

## ASSIGNMENT -6.3

2303a51343

b-10

### TASK-1

**Prompt:** Create a Python Student class with name, roll\_number, and branch.

Take all values from the user using input() Include \_\_init\_\_() and display\_details() methods.

Create an object using user input and display the details

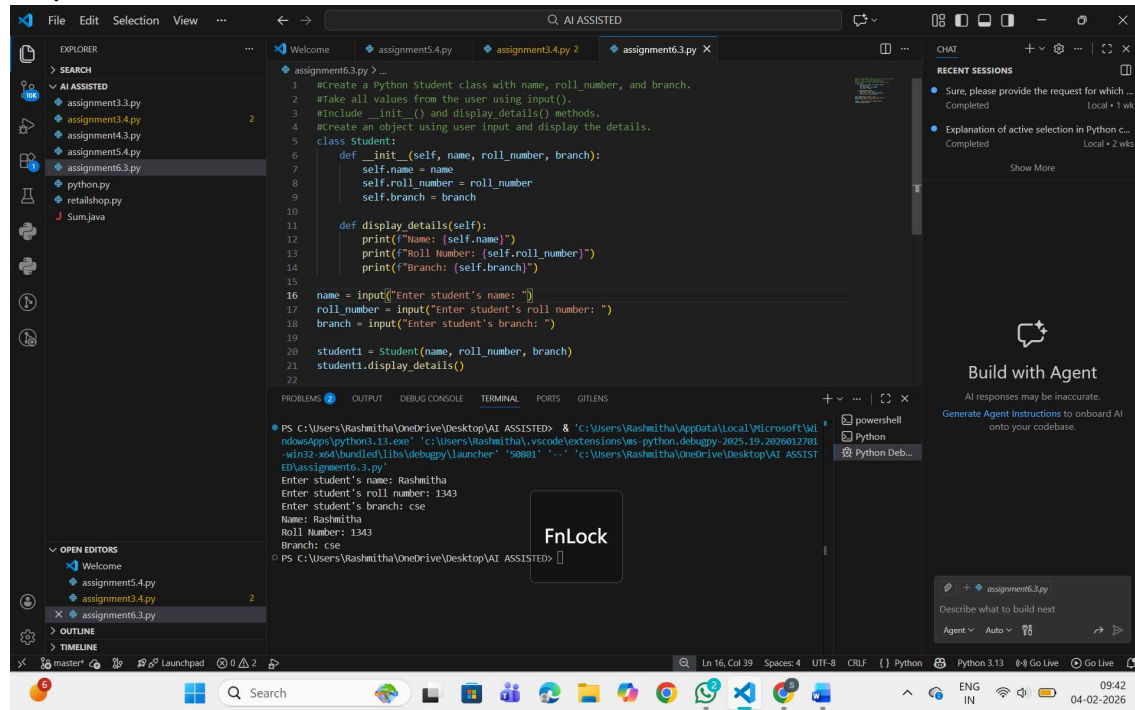
**Code:** class Student:

```
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}")

name = input("Enter student's name: ")
roll_number = input("Enter student's roll number: ")
branch = input("Enter student's branch: ")
student1 = Student(name, roll_number, branch)
student1.display_details()
```

## Output:



## Analysis:

- AI-generated code correctly defines a Student class with a constructor to initialize name, roll number, and branch.
- The display\_details() method clearly prints the student information, making the output easy to understand.
- Overall, the code is simple, readable, and follows proper object-oriented programming

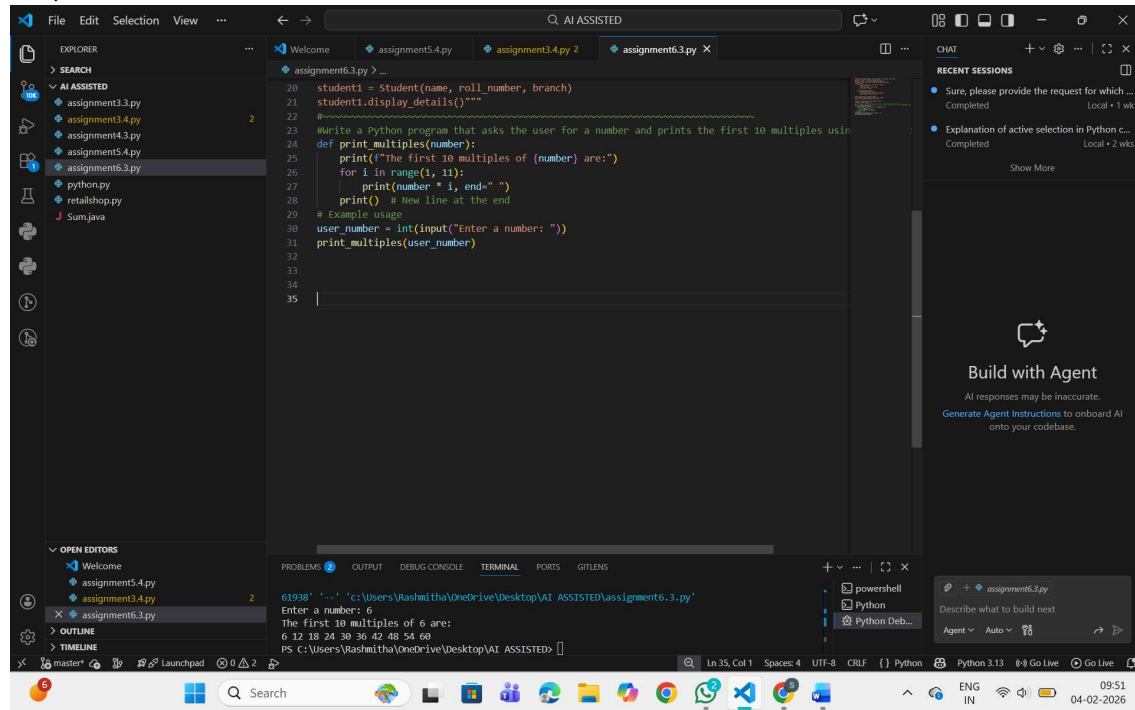
## TASK-2

**Prompt:** Write a Python program that asks the user for a number and prints the first 10 multiples using a for loop

### Code:

```
def print_multiples(number):  
    print(f"The first 10 multiples of {number} are:")  
    for i in range(1, 11):  
        print(number * i, end=" ")  
    print() # New line at the end  
  
# Example usage  
user_number = int(input("Enter a number: "))  
print_multiples(user_number)
```

Output:



The screenshot shows a Visual Studio Code editor with a Python file named 'assignment6.3.py'. The code defines a 'Student' class and a 'print\_multiples' function. The function uses a for loop to print the first 10 multiples of a given number. The terminal output shows the program running successfully, displaying the first 10 multiples of 6.

```
20 student1 = Student(name, roll_number, branch)
21 student1.display_details()
22
23 #Write a Python program that asks the user for a number and prints the first 10 multiples using a loop
24 def print_multiples(number):
25     print(f"The first 10 multiples of {number} are:")
26     for i in range(1, 11):
27         print(number * i, end=" ")
28     print() # New line at the end
29
30 # Example usage
31 user_number = int(input("Enter a number: "))
32 print_multiples(user_number)
33
34
35
```

Terminal Output:

```
61938' ... 'c:\Users\Rashmi\OneDrive\Desktop\AI ASSISTED\assignment6.3.py'
Enter a number: 6
The first 10 multiples of 6 are:
6 12 18 24 30 36 42 48 54 60
PS C:\Users\Rashmi\OneDrive\Desktop\AI ASSISTED>
```

### Analysis:

The AI-generated code correctly uses a loop to print the first 10 multiples of a given number.

The loop logic is clear and easy to understand, Both the for loop and while loop versions work properly and produce the expected

### TASK-3:

**Prompt:** Write a Python program that takes age as user input and classifies it as child, teenager, adult, or senior using if-elif-else.

**Code:** def classify\_age(age):

if age < 13:

return "Child"

elif 13 <= age < 20:

return "Teenager"

elif 20 <= age < 60:

return "Adult"

else:

return "Senior"

# Example usage

user\_age = int(input("Enter your age: "))

```
age_group = classify_age(user_age)

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

### Output:

```
assignment6.3.py > ...
30 user_number = int(input("Enter a number: "))
31 print_multiples(user_number)"""
32 #-----
33 #Write a Python program that takes age as user input and classifies it as child, teenager, adult, or senior.
34 def classify_age(age):
35     if age < 13:
36         return "child"
37     elif 13 <= age < 20:
38         return "teenager"
39     elif 20 <= age < 60:
40         return "adult"
41     else:
42         return "senior"
43 # Example usage
44 user_age = int(input("Enter your age: "))
45 age_group = classify_age(user_age)
46 print(f"You are classified as: {age_group}")
47
48
49
50
51
```

```
PS C:\Users\Rashmi\OneDrive\Desktop\AI ASSISTED> & 'C:\Users\Rashmi\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Rashmi\OneDrive\Desktop\AI ASSISTED\assignment6.3.py'
Enter your age: 15
You are classified as: Teenager
PS C:\Users\Rashmi\OneDrive\Desktop\AI ASSISTED>
```

**Analysis:** The AI-generated code uses simple if-elif-else conditions to check the age entered by the user. Each condition clearly matches an age group such as child, teenager, adult, or senior. The code is easy to read and gives correct results for different age values.

### TASK-4

**Prompt:** Write a Python program that takes n as user input and calculates the sum of the first n natural numbers using a for loop.

**Code:** `def sum_of_natural_numbers(n):`

```
    total = 0
```

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

```
        total += i
```

```
    return total
```

**# Example usage**

```
user_n = int(input("Enter a natural number n: "))
```

```
sum_n = sum_of_natural_numbers(user_n)
```

```
print(f"The sum of the first {user_n} natural numbers is: {sum_n}")
```

## Output:

The screenshot shows a Visual Studio Code editor with a dark theme. The Explorer panel on the left shows a project with several Python files. The main editor window displays two Python scripts. The first script, 'assignment6.3.py', contains an age classification function and an example usage. The second script, 'assignment6.3.py', contains a function to calculate the sum of the first n natural numbers and an example usage. The terminal at the bottom shows the execution of the second script, with the user entering '12' and the program outputting 'The sum of the first 12 natural numbers is: 78'.

```
37 elif 13 <= age < 20:
38     return "Teenager"
39 elif 20 <= age < 60:
40     return "Adult"
41 else:
42     return "Senior"
43
44 # Example usage
45 user_age = int(input("Enter your age: "))
46 age_group = classify_age(user_age)
47 print(f"You are classified as: {age_group}")
48
49 # Write a python program that takes n as user input and calculates the sum of the first n natural
50 # numbers
51 def sum_of_natural_numbers(n):
52     total = 0
53     for i in range(1, n + 1):
54         total += i
55     return total
56
57 # Example usage
58 user_n = int(input("Enter a natural number n: "))
59 sum_n = sum_of_natural_numbers(user_n)
60 print(f"The sum of the first {user_n} natural numbers is: {sum_n}")
61
62
```

```
buggy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher "51838" -... 'c:\Users\Rashmitha\OneDrive\Desktop\AI'
ASSISTED\assignment6.3.py
Enter a natural number n: 12
The sum of the first 12 natural numbers is: 78
PS C:\Users\Rashmitha\OneDrive\Desktop\AI> ASSISTED>
```

**Analysis:** The AI-generated code correctly calculates the sum of the first n natural numbers using a loop. The logic is simple and easy to understand, whether it uses a for loop, a while loop, or a formula. The program gives the correct result for any positive number entered by the user.

## TASK-5

**Prompt:** Create a Python Bankaccount class that takes account holder name and initial balance from the user. Include deposit(), withdraw(), and check\_balance() methods. Allow the user to perform deposit and withdrawal operations using input().

**Code:** class BankAccount:

```
def __init__(self, holder_name, initial_balance):
    self.holder_name = holder_name
    self.balance = initial_balance

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

def withdraw(self, amount):
    if amount > self.balance:
        print("Insufficient balance!")
```

else:

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

```
print(f"Withdrew: {amount}. New balance: {self.balance}")
```

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

```
    print(f"Current balance: {self.balance}")
```

```
holder_name = input("Enter account holder name: ")
```

```
initial_balance = float(input("Enter initial balance: "))
```

```
account = BankAccount(holder_name, initial_balance)
```

```
account.check_balance()
```

```
deposit_amount = float(input("Enter amount to deposit: "))
```

```
account.deposit(deposit_amount)
```

```
withdraw_amount = float(input("Enter amount to withdraw: "))
```

```
account.withdraw(withdraw_amount)
```

```
account.check_balance()
```

## Output:

The screenshot shows a VS Code editor with a Python file named `assignment6.3.py`. The code defines a `BankAccount` class with methods for deposit, withdrawal, and balance checking. The main script creates an account, checks the balance, and performs a withdrawal. The terminal output shows the execution of the script, including the prompts for user input and the resulting balance changes.

```
57 print(f"The sum of the first {user_n} natural numbers is: {sum_n}")
58
59 #Create a Python BankAccount class that takes account holder name and initial balance from the user.
60 #Include deposit(), withdraw(), and check_balance() methods#Allow the user to perform deposit and withdrawal operations using inputs
61 class BankAccount:
62     def __init__(self, holder_name, initial_balance):
63         self.holder_name = holder_name
64         self.balance = initial_balance
65
66     def deposit(self, amount):
67         self.balance += amount
68         print(f"Deposited: {amount}. New balance: {self.balance}")
69
70     def withdraw(self, amount):
71         if amount > self.balance:
72             print("Insufficient balance!")
73         else:
74             self.balance -= amount
75             print(f"Withdrew: {amount}. New balance: {self.balance}")
76
77     def check_balance(self):
78         print(f"Current balance: {self.balance}")
79
80 holder_name = input("Enter account holder name: ")
81 initial_balance = float(input("Enter initial balance: "))
82 account = BankAccount(holder_name, initial_balance)
83 account.check_balance()
84 deposit_amount = float(input("Enter amount to deposit: "))
85 account.deposit(deposit_amount)
86 withdraw_amount = float(input("Enter amount to withdraw: "))
87 account.withdraw(withdraw_amount)
88 account.check_balance()
89
90
91
92
```

Terminal Output:

```
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED> cd 'C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED' & 'C:\Users\Rashmitha\AppData\Local\MicrosoftApps\python3.13.exe' 'C:\Users\Rashmitha\vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '59292' '-
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED> cd 'C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED' & 'C:\Users\Rashmitha\AppData\Local\MicrosoftApps\python3.13.exe' 'C:\Users\Rashmitha\vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundled\libs\debugpy\launcher' '59292' '-
Enter amount to withdraw: 200
Withdrew: 200.0. New balance: 4900.0
Current balance: 4900.0
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED>
```

**Analysis:** Your Bankaccount code is functional using classes and methods for deposit, withdrawal, and balance checks.

It handles basic operations and prevents overdrafts but lacks input validation and only allows one transaction of each type.

