# AI ASSISTANT CODING

## ASSIGNMENT-6.3

## 2303A52376

## BT.NO.45

## TASK1:

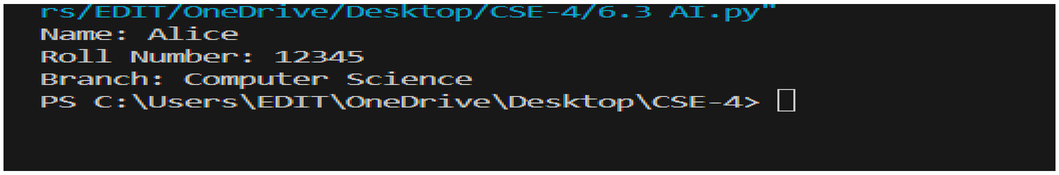
### PROMPT:

Develop a simple student information management module. The class should include attributes such as name, roll number, and branch.Add a method display_details() to print student information. Analyze the code generated by the AI tool for correctness and clarity.

### 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_
student1 = Student("Alice", "12345", "Computer Science")
student1.display_details()
```

### OUTPUT:



```
rs/EDIT/OneDrive/Desktop/CSE-4/6.3 AI.py"
Name: Alice
Roll Number: 12345
Branch: Computer Science
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> □
```

The AI-generated code for the Student class is correct and follows standard Python conventions. The constructor (__init__) initializes the attributes name, roll_number, and branch, while the display_details() method neatly formats and prints the student's information. The code is clear and easy to understand, making it suitable for educational purposes. Overall, the AI tool effectively generated a functional and well-structured class based on the given requirements.

## EXPLANATION:

Write a code for printing student information like name,branch,year etc.. and analyze the code clarity and corrections.

## TASK 2:

## PROMPT:

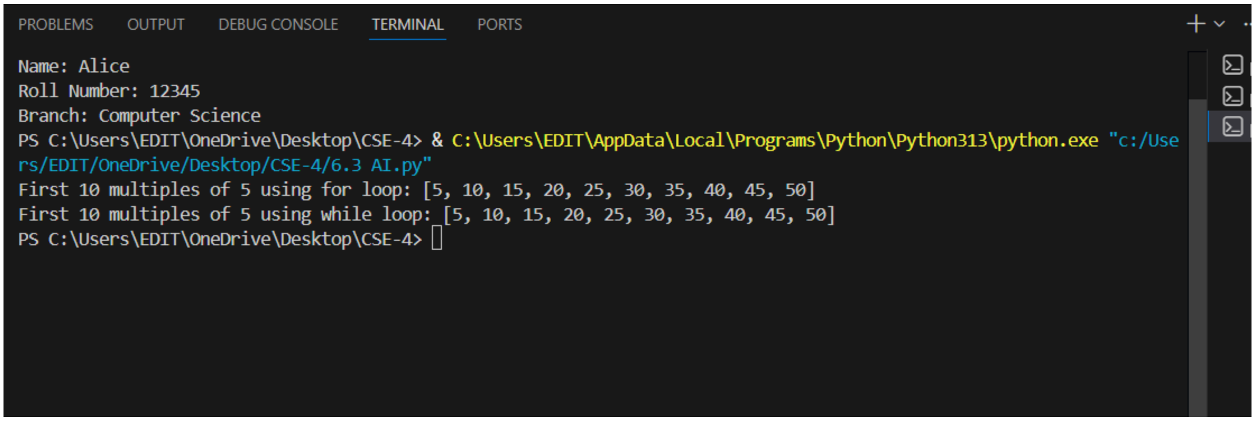
generate a function that prints the first 10 multiples of a given number
using a for and while loop.

## CODE:

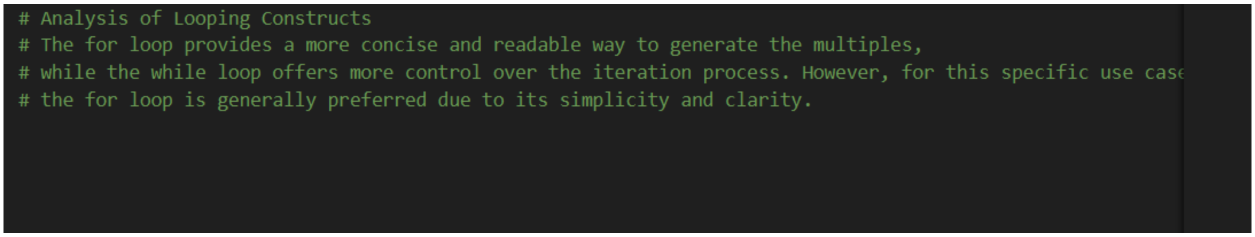
```
def print_multiples_for_loop(number):
    multiples = []
    for i in range(1, 11):
        multiples.append(number * i)
    return multiples
def print_multiples_while_loop(number):
    multiples = []
    i = 1
    while i <= 10:
        multiples.append(number * i)
        i += 1
    return multiples
number = 5
```

```
multiples_for = print_multiples_for_loop(number)

multiples_while = print_multiples_while_loop(number)

print(f"First 10 multiples of {number} using for loop: {multiples_for}")

print(f"First 10 multiples of {number} using while loop: {multiples_while}")
```

## OUTPUT:



```
Name: Alice
Roll Number: 12345
Branch: Computer Science
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> & C:\Users\EDIT\AppData\Local\Programs\Python\Python313\python.exe "c:/Use
rs/EDIT/OneDrive/Desktop/CSE-4/6.3 AI.py"
First 10 multiples of 5 using for loop: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
First 10 multiples of 5 using while loop: [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> 
```



```
# Analysis of Looping Constructs
# The for loop provides a more concise and readable way to generate the multiples,
# while the while loop offers more control over the iteration process. However, for this specific use case
# the for loop is generally preferred due to its simplicity and clarity.
```

## EXPLANATION:

Print first 10 multiples of a given number using for and while loop and analyze the differences and observe the logics.

## TASK 3:

## PROMPT:

Build a basic classification system based on age. Use nested if-elif-else conditional statements to classify age groups. Analyze the generated conditions and logic. generate the same classification using alternative conditional structures (e.g.,
simplified conditions or dictionary-based logic).

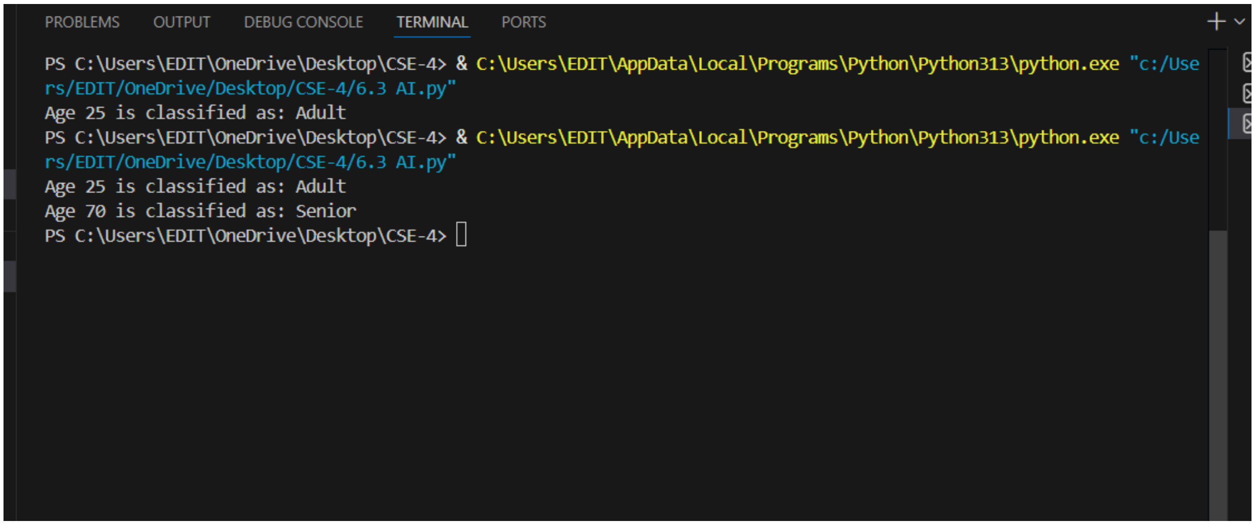
## CODE:

```
def classify_age(age):
    if age < 13:
        return "Child"
    elif 13 <= age < 20:
        return "Teenager"
    elif 20 <= age < 60:
        return "Adult"
    else:
        return "Senior"
age = 25
age_group = classify_age(age)
print(f"Age {age} is classified as: {age_group}")
```

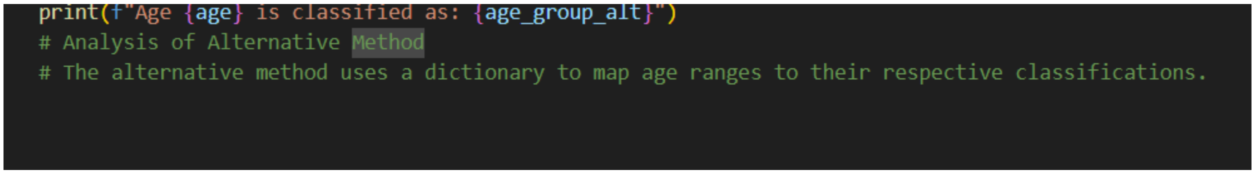
```
def classify_age_alternative(age):
    age_groups = {
        range(0, 13): "Child",
        range(13, 20): "Teenager",
        range(20, 60): "Adult",
        range(60, 150): "Senior"
    }
    for age_range, group in age_groups.items():
        if age in age_range:
            return group
age = 70
```

```
age_group_alt = classify_age_alternative(age)

print(f"Age {age} is classified as: {age_group_alt}")
```

## OUTPUT:



```
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> & C:\Users\EDIT\AppData\Local\Programs\Python\Python313\python.exe "c:/Use
rs/EDIT/OneDrive/Desktop/CSE-4/6.3 AI.py"
Age 25 is classified as: Adult
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> & C:\Users\EDIT\AppData\Local\Programs\Python\Python313\python.exe "c:/Use
rs/EDIT/OneDrive/Desktop/CSE-4/6.3 AI.py"
Age 25 is classified as: Adult
Age 70 is classified as: Senior
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> 
```



```
print(f"Age {age} is classified as: {age_group_alt}")
# Analysis of Alternative Method
# The alternative method uses a dictionary to map age ranges to their respective classifications.
```

## EXPLANATION:

Writing a code to classify ages by categories like adult,child etc..by using nested if elif else conditions and use altenative method and compare both and analyze it.

## TASK 4:

## PROMPT:

calculate the sum of the first n natural numbers.generate a sum_to_n() function using a for loop.suggest an alternative implementation using a while loop or a mathematical FORMULA Mathematical formula implementation.Analysis of Different Implementations.

## CODE:

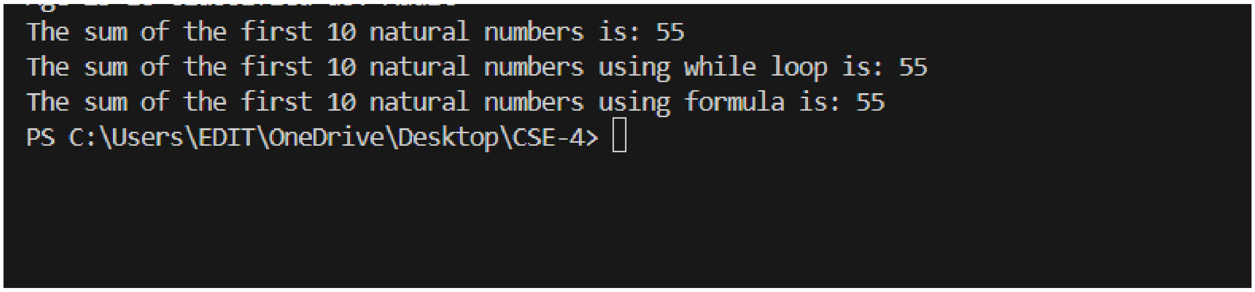
```
def sum_to_n(n):

  total = 0
```

```
    for i in range(1, n + 1):
        total += i
    return total
# Example usage
n = 10
result = sum_to_n(n)
print(f"The sum of the first {n} natural numbers is: {result}")


def sum_to_n_while(n):
    total = 0
    i = 1
    while i <= n:
        total += i
        i += 1
    return total
# Example usage of while loop implementation
n = 10
result_while = sum_to_n_while(n)
print(f"The sum of the first {n} natural numbers using while loop is: {result_while}")


def sum_to_n_formula(n):
    return n * (n + 1) // 2
# Example usage of formula implementation
n = 10
```

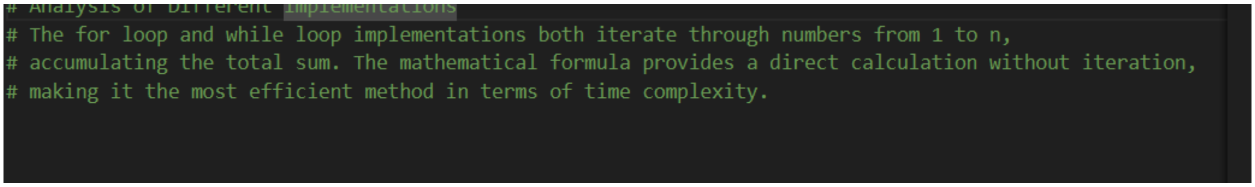
result_formula = sum_to_n_formula(n)

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

### OUTPUT:



```
The sum of the first 10 natural numbers is: 55
The sum of the first 10 natural numbers using while loop is: 55
The sum of the first 10 natural numbers using formula is: 55
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4>
```



```
# Analysis of Different Implementations
# The for loop and while loop implementations both iterate through numbers from 1 to n,
# accumulating the total sum. The mathematical formula provides a direct calculation without iteration,
# making it the most efficient method in terms of time complexity.
```

### EXPLANATION:

Print sum of first ten natural numbers.use alternative methods and use mathematical formula method also.

### TASK 5:

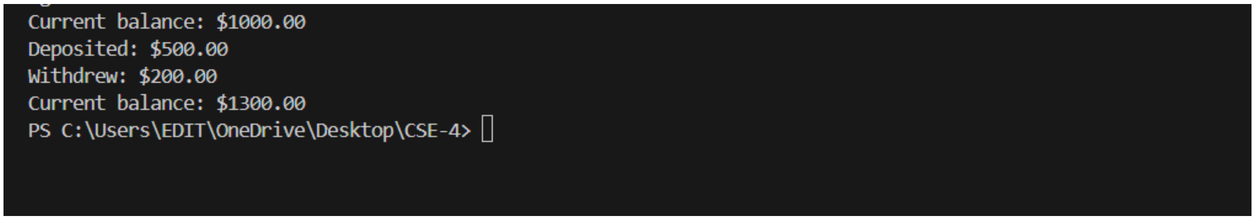
### PROMPT:

Design a basic banking application.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.

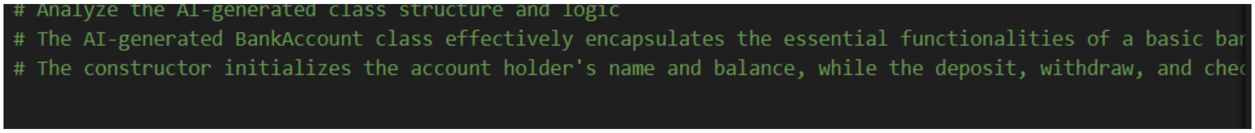
### CODE:

```
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 funds or invalid withdrawal amount.")

  def check_balance(self):
    print(f"Current balance: ${self.balance:.2f}")
# Sample Usage
account = BankAccount("John Doe", 1000)
account.check_balance()
account.deposit(500)
account.withdraw(200)
account.check_balance()
```

**OUTPUT:**



```
Current balance: $1000.00
Deposited: $500.00
Withdrew: $200.00
Current balance: $1300.00
PS C:\Users\EDIT\OneDrive\Desktop\CSE-4> 
```



```
# Analyze the AI-generated class structure and logic
# The AI-generated BankAccount class effectively encapsulates the essential functionalities of a basic ban
# The constructor initializes the account holder's name and balance, while the deposit, withdraw, and chec
```

## EXPLANATION:

Here we design a back account code to withdraw or deposit money and check savings etc..