

Name: Muhammad Abdullah

Roll no: 016

Section: 3A

Department: BSDSM

Task #2: Artificial Intelligence Lab

Q1. Write a function `student_info` that accepts: `name` (positional) `roll_no` (positional) `department` (default = "DS") `courses` (*args) → any number of courses `extra_details` (**kwargs) → any extra info (e.g. `cgpa`, `city`)

| main.py | Output |
|---|--|
| <pre>1- def student_info(name, roll_no, department="DS", *courses, **extra_details): 2- print("Name:", name) 3- print("Roll No:", roll_no) 4- print("Department:", department) 5- 6- print("Courses:") 7- for course in courses: 8- print("-", course) 9- 10- print("Extra Details:") 11- for key, value in extra_details.items(): 12- print(f"{key}: ", value) 13- 14- student_info(15- "Jeffery Epstein", 16- 101, 17- "AI", 18- "Python", 19- "Data Science", 20- cgpa=3.8, 21- city="Epstein Island" 22-)</pre> | <pre>Name: Jeffery Epstein Roll No: 101 Department: AI Courses: - Python - Data Science Extra Details: cgpa: 3.8 city: Epstein Island === Code Execution Successful ===</pre> |

Q2. Write a small program containing:

1. A global variable `total_students = 120`
2. A function `outer()` that has a local variable `batch = "Fall 2025"`
3. Inside `outer()`, define a nested function `inner()` that uses the `nonlocal` keyword to modify `batch`
4. Show three `print()` statements inside `inner()`, `outer()` and outside everything to prove which value is visible/updated where.

```
main.py | Run | Share | Clear
1 total_students = 120
2
3 def outer():
4     batch = "Fall 2025"
5
6     def inner():
7         nonlocal batch
8         batch = "Spring 2026"
9         print("Inside inner():")
10        print("Batch =", batch)
11        print("Total Students =", total_students)
12
13    inner()
14    print("\nInside outer():")
15    print("Batch =", batch)
16
17
18 outer()
19
20 print("\nOutside everything:")
21 print("Total Students =", total_students)
```

```
Output
Inside inner():
Batch = Spring 2026
Total Students = 120

Inside outer():
Batch = Spring 2026

Outside everything:
Total Students = 120

=== Code Execution Successful ===
```

Q3. Design a class LibraryBook that represents one book in a university library.

Requirements:

🔗 Attributes (choose access level wisely):

- o title (public)
- o author (public)
- o isbn (private)
- o is_borrowed (protected, default False)
- o borrowed_by (protected, default None)

main.py

Share

Run

```
1 class LibraryBook:
2
3     def __init__(self, title, author, isbn):
4         self.title = title
5         self.author = author
6
7         self.__isbn = isbn
8
9         self._is_borrowed = False
10        self._borrowed_by = None
11
12    def borrow_book(self, student_name):
13        if not self._is_borrowed:
14            self._is_borrowed = True
15            self._borrowed_by = student_name
16            print(f"{self.title} has been borrowed by {student_name}.")
17        else:
18            print("Book is already borrowed.")
19
20    def return_book(self):
21        if self._is_borrowed:
22            print(f"{self.title} has been returned.")
23            self._is_borrowed = False
24            self._borrowed_by = None
25        else:
26            print("Book was not borrowed.")
27    def get_isbn(self):
28        return self.__isbn
29
30 book1 = LibraryBook("Epstein Files", "Jeffery Epstein", "123-456-789")
31
32 print("Title:", book1.title)
33 print("Author:", book1.author)
34 print("ISBN:", book1.get_isbn())
35
36 book1.borrow_book("Rasikh Ali")
```

Output

Title: Epstein Files
Author: Jeffery Epstein
ISBN: 123-456-789
Epstein Files has been borrowed by Rasikh Ali.
Epstein Files has been returned.

=== Code Execution Successful ===