

AI ASSISTED CODING Hall

Name : G. Sanjay

Ticket No: 2303A510B9

Batch:13

Assignment-6.1 **Task-1:**

Task Description #1 (AI-Based Code Completion for Loops) Task: Use an AI code completion tool to generate a loop-based Program

Prompt :

#Ok Generate Python code to print all even numbers between 1 and N using a loop.

Code and Out put: Using For loop

A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with several Python files. The main editor window displays the following code:

```
1 ##
2 # Using the for Loop:
3 N = int(input("Enter N: "))
4 for i in range(2, N + 1, 2):
5     print(i)
```

The terminal at the bottom shows the output of running the script:

```
PS C:\Users\gandr\Documents\3Year-2Sem\Ai assistants Coding\AI code> & C:/Users/gandr/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/gandr/Documents/3Year-2Sem/Ai assistants Coding/AI code/Assess 6.1.py"
Enter N: 20
2
4
6
8
10
12
14
16
18
20
```

The right panel contains AI-generated code completion suggestions:

- Efficient (for loop with step 2):

```
N = int(input("Enter N: "))
for i in range(2, N + 1, 2):
    print(i)
```
- Alternative (while loop):

```
N = int(input("Enter N: "))
i = 2
while i <= N:
    print(i)
    i += 2
```

A tip at the bottom of the completion panel says: "Tip: If you want validation, wrap `int(input(...))` in a `try/except` block to handle non-integer input."

Code : Using While Loop/

A screenshot of the Visual Studio Code interface, similar to the previous one but with a different code snippet. The main editor window displays the following code:

```
1 ## Using the while Loop:
2 N = int(input("Enter N: "))
3 i = 2
4 while i <= N:
5     print(i)
6     i += 2
```

The terminal output is identical to the first screenshot:

```
PS C:\Users\gandr\Documents\3Year-2Sem\Ai assistants Coding\AI code> & C:/Users/gandr/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/gandr/Documents/3Year-2Sem/Ai assistants Coding/AI code/Assess 6.1.py"
Enter N: 2
Enter N: 23
2
4
6
8
10
12
14
16
18
20
22
```

The right panel contains the same AI-generated code completion suggestions as the first screenshot.

Task-2:

Task Description #2 (AI-Based Code Completion for Loop with Conditionals)

Task: Use an AI code completion tool to combine loops and conditionals program.

Prompt:

```
# ok Generate Python code to count how many numbers in a list are even and odd.
```

Code:

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files: 'Assess 6.1.py' (the active file), 'Ass-2.4.py', 'Ass-3.4.py', 'Task2-BiasDetection.py', 'users_protected дем..', and 'users_protected.csv'. The main editor area contains the following Python code:

```
16 #Task: 2
17 #####
18 def count_even_odd(numbers):
19     evens = sum(1 for x in numbers if isinstance(x, int) and x % 2 == 0)
20     odds = sum(1 for x in numbers if isinstance(x, int) and x % 2 != 0)
21     return evens, odds
22
23 # Example
24 nums = [1, 2, 3, 4, 5, 6]
25 ev, od = count_even_odd(nums)
26 print(f"Even: {ev}, Odd: {od}") # Even: 3, Odd: 3
```

To the right of the editor is a 'CHAT' pane with the title 'PRINT EVEN NUMBERS IN PYTHON USING...'. It displays the following text:

Simple function (handles non-integers by ignoring them):

```
def count_even_odd(numbers):
    evens = sum(1 for x in numbers if isinstance(x, int) and x % 2 == 0)
    odds = sum(1 for x in numbers if isinstance(x, int) and x % 2 != 0)
    return evens, odds
```

Example

```
nums = [1, 2, 3, 4, 5, 6]
ev, od = count_even_odd(nums)
print(f"Even: {ev}, Odd: {od}") # Even: 3, Odd: 3
```

One-line alternative:

```
evens = sum(x % 2 == 0 for x in numbers)
odds = len([x for x in numbers if x % 2 != 0])
```

Output:

The screenshot shows a terminal window with the following output:

```
NameError: name 'count_even_odd' is not defined
PS C:\Users\gandr\Documents\3Year-2Sem\AI assistants Coding\AI code> & C:/Users/gandr/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/gandr/Documents/3Year-2Sem/AI assistants Coding/AI code/Assess_6.1.py"
Even: 3, Odd: 3
PS C:\Users\gandr\Documents\3Year-2Sem\AI assistants Coding\AI code>
```

Task-3:

Task Description #3 (AI-Based Code Completion for Class Attributes Validation)

Task: Use an AI tool to complete a Python class that validates user input

Input

Prompt:

#Generate a Python class User that validates age and email using conditional statements

Code:

```

# Asses 6.1.py
class User:
    """User with simple age and email validation using conditional statements."""
    def __init__(self, name: str, age: int, email: str):
        self._name = name
        self._age = age
        self._email = email

    @property
    def age(self) -> int:
        return self._age

    @age.setter
    def age(self, value: int):
        if not isinstance(value, int):
            raise ValueError("Age must be an integer.")
        if value < 0:
            raise ValueError("Age must be non-negative.")
        self._age = value

    @property
    def email(self) -> str:
        return self._email

    @email.setter
    def email(self, value: str):
        if not isinstance(value, str):
            raise ValueError("Email must be a string.")
        if " " in value:
            raise ValueError("Email must not contain spaces.")
        if value[0] == "@":
            raise ValueError("Email must contain exactly one '@'.")
        local, domain = value.split("@")
        if not local:
            raise ValueError("Email local part must not be empty.")
        if "." not in domain:
            raise ValueError("Email domain must contain a '.'")
        if domain[-1] == ".":
            raise ValueError("Email domain must not start or end with a '.'")
        self._email = value

    def __repr__(self):
        return f"User(name={self.name}, age={self.age}, email={self.email})"

def input_user():
    while True:
        try:
            name = input("Enter name: ").strip()
            if not name:
                continue
            raw_age = input("Enter age: ").strip()
            try:
                age = int(raw_age)
            except ValueError:
                print("Age must be an integer.")
                continue
            email = input("Enter email: ").strip()
            user = User(name, age, email)
            print("Created", user)
            return user
        except ValueError as e:
            print(f"Validation error: {e}")
        except KeyboardInterrupt:
            print("\nCancelled by user.")
            return None
        finally:
            if __name__ == "__main__":
                input_user()

```

Output:

```

a/Local/Programs/Python/Python314/python.exe "c:/Users/gandr/Documents/3Year-2Sem/AI assistants Coding/AI code/Assess 6.1.py"
Enter name: Rakesh Rao
Enter age: 19
Enter email: 2303A51851@sru.edu.in
Created: User(name='Rakesh Rao', age=19, email='2303A51851@sru.edu.in')
PS C:\Users\gandr\Documents\3Year-2Sem\AI assistants Coding\AI code> & C:/Users/gandr/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/gandr/Documents/3Year-2Sem/AI assistants Coding/AI code/Assess 6.1.py"

```

Task-4:

Task Description #4 (AI-Based Code Completion for Classes)

Task: Use an AI code completion tool to generate a Python class for managing student details.

Prompt:

```
# Generate a Python class Student with attributes (name, roll number, marks)  
and methods to calculate total and average marks.
```

Code:

```
File Edit Selection View Go Run Terminal Help ← → ⌘ AI code  
EXPLORER CODE Asses 6.1.py ~  
Ass 2-4.py  
Ass 3-4.py  
Ass 6.1.py  
Asses 6.1.py  
Asses 6.1.py  
users_protected.htm  
users_protected.csv  
Code (copy from here):  
← PRINT EVEN NUMBERS IN PYTHON ISNAI  
task 4 - student class & example  
# Task 4: Student class with total and average  
class Student:  
    def __init__(self, name: str, roll: int, marks: list[float]):  
        if not isinstance(name, str):  
            raise ValueError("Name must be a non-empty string.")  
        if not isinstance(roll, int) or roll < 0:  
            raise ValueError("Roll must be a non-negative integer.")  
        if not isinstance(marks, (list, tuple)):  
            raise ValueError("Marks must be a list or tuple of numbers.")  
        for m in marks:  
            if not isinstance(m, (int, float)) or m < 0:  
                raise ValueError("Each mark must be a non-negative number.")  
  
        self.name = name.strip()  
        self.roll = roll  
        self.marks = list(marks)  
  
    def total(self) -> float:  
        return sum(self.marks)  
  
    def average(self) -> float:  
        return self.total() / len(self.marks) if self.marks else 0.0  
  
    def add_marks(self, mark: float):  
        if not isinstance(mark, (int, float)) or mark < 0:  
            raise ValueError("Mark must be a non-negative number.")  
        self.marks.append(mark)  
  
    def __repr__(self):  
        return f"Student(name={self.name}, roll={self.roll}, marks={self.marks})"  
  
# Example usage:  
s = Student("Aayush", 12, [78, 82, 91])  
print("Total:", s.total()) # Total: 251  
print("Average:", s.average()) # Average: 83.666...  
s.add_marks(85)  
print("New average:", s.average())  
# New average: 86.5
```

Output:

```
File Python Terminal Help  
PS C:\Users\gandr\Documents\3Year-2Sem\AI assistants Coding\AI code & C:\Users\gandr\AppData\Local\Programs\Python\Python314\python.exe "c:/Users/gandr/Documents/3Year-2Sem/AI assistants Coding/AI code/assess 6.1.py"  
Total: 251  
Average: 83.66666666666667  
New average: 86.5  
PS C:\Users\gandr\Documents\3Year-2Sem\AI assistants Coding\AI code>
```

Task-5:

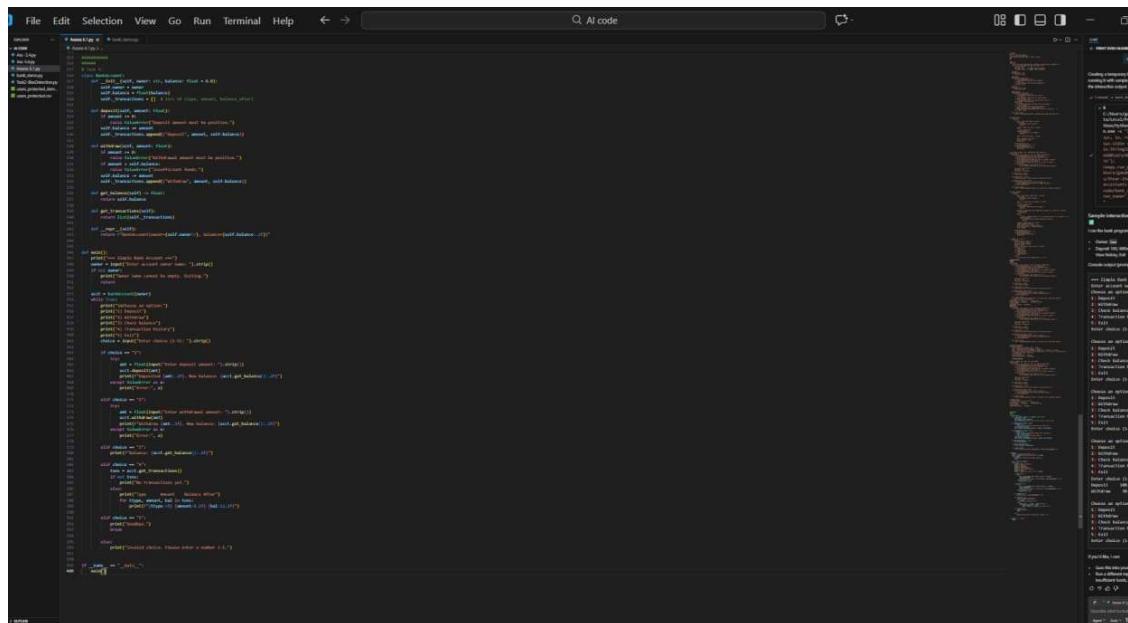
Task Description 5 (AI-Assisted Code Completion Review)

Task: Use an AI tool to generate a complete Python program using classes, loops, and conditionals together.

Prompt:

##Generate a Python program for a simple bank account system using class, loops, and conditional statements.

Code:



```
File Edit Selection View Go Run Terminal Help ← → Q: AI code

# Simple Bank Account System
class BankAccount:
    def __init__(self, owner_name, balance=0.0):
        self.owner_name = owner_name
        self.balance = 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:
            print("Withdrawal amount must be positive")
        elif amount > self.balance:
            print("Insufficient funds")
        else:
            self.balance -= amount
            print(f"Withdrew {amount}. New balance: {self.balance}")

    def get_balance(self):
        return self.balance

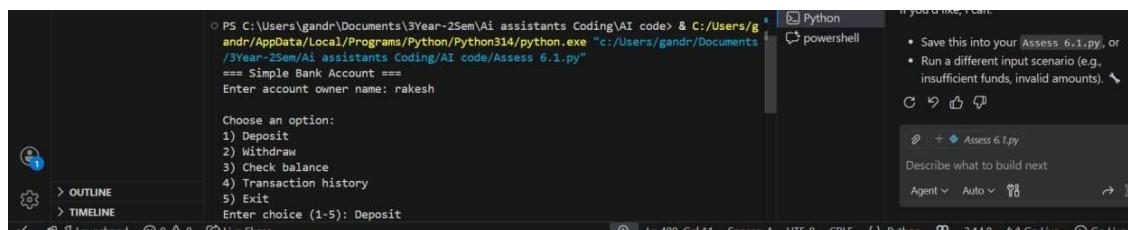
    def transaction(self, amount):
        if amount < 0:
            print("Transaction amount must be positive")
        else:
            self.balance += amount
            print(f"Transaction successful. New balance: {self.balance}")

def main():
    print("Simple Bank Account")
    print("1) Deposit")
    print("2) Withdraw")
    print("3) Check balance")
    print("4) Transaction history")
    print("5) Exit")
    print("Enter choice (1-5):")

    choice = input()
    if choice == "1":
        amount = float(input("Enter amount: "))
        account.deposit(amount)
    elif choice == "2":
        amount = float(input("Enter amount: "))
        account.withdraw(amount)
    elif choice == "3":
        print(f"Current balance: {account.get_balance()}")
    elif choice == "4":
        print("Transaction history not implemented yet")
    elif choice == "5":
        print("Goodbye! Please enter again later.")


if __name__ == "__main__":
    account = BankAccount("John Doe", 1000.0)
    main()
```

Out Put:



```
PS C:\Users\gandr\Documents\3Year-2Sem\AI assistants Coding\AI code & C:/Users/gandr/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/gandr/Documents/3Year-2Sem/AI assistants Coding/AI code/Assess 6.1.py"
== Simple Bank Account ==
Enter account owner name: rakesh

Choose an option:
1) Deposit
2) Withdraw
3) Check balance
4) Transaction history
5) Exit
Enter choice (1-5): Deposit
```

If you'd like, I can:

- Save this into your `Assess 6.1.py`, or
- Run a different input scenario (e.g., insufficient funds, invalid amounts), ↗