

# employee-module-project

Exercise: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.

Code:

```
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
```

```
    def display_info(self): # Method to display course info
```

```
        print(f"Course Code: {self.course_code}")
```

```
        print(f"Course Name: {self.course_name}")
```

```
        print(f"Credit Hours: {self.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) # Calling  
        parent's __init__  
        self.required_for_major = required_for_major  
  
    def display_info(self): # Overriding display_info  
        Course.display_info(self) # Calling parent's display_info  
        print(f"Required for Major: {self.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) # Calling  
        parent's __init__  
        self.elective_type = elective_type  
  
    def display_info(self): # Overriding display_info  
        Course.display_info(self) # Calling parent's display_info  
        print(f"Elective Type: {self.elective_type}")
```

```
cs101 = CoreCourse("CS101", "Computer Science", 3, True)  
ac302 = CoreCourse("AC302", "Arithmetic Calculus ", 4, True)  
eng101 = ElectiveCourse("ENG101", "Creative Writing", 3, "liberal arts")
```

```
es205 = ElectiveCourse("ES205", "Embedded System", 4, "technical")
```

```
courses = [cs101, ac302, eng101, es205]
```

```
for course in courses:
```

```
    course.display_info() # call display_info() method
```

```
print("-" * 20) # Separator between courses
```

```
catalog = []
```

```
catalog.append(cs101)
```

```
catalog.append(eng101)
```

```
for course in catalog:
```

```
    print(f"Course Code: {course.course_code}")
```

```
    if isinstance(course, CoreCourse):
```

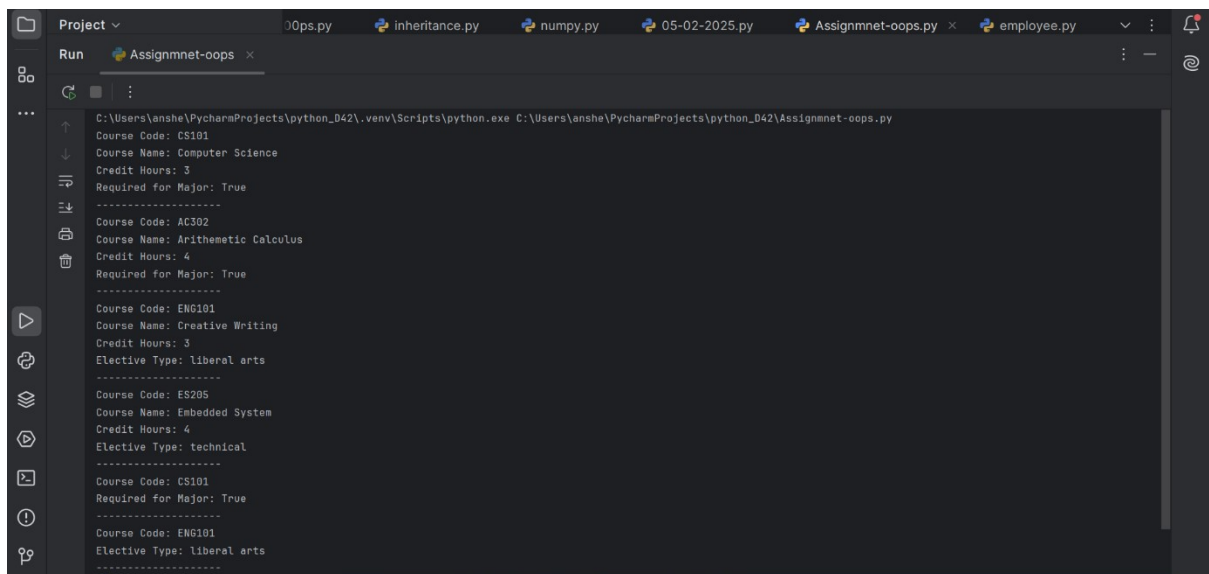
```
        print(f"Required for Major: {course.required_for_major}")
```

```
    elif isinstance(course, ElectiveCourse):
```

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

```
    print("-" * 20)
```

Result:



```
C:\Users\anshe\PycharmProjects\python_042\.venv\Scripts\python.exe C:\Users\anshe\PycharmProjects\python_042\Assignmnet-oops.py
Course Code: CS101
Course Name: Computer Science
Credit Hours: 3
Required for Major: True
-----
Course Code: AC302
Course Name: Arithmetic Calculus
Credit Hours: 4
Required for Major: True
-----
Course Code: ENG101
Course Name: Creative Writing
Credit Hours: 3
Elective Type: Liberal arts
-----
Course Code: ES205
Course Name: Embedded System
Credit Hours: 4
Elective Type: technical
-----
Course Code: CS101
Required for Major: True
-----
Course Code: ENG101
Elective Type: Liberal arts
-----
```

---

## Exercise :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.

Code:

Employee module:

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
```

module accessing:

```
import employee as em
```

```
emp1=em.Employee("ANU",30000)
```

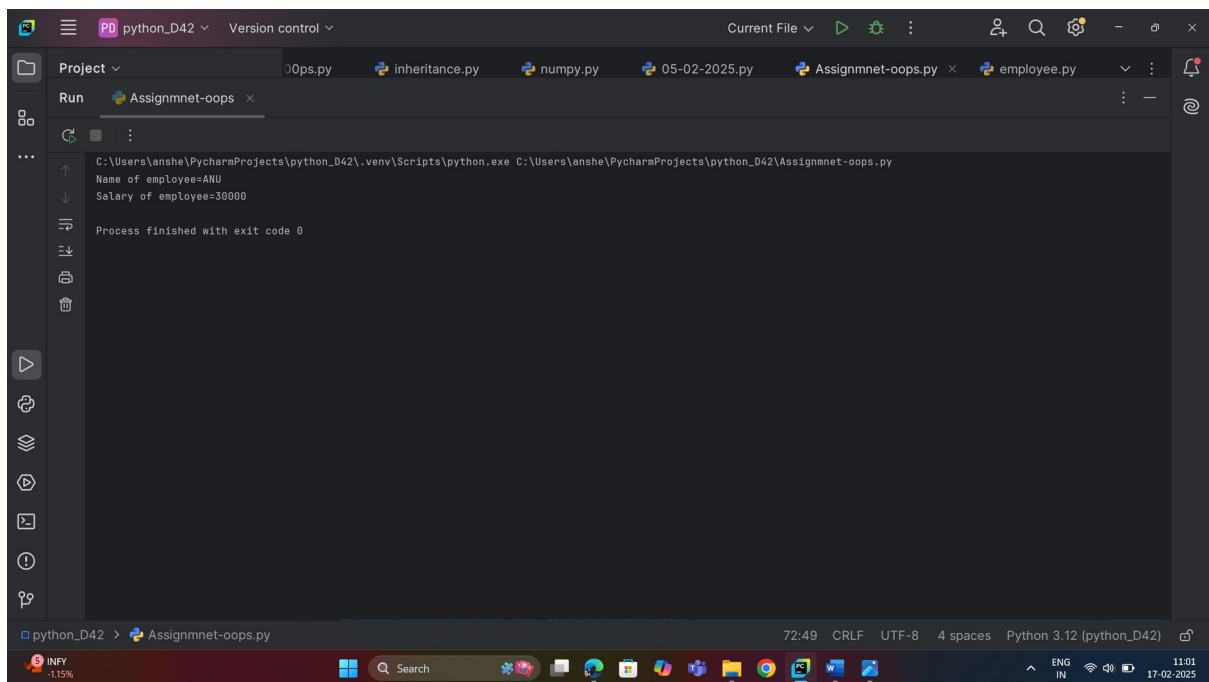
```
emp1.get_name()
```

```
emp1.get_salary()
```

```
print(f"Name of employee={emp1.get_name()}")
```

```
print(f"Salary of employee={emp1.get_salary()}")
```

Result:



```
python_D42  Version control  Current File  Run  Assignment-oops.py  inheritance.py  numpy.py  05-02-2025.py  Assignment-oops.py  employee.py  Run  Assignment-oops.py  C:\Users\anshe\PycharmProjects\python_D42\.venv\Scripts\python.exe C:\Users\anshe\PycharmProjects\python_D42\Assignment-oops.py  Name of employee=ANU  Salary of employee=30000  Process finished with exit code 0  python_D42  Assignment-oops.py  72:49  CRLF  UTF-8  4 spaces  Python 3.12 (python_D42)  11:01  17-02-2025
```