**AI-ASSISTANT-COADING**

**Lab6.3**
**2303A51247**
**Batch-18**

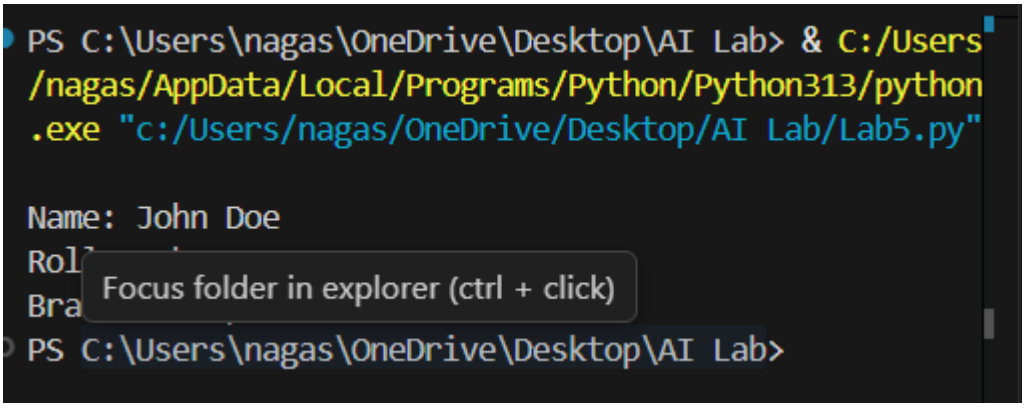
Description #1: Classes (Student Class)
Scenario
You are developing a simple student information management module.
Task
• Use an AI tool (GitHub Copilot / Cursor AI / Gemini) to complete a Student class.
• The class should include attributes such as name, roll number, and branch.
• Add a method display_details() 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.
Prompt:

```
#create a class so that the class should include attributes such as
name, roll number,branch, add a method display_details() to print
student info
```

Code:

```
class Student:
    def __init__(self, name, roll_number, branch):
        self.name = name
        self.roll_number = roll_number
        self.branch = branch
    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Branch: {self.branch}")

# Create an instance of the Student class
student1 = Student("John Doe", "12345", "Computer Science")

# Call the display_details method
student1.display_details()
```

Output:




Analysis:
The AI-generated Student class correctly implements object-oriented principles using a constructor and a method, resulting in clean, readable, and functional code.

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).
Prompt:

```
#print the first 10 multiples of a given number using a loop and
functions
```

Code:

```
# Function to print the first 10 multiples of a given number using a
for loop
def print_multiples_for_loop(number):
    print(f"First 10 multiples of {number} using for loop:")
    for i in range(1, 11):
        multiple = number * i
        print(f"{number} x {i} = {multiple}")

# Function to print the first 10 multiples of a given number using a
while loop
def print_multiples_while_loop(number):
    print(f"First 10 multiples of {number} using while loop:")
    i = 1
    while i <= 10:
        multiple = number * i
        print(f"{number} x {i} = {multiple}")
```

```
        i += 1

# Example usage
num = 5
print_multiples_for_loop(num)
print()
print_multiples_while_loop(num)
```

Output:

```
PS C:\Users\nagas\OneDrive\Desktop\AI Lab> & C:/Users
/nagas/AppData/Local/Programs/Python/Python313/python
.exe "c:/Users/nagas/OneDrive/Desktop/AI Lab/Lab5.py"

First 10 multiples of 5 using for loop:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

First 10 multiples of 5 using while loop:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
PS C:\Users\nagas\OneDrive\Desktop\AI Lab>
```

Analysis:

The AI-generated loop logic efficiently prints the first ten multiples of a number, with both for and while loop implementations producing correct and readable output.

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.

Prompt:

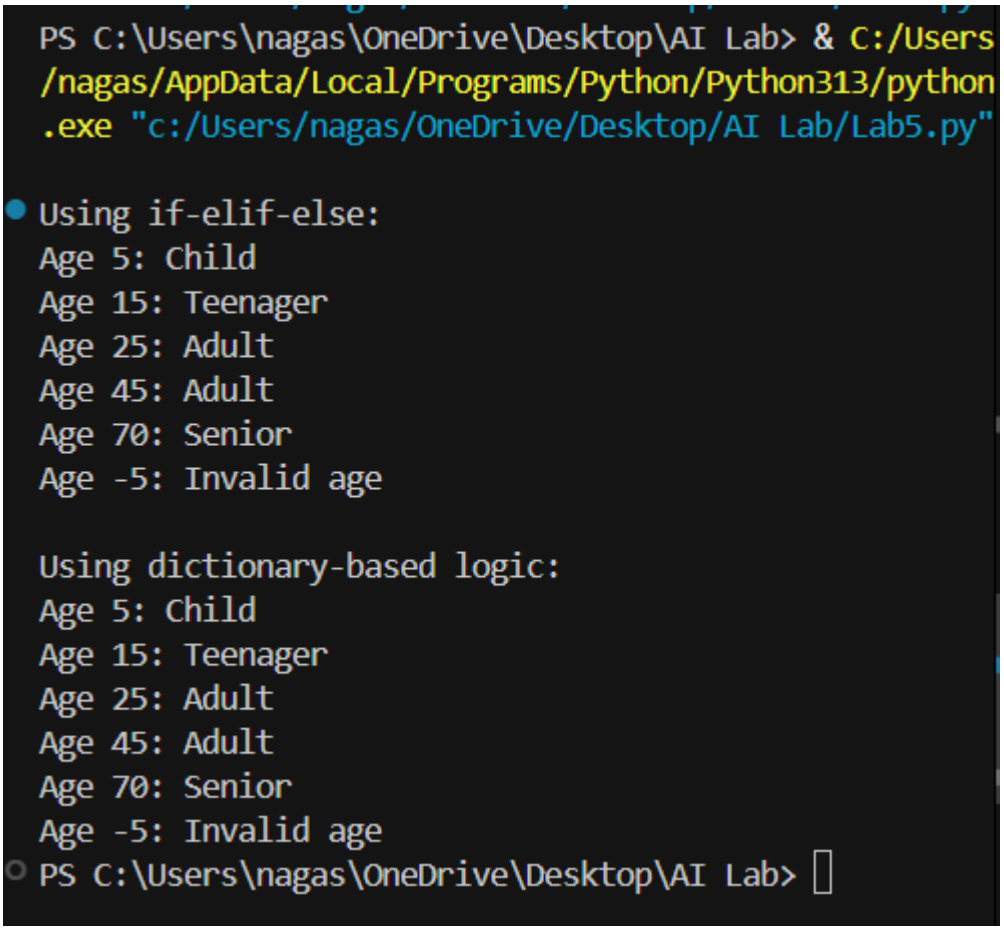
```
##generate nested if-else statements to classify the age gropus
(like:child,teenager,adult,senior)
```

Code:

```
# Function to classify age using nested if-elif-else statements
def classify_age_if_elif(age):
    if age < 0:
        return "Invalid age"
    elif age <= 12:
        return "Child"
    elif age <= 19:
        return "Teenager"
    elif age <= 64:
        return "Adult"
    else:
        return "Senior"

# Function to classify age using dictionary-based logic
def classify_age_dictionary(age):
    if age < 0:
        return "Invalid age"

    age_groups = {
        (0, 12): "Child",
        (13, 19): "Teenager",
        (20, 64): "Adult",
        (65, float('inf')): "Senior"
    }
```

```
    for (min_age, max_age), group in age_groups.items():
        if min_age <= age <= max_age:
            return group

    return "Senior"  # For ages above 65

# Example usage
ages = [5, 15, 25, 45, 70, -5]

print("Using if-elif-else:")
for age in ages:
    print(f"Age {age}: {classify_age_if_elif(age)}")

print("\nUsing dictionary-based logic:")
for age in ages:
    print(f"Age {age}: {classify_age_dictionary(age)}")
```

Output:




Analysis:
The AI-generated conditional statements accurately classify age groups using clear if-elif-else logic, ensuring correctness and easy understanding.

ask 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.

Prompt:

```
#generate a sum_to_n() function usig a for loop
```

Code:

```
# Function to calculate sum of first n natural numbers using a for loop
def sum_to_n_for_loop(n):
    if n < 1:
        return 0
    total = 0
    for i in range(1, n + 1):
        total += i
    return total

# Function to calculate sum of first n natural numbers using a while
loop
def sum_to_n_while_loop(n):
    if n < 1:
        return 0
    total = 0
    i = 1
    while i <= n:
        total += i
        i += 1
    return total

# Function to calculate sum of first n natural numbers using
mathematical formula
def sum_to_n_formula(n):
    if n < 1:
        return 0
    return n * (n + 1) // 2
```

```python
# Example usage and comparison
n_values = [5, 10, 100]

for n in n_values:
    for_result = sum_to_n_for_loop(n)
    while_result = sum_to_n_while_loop(n)
    formula_result = sum_to_n_formula(n)

    print(f"Sum of first {n} natural numbers:")
    print(f"  For loop: {for_result}")
    print(f"  While loop: {while_result}")
    print(f"  Formula: {formula_result}")
    print(f"  All results match: {for_result == while_result == formula_result}")
    print()
```

Output:

```
PS C:\Users\nagas\OneDrive\Desktop\AI Lab> & C:/Users/nagas/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/nagas/OneDrive/Desktop/AI Lab/Lab5.py"

Sum of first 5 natural numbers:
  For loop: 15
  While loop: 15
  Formula: 15
  All results match: True

Sum of first 10 natural numbers:
  For loop: 55
  While loop: 55
  Formula: 55
  All results match: True

Sum of first 100 natural numbers:
  For loop: 5050
  While loop: 5050
  Formula: 5050
  All results match: True

PS C:\Users\nagas\OneDrive\Desktop\AI Lab>
```

Analysis:

The AI-generated implementations using loops and a mathematical formula correctly compute the sum of the first n natural numbers, with the formula-based approach being more efficient.

ask 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

Prompt:

```
#generate a bank account class with methods such as
deposit,withdraw,check_balance
```

Code:

```
class BankAccount:

    def __init__(self, account_holder, initial_balance=0):

        self.account_holder = account_holder
        self.balance = initial_balance

    def deposit(self, amount):

        if amount <= 0:
            print("Error: Deposit amount must be positive.")
            return False

        self.balance += amount
        print(f"Successfully deposited ${amount:.2f}. New balance:
${self.balance:.2f}")
        return True

    def withdraw(self, amount):
```

```
        if amount <= 0:
            print("Error: Withdrawal amount must be positive.")
            return False

        if amount > self.balance:
            print(f"Error: Insufficient funds. Current balance:
${self.balance:.2f}")
            return False

        self.balance -= amount
        print(f"Successfully withdrew ${amount:.2f}. New balance:
${self.balance:.2f}")
        return True

    def check_balance(self):

        print(f"Account balance for {self.account_holder}:
${self.balance:.2f}")
        return self.balance

# Demonstration of the Bank Account class
if __name__ == "__main__":
    # Create a new bank account
    account = BankAccount("John Doe", 1000.00)

    # Check initial balance
    account.check_balance()

    # Perform some operations
    account.deposit(500.00)
    account.withdraw(200.00)
    account.withdraw(1500.00)  # This should fail due to insufficient
funds
    account.deposit(-100.00)   # This should fail due to negative
amount

    # Final balance check
    account.check_balance()
```

Output:

```
PS C:\Users\nagas\OneDrive\Desktop\AI Lab> & C:/Users
/nagas/AppData/Local/Programs/Python/Python313/python
.exe "c:/Users/nagas/OneDrive/Desktop/AI Lab/Lab5.py"

Account balance for John Doe: $1000.00
Successfully deposited $500.00. New balance: $1500.00
Successfully withdrew $200.00. New balance: $1300.00
Error: Insufficient funds. Current balance: $1300.00
Error: Deposit amount must be positive.
Account balance for John Doe: $1300.00
PS C:\Users\nagas\OneDrive\Desktop\AI Lab>
```

Analysis:

The AI-generated Bank Account class effectively models real-world banking operations with well-structured methods and clear logic for balance management.