

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: ")

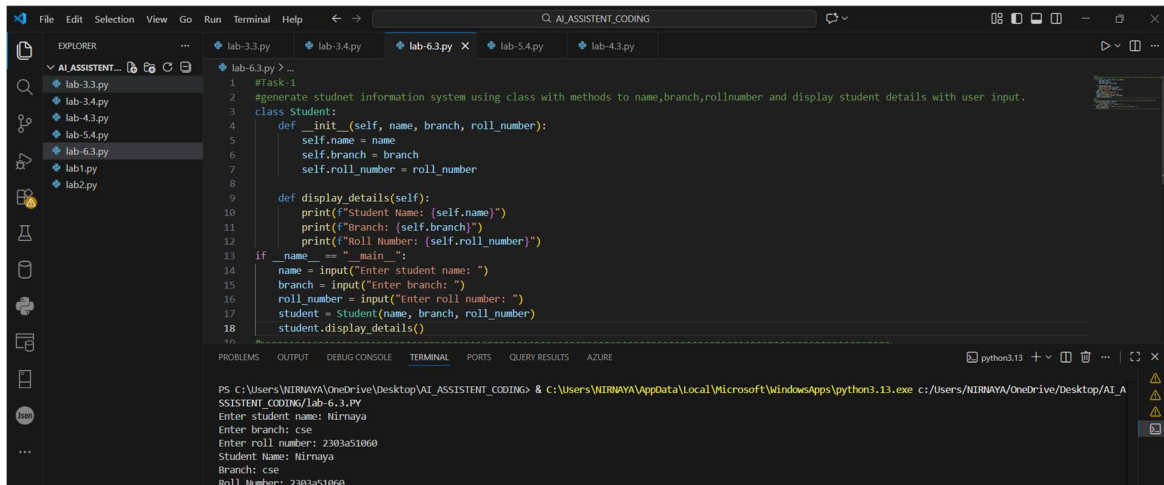
    branch = input("Enter branch: ")

    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(f"Student Name: {self.name}")
11         print(f"Branch: {self.branch}")
12         print(f"Roll Number: {self.roll_number}")
13 if __name__ == "__main__":
14     name = input("Enter student name: ")
15     branch = input("Enter branch: ")
16     roll_number = input("Enter roll number: ")
17     student = Student(name, branch, roll_number)
18     student.display_details()
```

PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & c:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.11.exe c:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING\lab-6.3.py

Enter student name: Nirnaya

Enter branch: cse

Enter roll number: 2303a51060

Student Name: Nirnaya

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

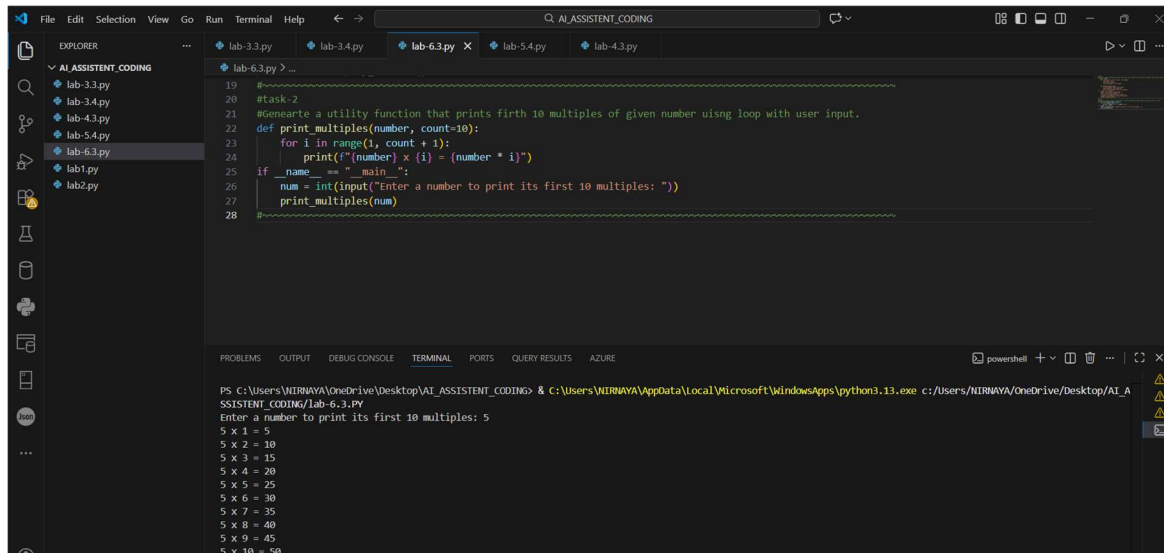
Prompt: Generate a utility function that prints first 10 multiples of a given number using a loop with user input.

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 Visual Studio Code editor window with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'AI_ASSISTENT_CODING' with several Python files. The active file is 'lab-6.3.py', which contains a Python script. The script defines a function 'print_multiples' that takes a number and a count as parameters. It uses a for loop to calculate and print the first 10 multiples of the given number. The terminal shows the command to run the script and the resulting output, which lists the first 10 multiples of 5.

```
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 #-----
```

```
PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_ASSISTENT_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 Visual Studio Code editor with a file explorer on the left containing several Python files. The main editor window displays a Python script named 'lab-6.3.py'. The script defines a function 'classify_age' that uses nested if-elif-else statements to categorize a person's age into 'Invalid age', 'Child', 'Teenager', 'Adult', or 'Senior Citizen'. Below the function, a main block uses 'if __name__ == "__main__":' to take user input, call the 'classify_age' function, and print the result. The terminal at the bottom shows the command to run the script and the output: 'Enter your age: 20' followed by 'You are classified as: Adult'.

```

#task-3
#Generate classification system based on age using nested if elif else with user input.
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}")

```

PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & c:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_ASSISTENT_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

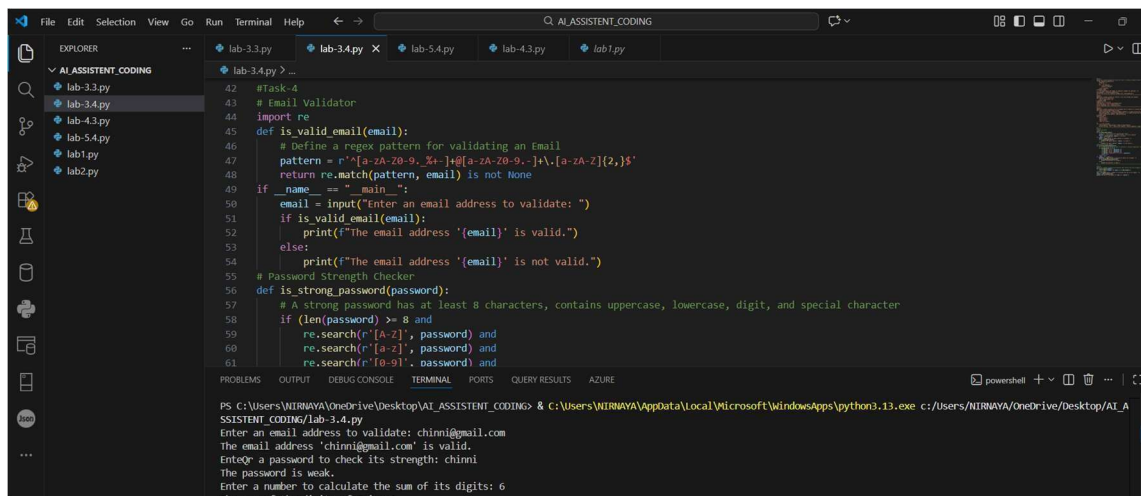
```

```

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 :



```

42 #Task-4
43 # Email Validator
44 import re
45 def is_valid_email(email):
46     # Define a regex pattern for validating an Email
47     pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
48     return re.match(pattern, email) is not None
49 if __name__ == "__main__":
50     email = input("Enter an email address to validate: ")
51     if is_valid_email(email):
52         print(f"The email address '{email}' is valid.")
53     else:
54         print(f"The email address '{email}' is not valid.")
55 # Password Strength Checker
56 def is_strong_password(password):
57     # A strong password has at least 8 characters, contains uppercase, lowercase, digit, and special character
58     if (len(password) >= 8 and
59         re.search(r'[A-Z]', password) and
60         re.search(r'[a-z]', password) and
61         re.search(r'[0-9]', password) and

```

PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & c:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_ASSISTENT_CODING/Lab-3.4.py
Enter an email address to validate: chinmi@gmail.com
The email address 'chinmi@gmail.com' is valid.
Enter a password to check its strength: chinmi
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}')

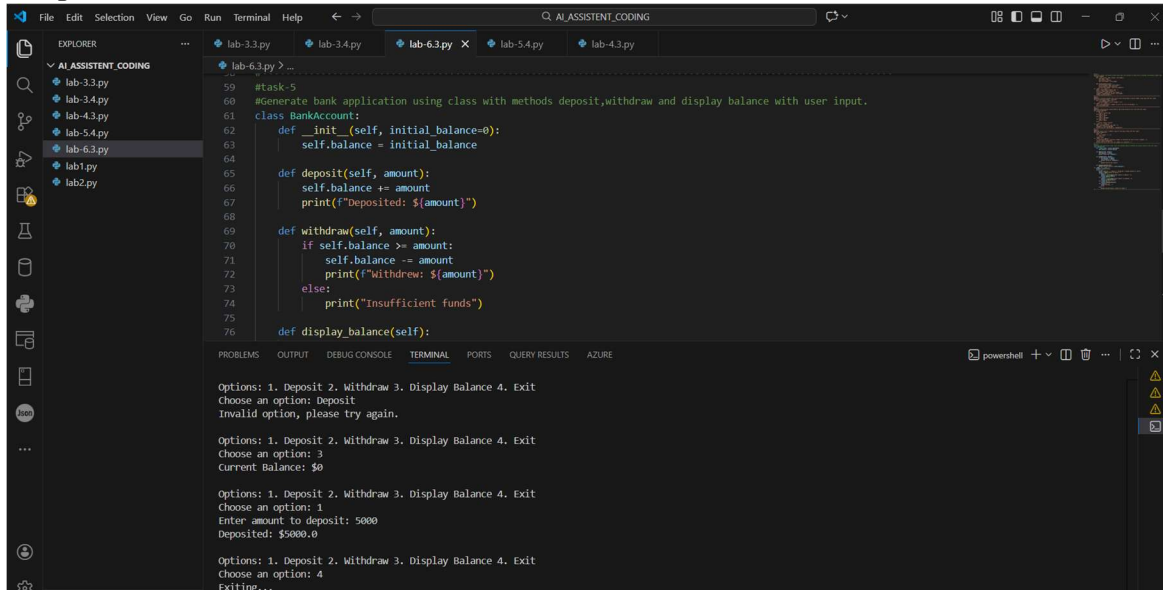
if __name__ == "__main__":
    account = BankAccount()
    while True:
        print("\nOptions: 1. Deposit 2. Withdraw 3. Display Balance 4. Exit")
        choice = input("Choose an option: ")
        if choice == '1':
            amount = float(input("Enter amount to deposit: "))
            account.deposit(amount)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



The screenshot shows a Visual Studio Code editor window with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'AI_ASSISTENT_CODING' with several files: lab-3.3.py, lab-3.4.py, lab-4.3.py, lab-5.4.py, lab-6.3.py, lab1.py, and lab2.py. The file 'lab-6.3.py' is selected and open in the editor. The code in the editor is a Python script for a bank application. It defines a 'BankAccount' class with methods for deposit, withdraw, and display balance. The terminal shows the output of the script, which is a menu-driven loop where the user can choose to deposit, withdraw, display balance, or exit. The output shows the user choosing to deposit 5000, which is successful, and then choosing to exit.

```
59 #task-5
60 #Generate bank application using class with methods deposit,withdraw and display balance with user input.
61 class BankAccount:
62     def __init__(self, initial_balance=0):
63         self.balance = initial_balance
64
65     def deposit(self, amount):
66         self.balance += amount
67         print(f"Deposited: ${amount}")
68
69     def withdraw(self, amount):
70         if self.balance >= amount:
71             self.balance -= amount
72             print(f"Withdraw: ${amount}")
73         else:
74             print("Insufficient funds")
75
76     def display_balance(self):
```

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.