

ASSIGNMENT – 6.3

2303A51060

Batch-10

Task-1

Prompt: generate studnet information system using a class with methods to name, branch,rollnumber and display student details with user input.

Code:

```
class Student:  
  
    def __init__(self, name, branch, roll_number):  
        self.name = name  
        self.branch = branch  
        self.roll_number = roll_number  
  
    def display_details(self):  
        print(f"Student Name: {self.name}")  
        print(f"Branch: {self.branch}")  
        print(f"Roll Number: {self.roll_number}")  
  
if __name__ == "__main__":  
    name = input("Enter student name: ")  
    branch = input("Enter branch: ")  
    roll_number = input("Enter roll number: ")  
    student = Student(name, branch, roll_number)  
    student.display_details()
```

Output :

```

1 #task-1
2 #generate student information system using class with methods to name,branch,rollnumber and display student details with user input.
3 class Student:
4     def __init__(self, name, branch, roll_number):
5         self.name = name
6         self.branch = branch
7         self.roll_number = roll_number
8
9     def display_details(self):
10        print("Student Name: " + self.name)
11        print("Branch: " + self.branch)
12        print("Roll Number: " + str(self.roll_number))
13
14 if __name__ == "__main__":
15     name = input("Enter student name: ")
16     branch = input("Enter branch: ")
17     roll_number = input("Enter roll number: ")
18     student = Student(name, branch, roll_number)
19     student.display_details()

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS AZURE

python3.13

```

PS C:\Users\NIRWANA\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRWANA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRWANA/Desktop/AI_A
SSISTENT_CODING/lab-6.3.py
Enter student name: Nirwana
Enter branch: cse
Enter roll number: 2303a51060
Student Name: Nirwana
Branch: cse
Roll Number: 2303a51060

```

Code Analysis :

- A `Student` class is created to represent student data.
- The constructor (`__init__`) initializes name, branch, and roll number.
- Instance variables store individual student details.
- `display_details()` method prints student information neatly.
- User input is used to create an object dynamically at runtime.

Task-2

Prompt: Generate a utility function that prints first 10 multiples of a given number using a loop with user input.

Code:

```

def print_multiples(number, count=10):

    for i in range(1, count + 1):

        print(f"{number} x {i} = {number * i}")

if __name__ == "__main__":

    num = int(input("Enter a number to print its first 10 multiples:"))

    print_multiples(num)

```

Output :

The screenshot shows a Microsoft Visual Studio Code interface. The left sidebar has icons for Explorer, Search, Open, Find, and others. The Explorer panel shows files: lab-3.3.py, lab-3.4.py, lab-6.3.py (which is the active tab), lab-4.3.py, lab-5.4.py, lab1.py, and lab2.py. The main editor area contains the following Python code:

```
19 #~~~~~
20 #task-2
21 #Generate a utility function that prints first 10 multiples of given number using loop with user input.
22 def print_multiples(number, count=10):
23     for i in range(1, count + 1):
24         print(f'{number} x {i} = {number * i}')
25 if __name__ == "__main__":
26     num = int(input("Enter a number to print its first 10 multiples: "))
27     print_multiples(num)
28 #~~~~~
```

Below the editor, tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, QUERY RESULTS, and AZURE are visible. The TERMINAL tab is selected, showing command-line output:

```
PS C:\Users\WIRNAYA\OneDrive\Desktop\AI_ASSISTANT_CODING> & C:\Users\WIRNAYA\AppData\Local\Microsoft\Windows\apps\python3.13.exe c:/Users/WIRNAYA/OneDrive/Desktop/AI_ASSISTANT_CODING/lab-6.3.py
Enter a number to print its first 10 multiples: 5
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
```

Code Analysis :

- The function `print_multiples()` accepts a number and a count as parameters.
 - A `for` loop iterates from 1 to 10 to generate multiples.
 - Multiplication logic is handled inside the loop.
 - The default parameter value ensures flexibility.
 - Function improves reusability for different inputs.

Task-3

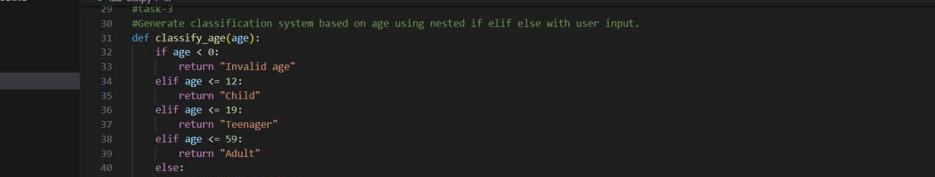
Prompt: Generate a classification system based on age using nested if, elif, and else with user input.

code:

```
def classify_age(age):  
    if age < 0:  
        return "Invalid age"  
    elif age <= 12:  
        return "Child"  
    elif age <= 19:  
        return "Teenager"  
    elif age <= 59:  
        return "Adult"  
    else:
```

```
return "Senior Citizen"  
  
if __name__ == "__main__":  
    age = int(input("Enter your age: "))  
    category = classify_age(age)  
    print(f"You are classified as: {category}")
```

Output :



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The title bar reads "Q AI_ASSISTANT_CODING". The left sidebar has sections for "EXPLORER", "ALASSISTANT_CODING", and "PROBLEMS". The main editor area contains Python code for classifying people based on age. The terminal at the bottom shows the command "python lab-6.3.py" being run and the output "You are classified as: Adult".

```
File Edit Selection View Go Run Terminal Help < > Q AI_ASSISTANT_CODING 08 EXPLORER ... ALASSISTANT_CODING lab-3.3.py lab-3.4.py lab-6.3.py x lab-5.4.py lab-4.3.py lab-6.3.py > ... 29 #Task-3 30 #Generate classification system based on age using nested if elif else with user input. 31 def classify_age(age): 32     if age < 0: 33         return "Invalid age" 34     elif age <= 12: 35         return "Child" 36     elif age <= 19: 37         return "Teenager" 38     elif age <= 59: 39         return "Adult" 40     else: 41         return "Senior Citizen" 42 if __name__ == "__main__": 43     age = int(input("Enter your age: ")) 44     category = classify_age(age) 45     print(f"You are classified as: {category}") 46 #
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS AZURE

```
PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTANT_CODING & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_ASSISTANT_CODING/lab-6.3.py  
Enter your age: 20  
You are classified as: Adult
```

Code Analysis :

- the function `classify_age()` categorizes age groups.
 - Conditional statements check age ranges sequentially.
 - Invalid inputs are handled using boundary checks.
 - Function returns a classification string.
 - User input is processed dynamically for classification.

Task-4

Prompt: Generate the sum of the first n numbers using for and while loops with user input.

Code:

```
def sum_of_numbers(n):
```

total = 0

```
for i in range(1, n + 1):
```

```
total += i
```

return total

```

if __name__ == "__main__":
    n = int(input("Enter a positive integer to calculate the sum of first n numbers: "))
    result = sum_of_numbers(n)
    print(f"The sum of the first {n} numbers is: {result}")

```

Output :

The screenshot shows a code editor interface with the following details:

- Explorer Panel:** Shows several Python files: lab-3.3.py, lab-3.4.py (which is currently selected), lab-4.3.py, lab-4.4.py, and lab1.py.
- Terminal Window:**
 - Shows the command: PS C:\Users\NIRWANA\OneDrive\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRWANA\AppData\Local\Microsoft\WindowsApps\python3.10.exe c:/users/NIRWANA/OneDrive/Desktop/AI_A
 - Shows the output of the script execution:
 - SSISTENT CODING/lab-3.4.py
 - Enter an email address to validate: chinm@gmail.com
 - The email address 'chinm@gmail.com' is valid.
 - Enter a password to check its strength: chinm
 - The password is weak.
 - Enter a number to calculate the sum of its digits: 6
 - The sum of the digits of 6 is: 6

Code Analysis :

- The function `sum_of_numbers()` computes sum iteratively.
- A `for` loop runs from 1 to `n` for accumulation.
- Variable `total` stores cumulative sum.
- Function returns computed result instead of printing directly.
- Clean separation of logic and input/output handling.

Task 5

Prompt: Generate a bank application using a class with methods deposit, withdraw and display balance with user input.

Code:

```

class BankAccount:

    def __init__(self, initial_balance=0):
        self.balance = initial_balance

```

```
def deposit(self, amount):
    self.balance += amount
    print(f"Deposited: ${amount}")

def withdraw(self, amount):
    if self.balance >= amount:
        self.balance -= amount
        print(f"Withdrew: ${amount}")
    else:
        print("Insufficient funds")

def display_balance(self):
    print(f"Current Balance: ${self.balance}")

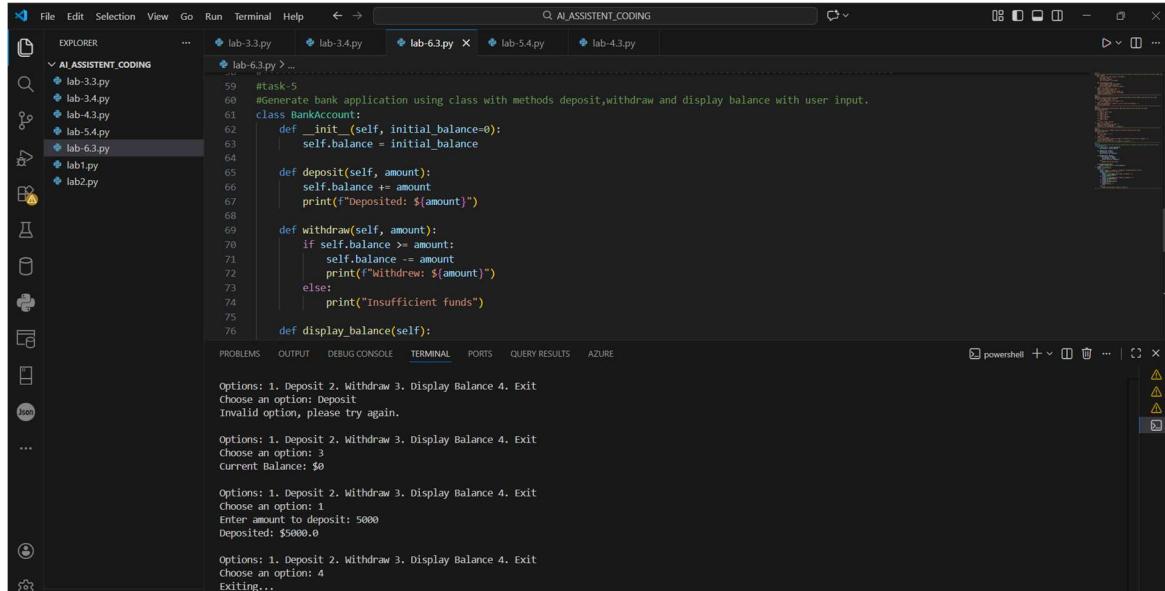
if __name__ == "__main__":
    account = BankAccount()
    while True:
        print("\nOptions: 1. Deposit 2. Withdraw 3. Display Balance 4. Exit")
        choice = input("Choose an option: ")
        if choice == '1':
            amount = float(input("Enter amount to deposit: "))
            account.deposit(amount)
        elif choice == '2':
            amount = float(input("Enter amount to withdraw: "))
            account.withdraw(amount)
        elif choice == '3':
            account.display_balance()
        elif choice == '4':
            print("Exiting...")
            break
```

```

else:
    print("Invalid option, please try again.")

```

Output :



```

File Edit Selection View Go Run Terminal Help ← → C ALASSISTENT_CODING 08:00
EXPLORER ... lab-3.3.py lab-3.4.py lab-6.3.py x lab-5.4.py lab-4.3.py
AI ASSISTENT CODING
lab-3.3.py
lab-3.4.py
lab-4.3.py
lab-5.4.py
lab-6.3.py
lab1.py
lab2.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS AZURE
powershell + 🗑️ 🗑️ ...
options: 1. Deposit 2. Withdraw 3. Display Balance 4. Exit
Choose an option: Deposit
Invalid option, please try again.

options: 1. Deposit 2. Withdraw 3. Display Balance 4. Exit
Choose an option: 3
Current Balance: $0

options: 1. Deposit 2. Withdraw 3. Display Balance 4. Exit
Choose an option: 1
Enter amount to deposit: 5000
Deposited: $5000.0

options: 1. Deposit 2. Withdraw 3. Display Balance 4. Exit
Choose an option: 4
Exiting...

```

Code Analysis :

- BankAccount class models a real-world bank account.
- Constructor initialises account balance.
- deposit() and withdraw() methods modify balance safely.
- Conditional checks prevent overdraft.
- A menu-driven loop allows continuous user interaction.