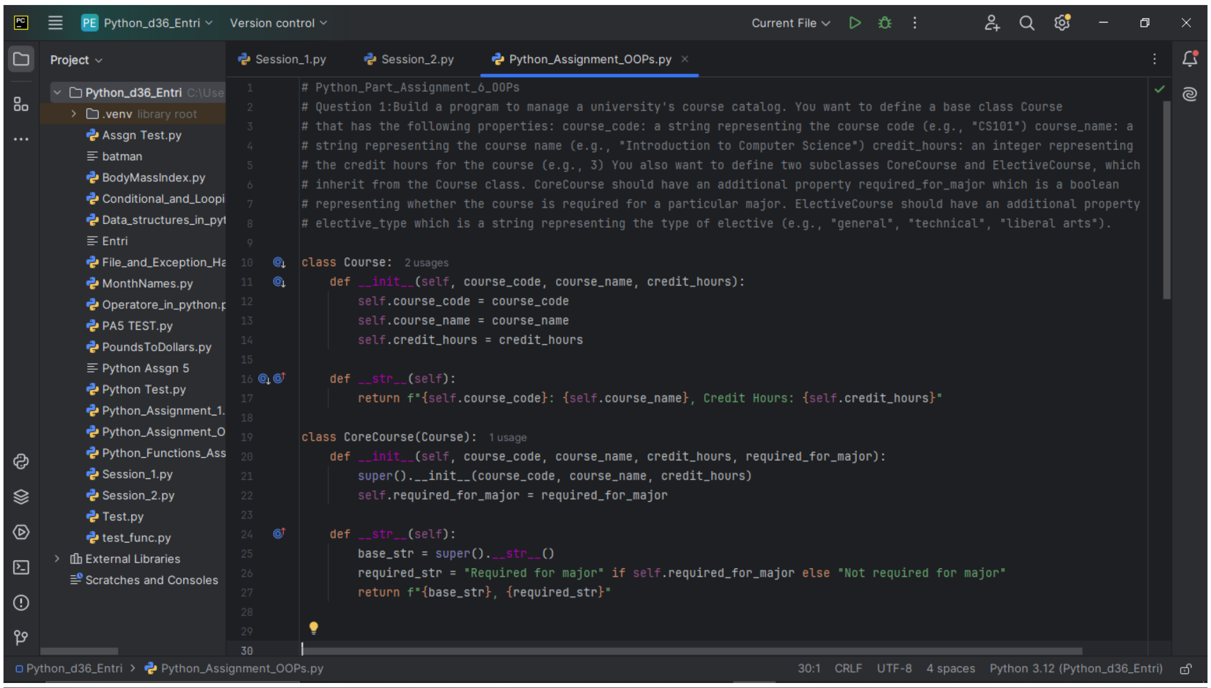
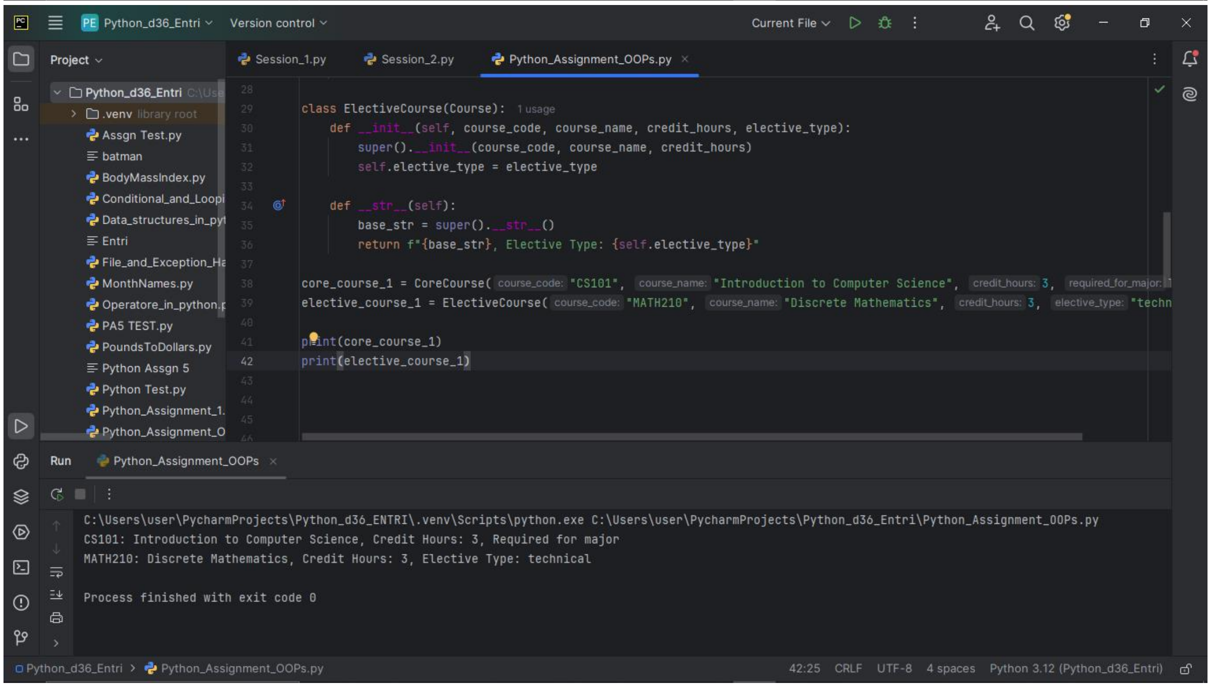
Python Assignment: 6 - OOPS

Question 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").



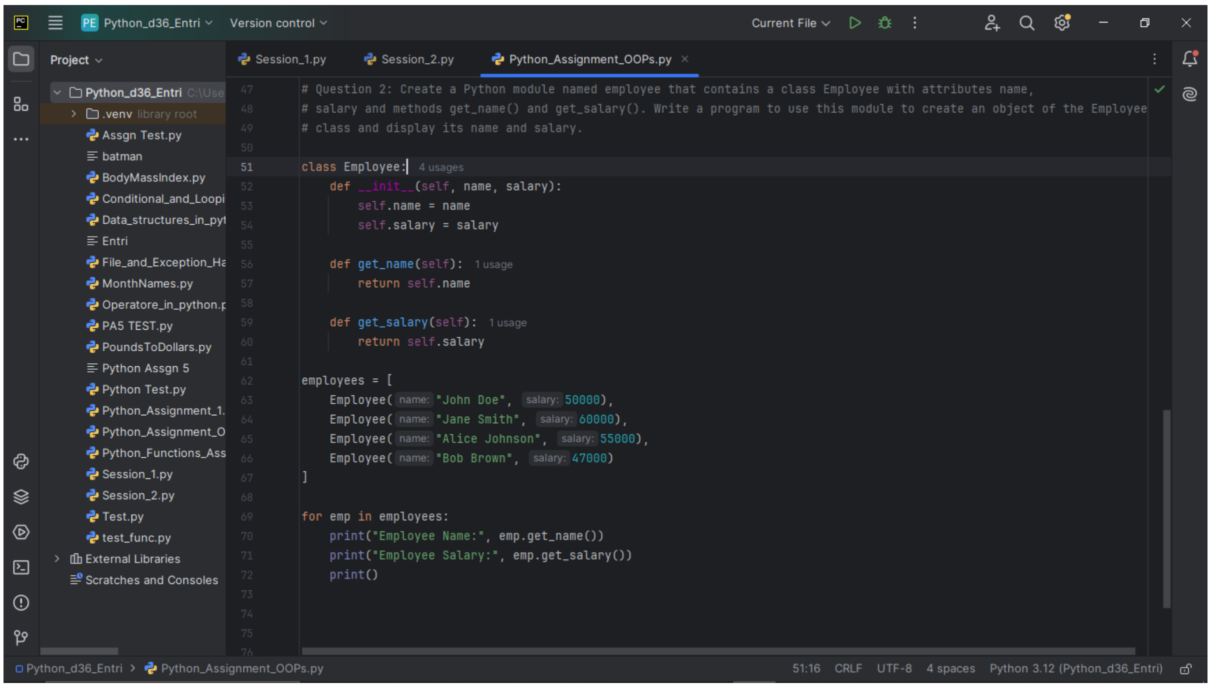
```python
# Python_Part_Assignment_6_OOPs
# Question 1:Build a program to manage a university's course catalog. You want to define a base class Course
# that has the following properties: course_code: a string representing the course code (e.g., "CS101") course_name: a
# string representing the course 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:  2 usages
    def __init__(self, course_code, course_name, credit_hours):
        self.course_code = course_code
        self.course_name = course_name
        self.credit_hours = credit_hours

    def __str__(self):
        return f"{self.course_code}: {self.course_name}, Credit Hours: {self.credit_hours}"

class CoreCourse(Course):  1 usage
    def __init__(self, course_code, course_name, credit_hours, required_for_major):
        super().__init__(course_code, course_name, credit_hours)
        self.required_for_major = required_for_major

    def __str__(self):
        base_str = super().__str__()
        required_str = "Required for major" if self.required_for_major else "Not required for major"
        return f"{base_str}, {required_str}"
```



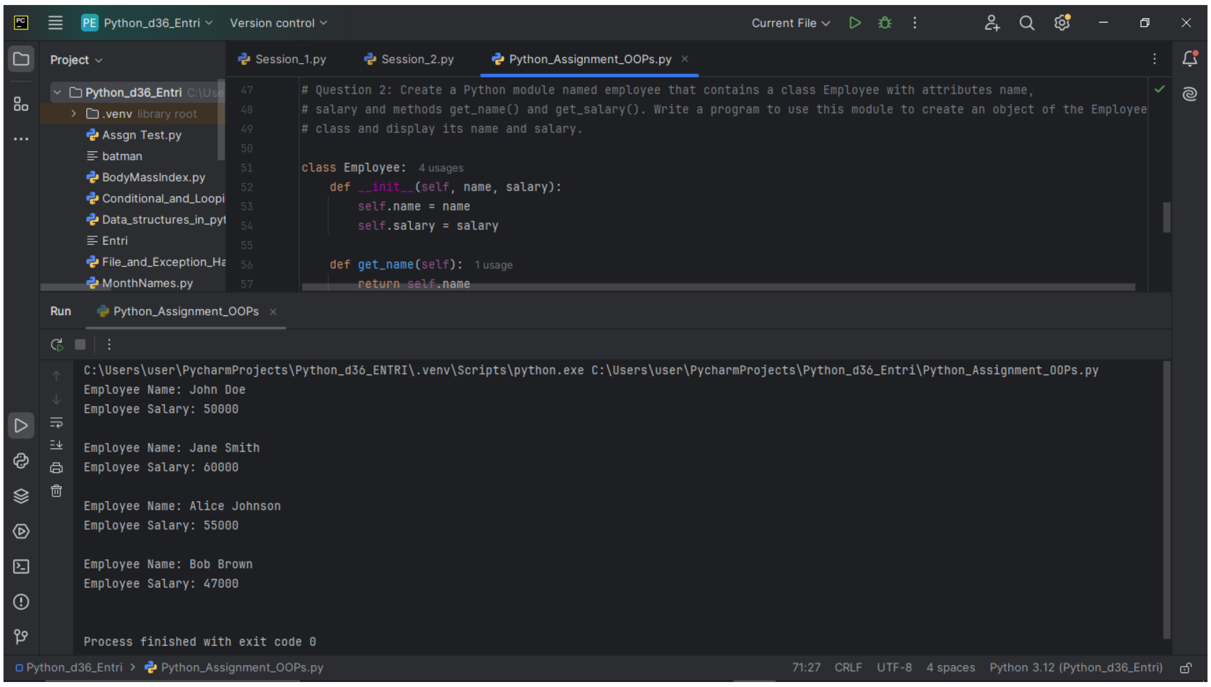
```python
class ElectiveCourse(Course):  1 usage
    def __init__(self, course_code, course_name, credit_hours, elective_type):
        super().__init__(course_code, course_name, credit_hours)
        self.elective_type = elective_type

    def __str__(self):
        base_str = super().__str__()
        return f"{base_str}, Elective Type: {self.elective_type}"

core_course_1 = CoreCourse( course_code: "CS101", course_name: "Introduction to Computer Science", credit_hours: 3, required_for_major: 
elective_course_1 = ElectiveCourse( course_code: "MATH210", course_name: "Discrete Mathematics", credit_hours: 3, elective_type: "techn

print(core_course_1)
print(elective_course_1)
```

```
C:\Users\user\PycharmProjects\Python_d36_ENTRI\.venv\Scripts\python.exe C:\Users\user\PycharmProjects\Python_d36_Entri\Python_Assignment_OOPs.py
CS101: Introduction to Computer Science, Credit Hours: 3, Required for major
MATH210: Discrete Mathematics, Credit Hours: 3, Elective Type: technical

Process finished with exit code 0
```

Question 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.



```python
# Question 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.

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

employees = [
    Employee("John Doe", 50000),
    Employee("Jane Smith", 60000),
    Employee("Alice Johnson", 55000),
    Employee("Bob Brown", 47000)
]

for emp in employees:
    print("Employee Name:", emp.get_name())
    print("Employee Salary:", emp.get_salary())
    print()
```



```python
# Question 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.

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

    def get_name(self):
        return self.name
```

```
C:\Users\user\PycharmProjects\Python_d36_ENTRI\.venv\Scripts\python.exe C:\Users\user\PycharmProjects\Python_d36_Entri\Python_Assignment_OOPs.py
Employee Name: John Doe
Employee Salary: 50000

Employee Name: Jane Smith
Employee Salary: 60000

Employee Name: Alice Johnson
Employee Salary: 55000

Employee Name: Bob Brown
Employee Salary: 47000


Process finished with exit code 0
```