

Lab-6.3

T.Rakshitha

2303a51172

Batch:18

Task Description #1: Classes (Student Class)

Scenario

You are developing a simple student information management module.

Task

- Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.**
- The class should include attributes such as name, roll number, and branch.**
- Add a method `display_details()` to print student information.**
- Execute the code and verify the output.**
- Analyze the code generated by the AI tool for correctness and clarity.**

Expected Output #1

- A Python class with a constructor (`__init__`) and a `display_details()` method.**
- Sample object creation and output displayed on the console.**
- Brief analysis of AI-generated code.**

Prompts:

#Generate a Python class named Student with attributes name, roll_number, and branch.

#Include a constructor to initialize these values and a method `display_details()` to print the student information.

#Also create a sample object and display its details.

Sample object creation

Display student details

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows several Python files including "AI ASSISTANT", "# Sales Data Processing Application.py", "#Generate a Python class named Student with attributes.py", "#Generate a Python class User that valid age...", "#read a list of numbers.py", "#read a number and check if it is prime co...", "#read a number and check if it is prime co...", "#read a.txt file and count number of lines in...", "#read input from user to check if it is even or odd...", "#take input from user and check even or odd...", "#take input from user and check if it is a prime number...", "#take input from user and check if it is arms...", "#take input from user and check if it is leap year...", "#take input from user and convert centimetres to feet...", "#take input from user and print factorial of a number...", "#take input from user as string and print words...", "1 to N using a loop.py", "armstrong.py", "calculate total and avg marks.py", "even and odd.py", "lab.txt", and "using class, loops, and conditional stat...".
- Code Editor:** Displays the following Python code for a `Student` class:

```

#Generate a Python class named Student with attributes.py > ...
1 #Generate a Python class named Student with attributes name, roll_number, and branch.
2 #Include a constructor to initialize these values and a method display_details() to print the student information.
3 #Also create a sample object and display its details.
4 class Student:
5     def __init__(self, name, roll_number, branch):
6         self.name = name
7         self.roll_number = roll_number
8         self.branch = branch
9
10    def display_details(self):
11        print("Name:", self.name)
12        print("Roll Number:", self.roll_number)
13        print("Branch:", self.branch)
14
15    # Sample object creation
16    student1 = Student("Rakshitha", 101, "Computer Science")
17
18    # Display student details
19    student1.display_details()
20
21
22

```
- Terminal:** Shows the command-line output of running the script:

```

PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT>
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> & c:/Users/raksh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/raksh/OneDrive/Desktop/AI ASSISTANT/#Generate a Python class named Student with attributes.py"
Name: Rakshitha
Roll Number: 101
Branch: Computer Science
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT>

```
- Bottom Status Bar:** Shows the current file is "lab.txt", the line and column are Ln 22, Col 1, the encoding is UTF-8, the file type is Python, the version is 3.13.2, the locale is ENG_IN, the date and time are 04-02-2026 11:22, and the Go Live button.

Analysis:

display_details() method correctly prints student information.

Code is simple, readable, and well-structured.

Task Description #2: Loops (Multiples of a Number)

Scenario

You are writing a utility function to display multiples of a given number.

Task

- Prompt the AI tool to generate a function that prints the first 10 multiples of a given number using a loop.
- Analyze the generated loop logic.
- Ask the AI to generate the same functionality using another controlled looping structure (e.g., while instead of for).

Expected Output #2

- Correct loop-based Python implementation.
- Output showing the first 10 multiples of a number.
- Comparison and analysis of different looping approaches.

Prompts:

#Generate a Python function that prints the first 10 multiples of a given number using a for loop.

Sample execution

The screenshot shows the Visual Studio Code interface with the AI ASSISTANT extension open. In the Explorer sidebar, there are several Python files listed under 'AI ASSISTANT'. One file, 'first 10 multiples.py', is selected and contains the following code:

```
1 #Generate a Python function that prints the first 10 multiples of a given number using a for loop.
2 def print_multiples_for(n):
3     for i in range(1, n+1):
4         print(n * i)
5
6
7 # Sample execution
8 print("Multiples using for loop:")
9 print_multiples_for(5)
```

In the terminal at the bottom, the command 'python first 10 multiples.py' is run, and the output is displayed:

```
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> python first 10 multiples.py
Multiples using for loop:
5
10
15
20
25
30
35
40
45
50
```

Analysis:

Uses for and while loops to generate multiples.

for loop is concise and suitable for fixed iterations.

while loop offers more control but requires manual counter handling.

Task Description #3: Conditional Statements (Age Classification)

Scenario

You are building a basic classification system based on age.

Task

- Ask the AI tool to generate nested if-elif-else conditional statements to classify age groups
(e.g., child, teenager, adult, senior).
- Analyze the generated conditions and logic.
- Ask the AI to generate the same classification using alternative conditional structures
(e.g.,

simplified conditions or dictionary-based logic).

Expected Output #3

- A Python function that classifies age into appropriate groups.
- Clear and correct conditional logic.
- Explanation of how the conditions work.

Prompts:

#Generate a Python function that classifies a person's age into categories:

#child, teenager, adult, and senior using nested if-elif-else statements.

Sample execution

The screenshot shows the Visual Studio Code interface with the AI Assistant extension open. The Explorer sidebar on the left lists various Python files and their descriptions. The main editor area contains a Python script named 'persons age into categories.py' with code for a 'classify_age' function using nested if-elif-else statements. Below the code, a '# Sample execution' section shows an example where age is set to 25 and the output is 'Age Group: Adult'. The bottom status bar shows the file path 'C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT\persons age into categories.py' and the Python version 'Python 3.13.2'. The bottom right corner shows the date and time '04-02-2026'.

```
# Generate a Python function that classifies a person's age into categories:  
#child, teenager, adult, and senior using nested if-elif-else statements.  
  
def classify_age(age):  
    if age < 13:  
        return "Child"  
    elif age < 20:  
        return "Teenager"  
    elif age < 60:  
        return "Adult"  
    else:  
        return "Senior"  
  
# Sample execution  
age = 25  
print("Age Group:", classify_age(age))
```

Analysis:

Uses if-elif-else to implement decision-making logic.

Conditions are ordered correctly to avoid overlap.

Task Description #4: For and While Loops (Sum of First n Numbers)

Scenario

You need to calculate the sum of the first n natural numbers.

Task

- Use AI assistance to generate a `sum_to_n()` function using a for loop.

- Analyze the generated code.
- Ask the AI to suggest an alternative implementation using a while loop or a mathematical formula.

Expected Output #4

- Python function to compute the sum of first n numbers.
- Correct output for sample inputs.
- Explanation and comparison of different approaches.

Prompts:

#Generate a Python function named sum_to_n() that calculates the sum of the first n natural numbers using a for loop.

Sample execution

```

File Edit Selection View Go Run Terminal Help < > Q AI ASSISTANT
EXPLORER AI ASSISTANT
# Sales Data Processing Application.py
# count vowels from a sentence.py
#Generate a Python class named Student w...
#Generate a Python class User that valid ag...
#read a list of numbers.py
#read a number and check if it is prime co...
#read a number and check if it is prime co...
#read a.txt file and count number of lines i...
#read input from user number to check if it ...
#read tuple from user and print sum of even...
#take input from user and check even or odd...
#take input from user and check if it is a pe...
#take input from user and check if it is arms...
#take input from user and check if it is leap ...
#take input from user and convert continu...
#take input from user and print factorial of ...
#take input from user as string and print w...
1 To N using a loop.py
armstrong.py
calculate total and avg marks.py
even and odd.py
first 10 multiples.py
first n natural numbers using a loop.py
lab.txt
persons age into categories.py
using class, loops, and conditional stat... 1
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> 
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> & C:/Users/raksh/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/raksh/OneDrive/Desktop/AI ASSISTANT/first n natural numbers using a loop.py"
sum using For Loop: 55
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> []

```

Analysis:

Demonstrates iterative and mathematical problem-solving approaches.

Task Description #5: Classes (Bank Account Class)

Scenario

You are designing a basic banking application.

Task

- Use AI tools to generate a Bank Account class with methods such as deposit(), withdraw(),

and check_balance().

- Analyze the AI-generated class structure and logic.
- Add meaningful comments and explain the working of the code.

Expected Output #5

- Complete Python Bank Account class.
- Demonstration of deposit and withdrawal operations with updated balance.
- Well-commented code with a clear explanation.

Prompts:

```
#Generate a Python class named BankAccount with attributes for account_holder and
balance. Include methods deposit(), withdraw(), and check_balance().
```

```
#Demonstrate the class with sample deposit and withdrawal operations and add
meaningful comments.
```

```
# Constructor to initialize account holder name and balance
```

```
# Method to deposit money into the account
```

```
# Method to withdraw money from the account
```

```
# Method to check current account balance
```

```
# Sample execution
```

```

File Edit Selection View Go Run Terminal Help ⏎ → Q AI ASSISTANT
File Edit Selection View Go Run Terminal Help ⏎ → Q AI ASSISTANT
EXPLORER ... Sales Data Processing Application.py AI ASSISTANT
AI ASSISTANT
# Generate a Python class named BankAccount with attributes for account_holder and
balance. Include methods deposit(), withdraw(), and check_balance().
1 #Demonstrate a Python class named BankAccount with attributes for account_holder and balance.
2 #Demonstrate the class with sample deposit and withdrawal operations and add meaningful comments.
3
4 class BankAccount:
5     # Constructor to initialize account holder name and balance
6     def __init__(self, account_holder, balance=0):
7         self.account_holder = account_holder
8         self.balance = balance
9
10    # Method to deposit money into the account
11    def deposit(self, amount):
12        if amount > 0:
13            self.balance += amount
14            print("Deposited:", amount)
15        else:
16            print("Invalid deposit amount")
17
18    # Method to withdraw money from the account
19    def withdraw(self, amount):
20        if amount > self.balance:
21            print("Insufficient balance")
22        elif amount <= 0:
23            print("Invalid withdrawal amount")
24        else:
25            self.balance -= amount
26            print("Withdrawn:", amount)
27
28    # Method to check current account balance
29    def check_balance(self):
30        print("Current Balance:", self.balance)
31
32 # Sample execution
33 account = BankAccount("Rakshitha", 1000)
34 account.check_balance()
35 account.deposit(500)
36 account.withdraw(300)
37 account.check_balance()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> python
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT> & C:/Users/raksh/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/raksh/Desktop/AI ASSISTANT/banking_application.py"
Current Balance: 1000
Deposited: 500
Withdrawn: 300
Current Balance: 1200
PS C:\Users\raksh\OneDrive\Desktop\AI ASSISTANT>

OUTLINE TIMELINE master* Launchpad 1 △ 0
Hot days ahead 24°C

```

Analysis:

Code structure is clear and easy to extend.