

## ASSIGNMENT – 6.3

2303A51355

Batch-10

Task-1

Prompt: generate a code to develop a student information system using class and display the student details like name, roll number and branch with user input.

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"Student Name: {self.name}")
```

```
        print(f"Roll Number: {self.roll_number}")
```

```
        print(f"Branch: {self.branch}")
```

```
def main():
```

```
    name = input("Enter student name: ")
```

```
    roll_number = input("Enter roll number: ")
```

```
    branch = input("Enter branch: ")
```

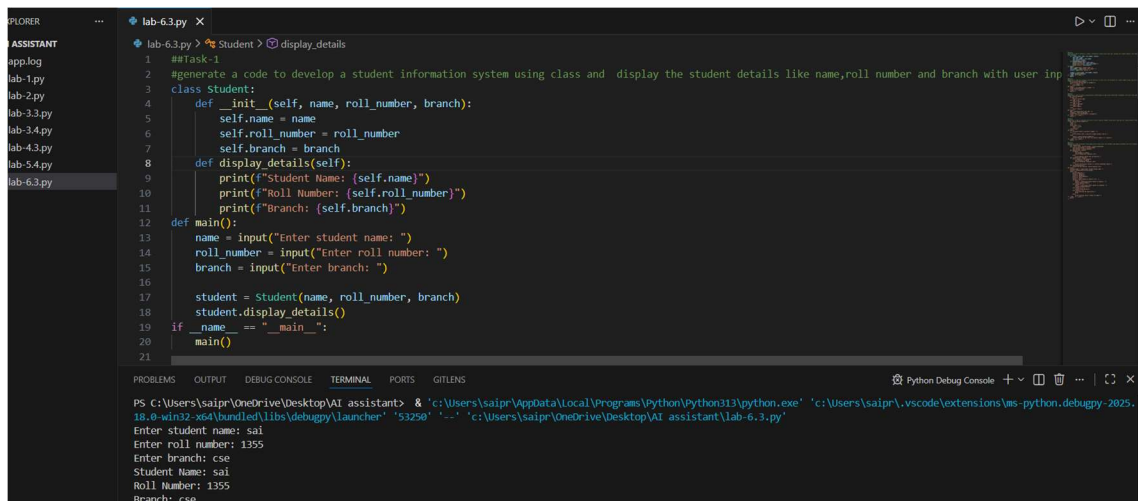
```
    student = Student(name, roll_number,
```

```
branch)
```

```
    student.display_details()
```

```
if __name__ == "__main__":
```

```
    main()
```



```
1 ##Task-1
2 #generate a code to develop a student information system using class and display the student details like name,roll number and branch with user input
3 class Student:
4     def __init__(self, name, roll_number, branch):
5         self.name = name
6         self.roll_number = roll_number
7         self.branch = branch
8     def display_details(self):
9         print(f"Student Name: {self.name}")
10        print(f"Roll Number: {self.roll_number}")
11        print(f"Branch: {self.branch}")
12 def main():
13     name = input("Enter student name: ")
14     roll_number = input("Enter roll number: ")
15     branch = input("Enter branch: ")
16
17     student = Student(name, roll_number, branch)
18     student.display_details()
19 if __name__ == "__main__":
20     main()
21
```

PS C:\Users\sai\OneDrive\Desktop\AI assistant> & 'c:\Users\sai\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\sai\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\lib\debugpy\launcher' '53250' '--' 'c:\Users\sai\OneDrive\Desktop\AI assistant\lab-6.3.py'

Enter student name: sai  
Enter roll number: 1355  
Enter branch: cse  
Student Name: sai  
Roll Number: 1355  
Branch: cse

## Code Analysis:

- ☐ A **class Student** is created to store student details like name, roll\_number, and branch.
- ☐ The `__init__` constructor initializes these values when an object is created.
- ☐ The `display_details()` method prints the stored student information.
- ☐ User input is taken inside the `main()` function.
- ☐ An object of Student class is created and used to display the details.

## Task-2

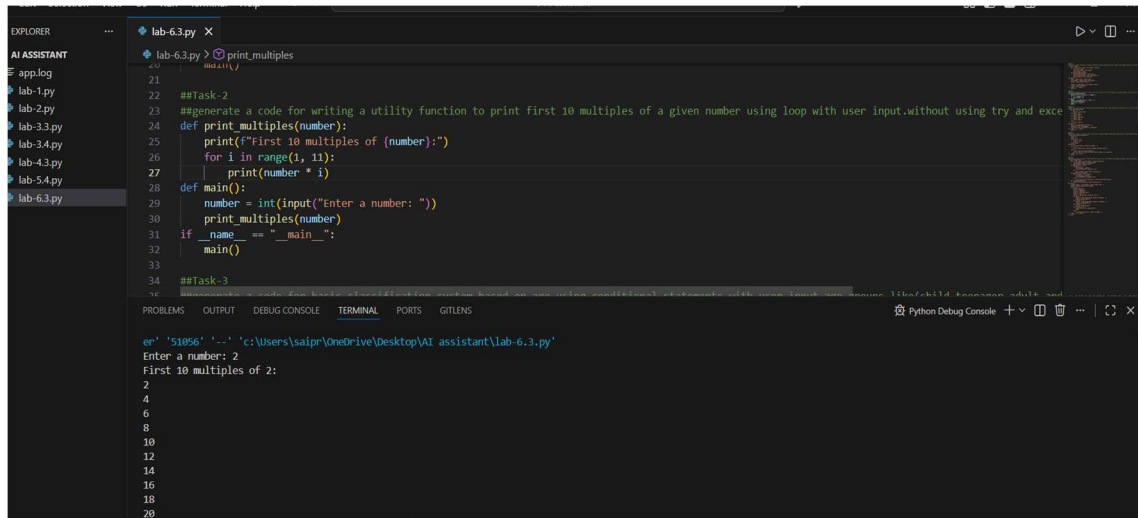
Prompt: generate a code for writing a utility function to print first 10 multiples of a given number using loops and with user input.

Code :

```
def print_multiples(number):
    print(f"First 10 multiples of {number}:")
    for i in range(1, 11):
        print(number * i)

def main():
    number = int(input("Enter a number: "))
    print_multiples(number)

if __name__ == "__main__":
    main()
```



### Code Analysis:

- ❑ A function `print_multiples()` is defined that accepts a number as a parameter.
- ❑ A **for loop** runs from 1 to 10 to calculate multiples.
- ❑ Each multiple is printed by multiplying the input number with the loop variable.
- ❑ The number is taken from the user using `input()` and converted to `int`.

### Task-3

**Prompt:** generate a code for basic classification system based on age using conditional statements  
take user input age groups like(child ,teenager , adult and senior)

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"

def main():

    age = int(input("Enter your age: "))

    category = classify_age(age)

    print(f"You are classified as: {category}")

if __name__ == "__main__":

    main()

```

The screenshot shows a VS Code editor with a file named 'lab-6.3.py'. The code is as follows:

```

33
34 #Task-3
35 #generate a code for basic classification system based on age using conditional statements with user input,age groups like(child,teenager,adult and
36 def classify_age(age):
37     if age < 0:
38         return "Invalid age"
39     elif age <= 12:
40         return "Child"
41     elif age <= 19:
42         return "Teenager"
43     elif age <= 59:
44         return "Adult"
45     else:
46         return "Senior"
47
48 def main():
49     age = int(input("Enter your age: "))
50     category = classify_age(age)
51     print(f"You are classified as: {category}")
52
53 if __name__ == "__main__":
54     main()
55
56

```

The terminal at the bottom shows the command prompt running the script, and the output is 'You are classified as: Adult'.

## Code Analysis :

- ☐ A function `classify_age()` checks age using **if–elif–else** conditions.
- ☐ Different age ranges are mapped to categories: Child, Teenager, Adult, Senior.
- ☐ Invalid age (negative value) is handled using a condition.
- ☐ User input is taken and passed to the function.
- ☐ The classification result is printed to the user.

## Task-4

Prompt: **generate a code to calculate the sum of first n natural numbers using while loop and user input.**

Code :

```
def sum_of_natural_numbers(n):

    total = 0

    count = 1

    while count <= n:

        total += count

        count += 1

    return total

def main():

    n = int(input("Enter a positive integer: "))

    if n < 1:

        print("Please enter a positive integer greater than 0.")

    else:

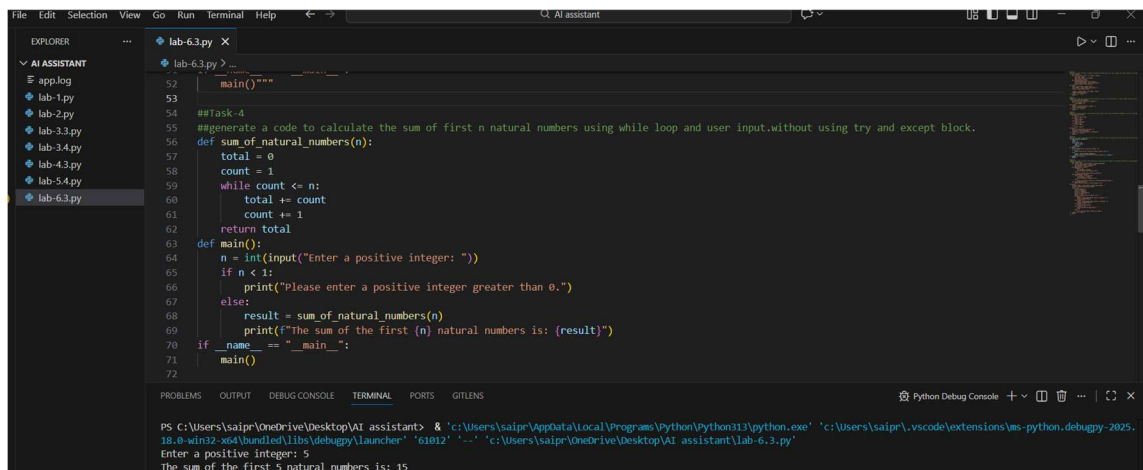
        result = sum_of_natural_numbers(n)

        print(f"The sum of the first {n} natural numbers is: {result}")

if __name__ == "__main__":

    main()
```

Output:



The screenshot shows a Visual Studio Code editor window with a Python file named 'lab-6.3.py'. The code in the file is as follows:

```
52 def main():
53     """
54     ##Task-4
55     #generate a code to calculate the sum of first n natural numbers using while loop and user input,without using try and except block.
56     def sum_of_natural_numbers(n):
57         total = 0
58         count = 1
59         while count <= n:
60             total += count
61             count += 1
62         return total
63     def main():
64         n = int(input("Enter a positive integer: "))
65         if n < 1:
66             print("Please enter a positive integer greater than 0.")
67         else:
68             result = sum_of_natural_numbers(n)
69             print(f"The sum of the first {n} natural numbers is: {result}")
70     if __name__ == "__main__":
71         main()
72
```

The terminal at the bottom shows the command prompt and the output of the script:

```
PS C:\Users\sai\r\OneDrive\Desktop\AI assistant> & 'c:\Users\sai\r\AppData\Local\Programs\Python\Python11\python.exe' 'c:\Users\sai\r\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61012' '-.' 'c:\Users\sai\r\OneDrive\Desktop\AI assistant\lab-6.3.py'
Enter a positive integer: 5
The sum of the first 5 natural numbers is: 15
```

Code Analysis :

- ☐ A function `sum_of_natural_numbers()` calculates the sum using a **while loop**.
- ☐ A counter starts from 1 and runs until n.
- ☐ Each number is added to the total variable.
- ☐ Input validation checks whether n is positive.
- ☐ The final sum is displayed to the user.

Task-5

Prompt: **genearte a code to design a bank application using class with methods like deposit, withdraw and check balance take user input.**

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:
```

```
        self.balance += amount
```

```
        print(f"Deposited: ${amount:.2f}")
```

```
    else:
```

```
        print("Deposit amount must be positive.")
```

```
def withdraw(self, amount):
```

```
    if 0 < amount <= self.balance:
```

```
        self.balance -= amount
```

```
        print(f"Withdrew: ${amount:.2f}")
```

```
    else:
```

```
        print("Insufficient balance or invalid withdrawal amount.")
```

```
def check_balance(self):
```

```

        print(f"Current balance: ${self.balance:.2f}")
def main():
    account_holder = input("Enter account holder name: ")
    account = BankAccount(account_holder)
    while True:
        print("\nOptions:")
        print("1. Deposit")
        print("2. Withdraw")
        print("3. Check Balance")
        print("4. Exit")
        choice = input("Choose an option (1-4): ")
        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.check_balance()
        elif choice == '4':
            print("Exiting the application.")
            break
        else:
            print("Invalid choice. Please try again.")
if __name__ == "__main__":
    main()

```

Output :

```
75 class BankAccount:
76     def __init__(self, account_holder, initial_balance=0):
77         self.account_holder = account_holder
78         self.balance = initial_balance
79     def deposit(self, amount):
80         if amount > 0:
81             self.balance += amount
82             print(f"Deposited: ${amount:.2f}")
83         else:
84             print("Deposit amount must be positive.")
85     def withdraw(self, amount):
86         if 0 < amount <= self.balance:
87             self.balance -= amount
88             print(f"Withdrew: ${amount:.2f}")
89         else:
90             print("Insufficient balance or invalid withdrawal amount.")
91     def check_balance(self):
92         print(f"Current balance: ${self.balance:.2f}")
93
94 def main():
95     account_holder = input("Enter account holder name: ")
96     account = BankAccount(account_holder)
97     while True:
98         print("\nOptions:")
99         print("1. Deposit")
100        print("2. Withdraw")
101        print("3. Check Balance")
102        print("4. Exit")
103        choice = input("Choose an option (1-4): ")
104        if choice == '1':
105            amount = float(input("Enter amount to deposit: "))
106            account.deposit(amount)
107        elif choice == '2':
108            amount = float(input("Enter amount to withdraw: "))
109            account.withdraw(amount)
110        elif choice == '3':
111            account.check_balance()
112        elif choice == '4':
113            break
114
115 if __name__ == '__main__':
116     main()
```

PS C:\Users\sai\r\OneDrive\Desktop\AI assistant> & 'c:\Users\sai\r\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\sai\r\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\_launcher' 58480 -c 'c:\Users\sai\r\OneDrive\Desktop\AI assistant\lab-6.3.py'

Enter account holder name: sai

Options:

1. Deposit

2. Withdraw

3. Check Balance

4. Exit

Choose an option (1-4): 1

Enter amount to deposit: 100

Deposited: \$100.00

## Code Analysis :

- ❑ A class BankAccount is used to represent a bank account.
- ❑ The constructor initializes account holder name and balance.
- ❑ deposit() method adds money after validating the amount.
- ❑ withdraw() method deducts money if sufficient balance is available.