AI Assisted Coding

Week:6.3

Ht.No:2303A52350

Batch:45

Task 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.

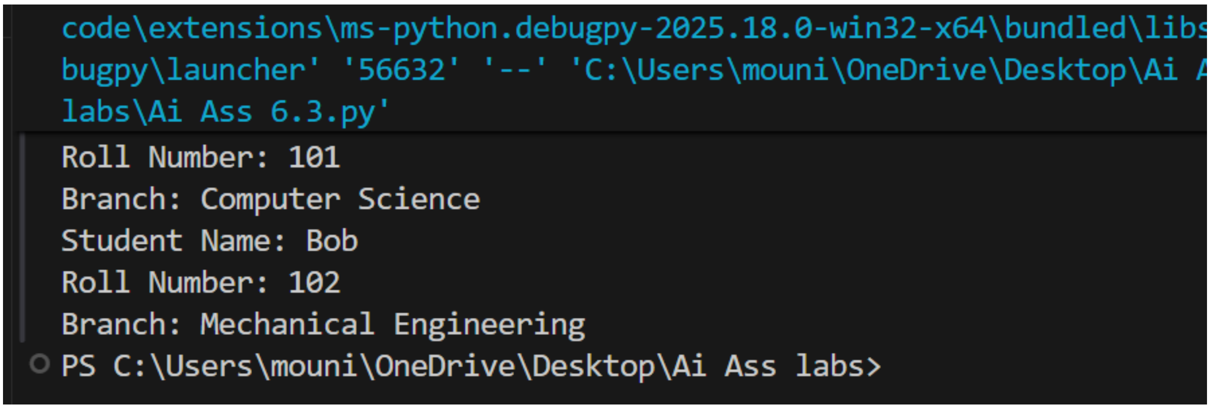
Prompt:

Create a student information management module 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.

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}")
# Example usage:
student1 = Student("Alice", "101", "Computer Science")
student1.display_details()
student2 = Student("Bob", "102", "Mechanical Engineering")
student2.display_details()
```

Output:



```
code\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\lib
bugpy\launcher' '56632' '--' 'C:\Users\mouni\OneDrive\Desktop\Ai
labs\Ai Ass 6.3.py'
Roll Number: 101
Branch: Computer Science
Student Name: Bob
Roll Number: 102
Branch: Mechanical Engineering
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>
```

Explanation:

Here we need to create a student management system where their names,branch and roll no include in attributes and display the student details.

Task 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)

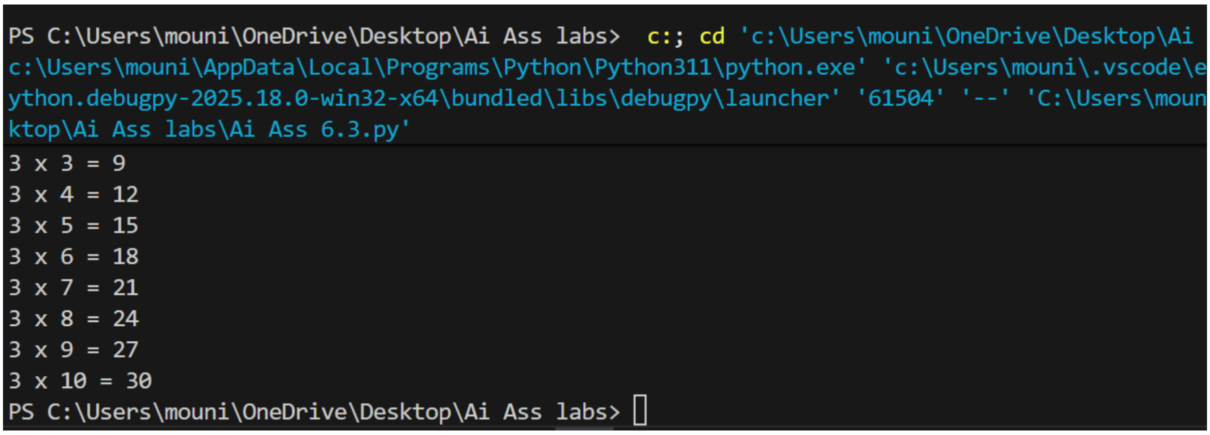
Prompt:2.1

To generate a function that prints the first 10 multiples of a given number using a loop.

Code:2.1

```
class Multiples:
    def __init__(self, number):
        self.number = number
    def print_multiples(self):
        for i in range(1, 11):
            print(f"{self.number} x {i} = {self.number * i}")
# Example usage:
num = int(input("Enter a number to generate its first 10 multiples: "))
multiples = Multiples(num)
multiples.print_multiples()
```

Output:2.1



```
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>  c:; cd 'c:\Users\mouni\OneDrive\Desktop\Ai
c:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\mouni\.vscode\e
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61504' '--' 'C:\Users\moun
ktop\Ai Ass labs\Ai Ass 6.3.py'
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs> 
```

Explanation:2.1

We have generated a python code in basic version for the first 10 multiples of a given number.

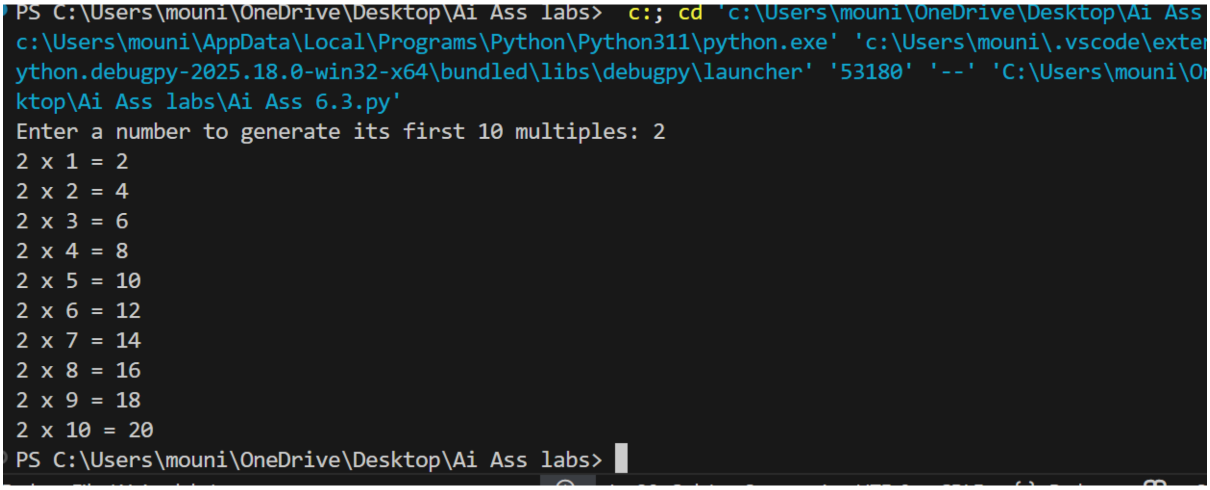
Prompt:2.2

Generate the same funcionality using another controlled looping structure

Code:2.2

```
class Multiples:
    def __init__(self, number):
        self.number = number
    def print_multiples(self):
        count = 1
        while count <= 10:
            print(f"{self.number} x {count} = {self.number * count}")
            count += 1
# Example usage:
num = int(input("Enter a number to generate its first 10 multiples: "))
multiples = Multiples(num)
multiples.print_multiples()
```

Output:2.2



```
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>  c:; cd 'c:\Users\mouni\OneDrive\Desktop\Ai Ass
c:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\mouni\.vscode\exte
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53180' '--' 'C:\Users\mouni\O
ktop\Ai Ass labs\Ai Ass 6.3.py'
Enter a number to generate its first 10 multiples: 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs> 
```

Explanation:2.2

We need to generate a code which prints the first 10 multiples of a number using a loop, and with same functionality but with different looping structure we need to generate another code.

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).

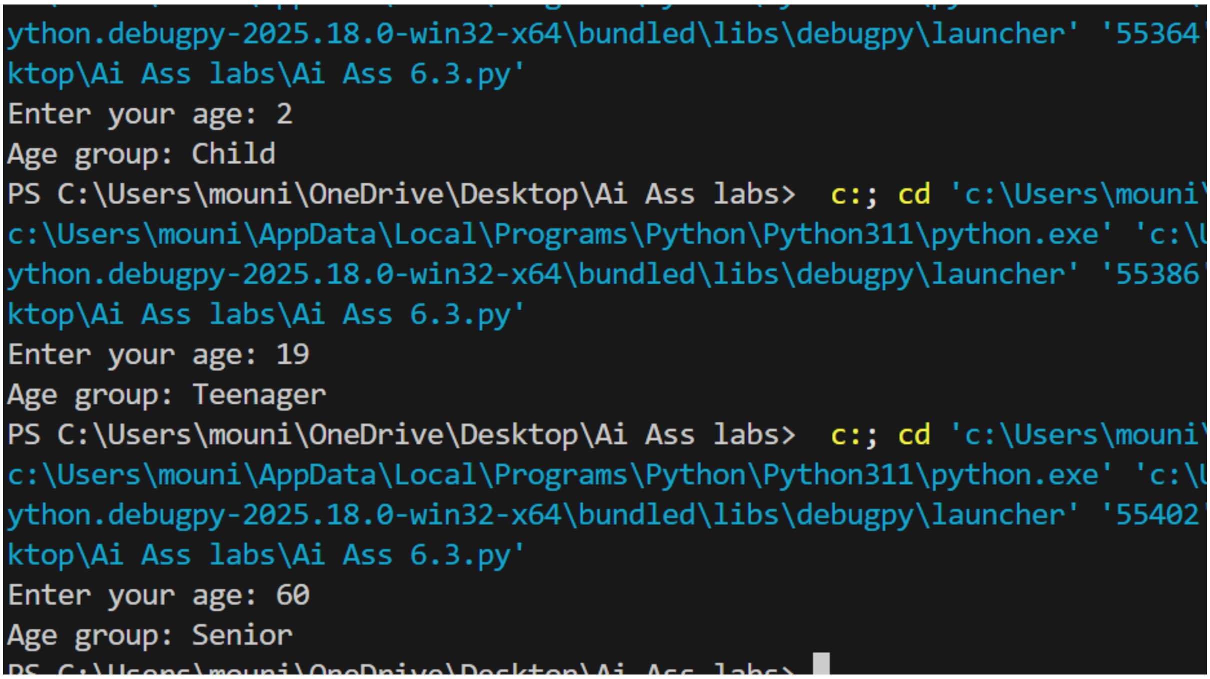
Prompt:3.1

To generate a python code using nested if-elif-else conditional statements to classify age groups (e.g., child, teenager, adult, senior).

Code:3.1

```
class AgeClassifier:
    def __init__(self, age):
        self.age = age
    def classify_age(self):
        if self.age < 0:
            return "Invalid age"
        elif self.age <= 12:
            return "Child"
        elif self.age <= 19:
            return "Teenager"
        elif self.age <= 59:
            return "Adult"
        else:
            return "Senior"
# Example usage:
age = int(input("Enter your age: "))
classifier = AgeClassifier(age)
print(f"Age group: {classifier.classify_age()}")
```

Output:3.1



```
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55364
ktop\Ai Ass labs\Ai Ass 6.3.py'
Enter your age: 2
Age group: Child
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>  c:; cd 'c:\Users\mouni
c:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe' 'c:\
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55386
ktop\Ai Ass labs\Ai Ass 6.3.py'
Enter your age: 19
Age group: Teenager
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>  c:; cd 'c:\Users\mouni
c:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe' 'c:\
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55402
ktop\Ai Ass labs\Ai Ass 6.3.py'
Enter your age: 60
Age group: Senior
```

Explanation:3.1

Here we need to genearate the code by using nested-if, if-elif-else conditional statements of the age groups like whether they are children,teenagers,adults etc.

Prompt:3.2

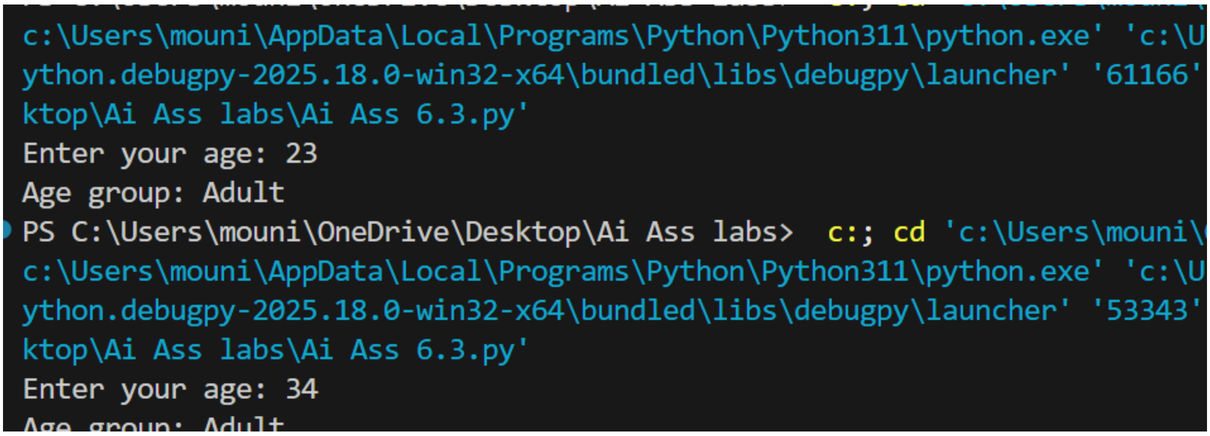
Generate the same classification using alternative conditional structures (e.g.,simplified conditions or dictionary-based logic).

Code:3.2

```
def classify_age(age):
    age_groups = {
        range(0, 13): "Child",
        range(13, 20): "Teenager",
        range(20, 60): "Adult",
        range(60, 150): "Senior"    }
    for age_range, group in age_groups.items():
        if age in age_range:
```

```
        return group
    return "Invalid age"
# Example usage:
age = int(input("Enter your age: "))
print(f"Age group: {classify_age(age)}")
```

Output:3.2



```
c:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe' 'c:\U
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61166'
ktop\Ai Ass labs\Ai Ass 6.3.py'
Enter your age: 23
Age group: Adult
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>  c:; cd 'c:\Users\mouni\
c:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe' 'c:\U
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53343'
ktop\Ai Ass labs\Ai Ass 6.3.py'
Enter your age: 34
Age group: Adult
```

Explanation:3.2

The same code we have generated by using for loop and if condition to check whether they are senior,adults,children and teenagers.

Task 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.

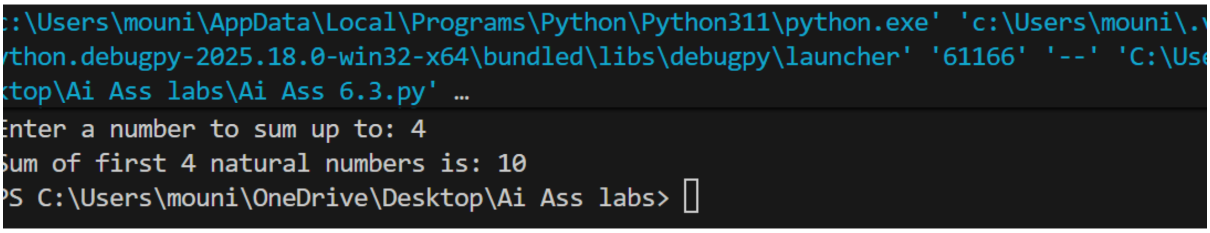
Prompt:4.1

Generate python code into a sum_to_n() function using a for loop.

Code:4.1

```
def sum_to_n(n):
    total = 0
    for i in range(1, n + 1):
        total += i
    return total
# Example usage:
n = int(input("Enter a number to sum up to: "))
print(f"Sum of first {n} natural numbers is: {sum_to_n(n)}")
```

Output:4.1



```
:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\mouni\.
thon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61166' '--' 'C:\Use
top\Ai Ass labs\Ai Ass 6.3.py' …
nter a number to sum up to: 4
um of first 4 natural numbers is: 10
S C:\Users\mouni\OneDrive\Desktop\Ai Ass labs> 
```

Explanation:4.1

We have generated a python code by using for loop to generate sum of n numbers.

Prompt4.2:

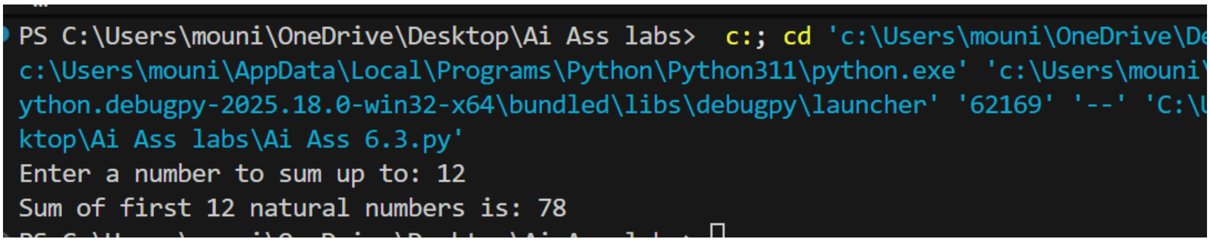
Generate a python code to implement sum of n numbers using a while loop or a mathematical formula.

Code:4.2

```
def sum_to_n(n):
 # Using mathematical formula: n * (n + 1) / 2
    return n * (n + 1) // 2
```

```
# Example usage:
n = int(input("Enter a number to sum up to: "))
print(f"Sum of first {n} natural numbers is: {sum_to_n(n)}")
```

Output:4.2



```
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>  c:; cd 'c:\Users\mouni\OneDrive\De
c:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\mouni
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62169' '--' 'C:\
ktop\Ai Ass labs\Ai Ass 6.3.py'
Enter a number to sum up to: 12
Sum of first 12 natural numbers is: 78
```

Explanation:4.2

Compared to code 4.1 the code 4.2 is done by using a mathematical formula for a same question sum of n numbers.

Task 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.

Prompt:

Generate a Bank Account class with methods such as deposit(), withdraw(),and check_balance().In python.

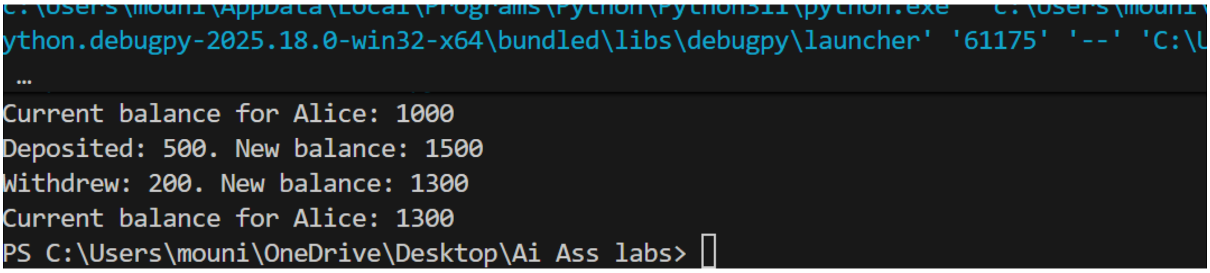
Code:

```
def generate_bank_account_class():
    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}. New balance: {self.balance}")
            else:
                print("Deposit amount must be positive.")
        def withdraw(self, amount):
            if amount > 0:
                if amount <= self.balance:
                    self.balance -= amount
                    print(f"Withdrew: {amount}. New balance: {self.balance}")
                else:
                    print("Insufficient funds.")
            else:
                print("Withdrawal amount must be positive.")
        def check_balance(self):
            print(f"Current balance for {self.account_holder}: {self.balance}")
    return BankAccount
# Example usage:
BankAccount = generate_bank_account_class()
account = BankAccount("Alice", 1000)
account.check_balance()
account.deposit(500)

account.withdraw(200)

account.check_balance()
```

Output:



```
C:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe  c:\Users\mouni\
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61175' '--' 'C:\U
 …
Current balance for Alice: 1000
Deposited: 500. New balance: 1500
Withdrew: 200. New balance: 1300
Current balance for Alice: 1300
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs> 
```

Explanation:

We have generated the python code to check bank details by using if else condition and to check available balance ,new balance and the amount withdrawl.