course.__init__(self, course_code, course_name, credit_hours)
        self.elective_type = elective_type

core_course1 = CoreCourse(course_code="CS101", course_name="Introduction to Computer Science", credit_hours=3, required_for_major=True)
print(f"\nCore Course: {core_course1.course_code} - {core_course1.course_name}")
print(f"Credit Hours: {core_course1.credit_hours}")
print(f"Required for Major: {'Yes' if core_course1.required_for_major else 'No'}")

elective_course1 = ElectiveCourse(course_code="MATH201", course_name="Calculus", credit_hours=4, elective_type="technical")
print(f"\nElective Course: {elective_course1.course_code} - {elective_course1.course_name}")
print(f"Credit Hours: {elective_course1.credit_hours}")
print(f"Elective Type: {elective_course1.elective_type}")
```

Run: ASSIGNMENT_6_OOPS x

C:\Users\SIXCOUSER\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\SIXCOUSER\PycharmProjects\pythonProject\.venv\Scripts\ASSIGNMENT_6_OOPS.py

```
Core Course: CS101 - Introduction to Computer Science
Credit Hours: 3
Required for Major: Yes

Elective Course: MATH201 - Calculus
Credit Hours: 4
Elective Type: technical

Process finished with exit code 0
```

2. Create a Python module named employee that contains a class Employee with attributes name, salary and methods get_name() and get_salary(). Write a program to use this module to create an object of the Employee class and display its name and salary.

employee.py

class Employee:

```
def __init__(self, name, salary):
```

```
    self.name = name
```

```
    self.salary = salary
```

```
def get_name(self):
```

```
    return self.name
```

```
def get_salary(self):
```

```
    return self.salary
```

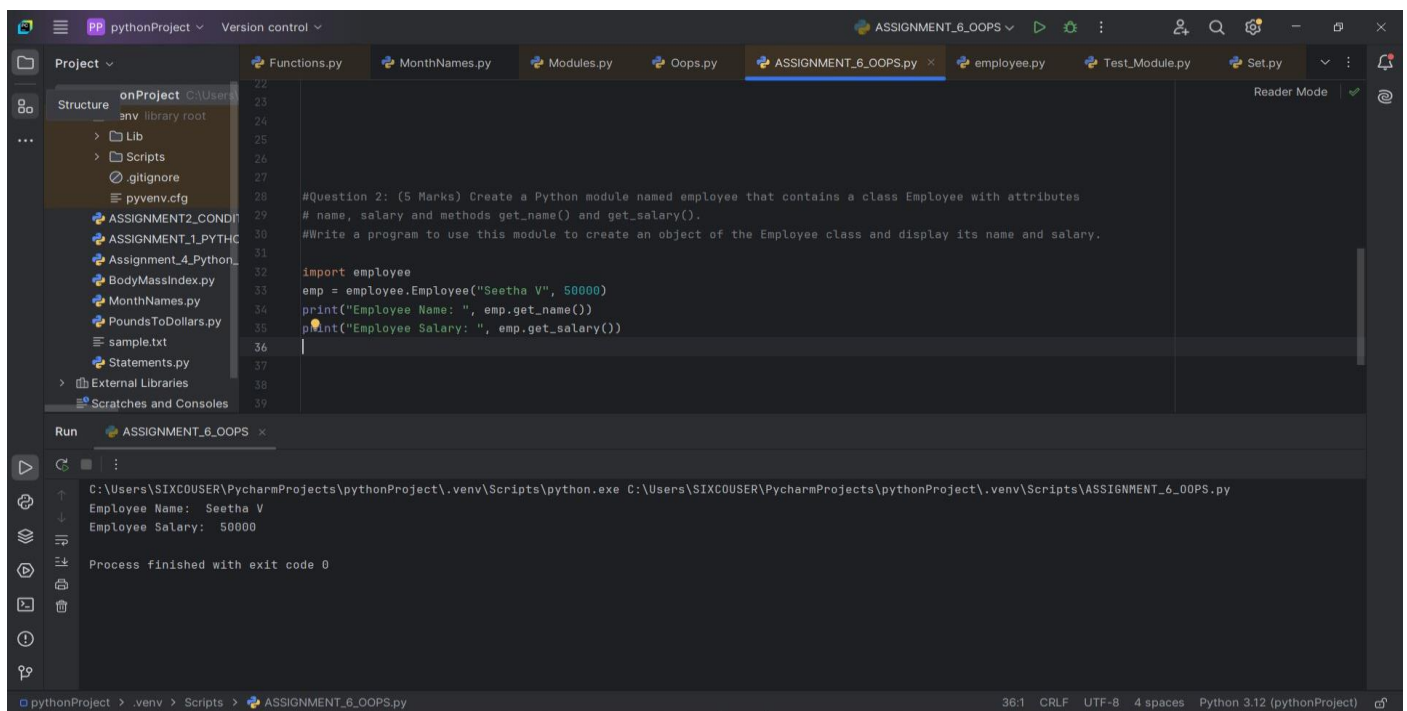
ASSIGNMENT 6 OOPS.py

```
import employee
```

```
emp = employee.Employee("Seetha V", 50000)
```

```
print("Employee Name: ", emp.get_name())
```

```
print("Employee Salary: ", emp.get_salary())
```



```
22
23
24
25
26
27
28 #Question 2: (5 Marks) Create a Python module named employee that contains a class Employee with attributes
29 # name, salary and methods get_name() and get_salary().
30 #Write a program to use this module to create an object of the Employee class and display its name and salary.
31
32 import employee
33 emp = employee.Employee("Seetha V", 50000)
34 print("Employee Name: ", emp.get_name())
35 print("Employee Salary: ", emp.get_salary())
36
37
38
39
```

Run

```
C:\Users\SIXCOUSER\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\SIXCOUSER\PycharmProjects\pythonProject\.venv\Scripts\ASSIGNMENT_6_OOPS.py
Employee Name: Seetha V
Employee Salary: 50000
Process finished with exit code 0
```

pythonProject > .venv > Scripts > ASSIGNMENT_6_OOPS.py 36:1 CRLF UTF-8 4 spaces Python 3.12 (pythonProject)

