

ASSIGNMENT-6.1

2303A51494

Batch-27

Task-1:

Task Description #1 (AI-Based Code Completion for Loops)

Task: Use an AI code completion tool to generate a loop-based

Prompt:

Write a Python program that uses a loop to print all even numbers between 1 and a user-defined integer N (inclusive). The program should clearly demonstrate the loop logic, specify whether a for loop or a while loop is used, and include a sample execution showing the output for a given value of N.

Code and output:

```
1 def print_even_numbers(N):
2     for i in range(1, N + 1):
3         if i % 2 == 0:
4             print(i)
5 print_even_numbers(10)
6
```

Shell ×

```
>>> %Run -c $EDITOR_CONTENT
```

```
2
4
6
8
10
```

```
>>>
```

Identification of loop type used (for or while).

- Loop used: for loop
- A for loop is ideal when the range (1 to N) is known in advance

Explanation:

- An AI code completion tool is used to generate Python code that prints even numbers from **1 to N**.
- A **loop** (for or while) iterates through numbers in the given range.
- A **conditional check** (`number % 2 == 0`) identifies even numbers.
- The program is validated using sample input values to ensure correct output.
- This task demonstrates how AI helps quickly generate correct loop logic with minimal effort.

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.

Prompt:

"Generate Python code that uses a loop and `if-else` conditionals to count the number of even and odd elements in a given list. Include sample input, output, and a brief explanation of the logic."

Code and Output:

```
hpc.py > ...
1  def count_even_odd(numbers):
2      even_count = 0
3      odd_count = 0
4
5      for num in numbers:
6          if num % 2 == 0:
7              even_count += 1
8          else:
9              odd_count += 1
10
11     return even_count, odd_count
12 n = int(input("Enter the number of elements: "))
13 numbers = []
14
15 for i in range(n):
16     num = int(input(f"Enter element {i + 1}: "))
17     numbers.append(num)
18
19 even, odd = count_even_odd(numbers)
20
21 print("Even numbers count:", even)
22 print("Odd numbers count:", odd)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\manis\OneDrive\Desktop\PYTHON TRA> & 'c:\Users\manis\AppData\Local\Microsoft\Windows\Terminal\apps\Python\Python311\python.exe' -c 'import sys; sys.path.append('c:\Users\manis\AppData\Local\Microsoft\Windows\Terminal\apps\Python\Python311\python.exe'); import hpc; hpc.count_even_odd([10, 8, 9, 5, 6])'
```

Enter element 1: 10
Enter element 2: 8
Enter element 3: 9
Enter element 4: 5
Enter element 5: 6
Even numbers count: 3
Odd numbers count: 2
PS C:\Users\manis\OneDrive\Desktop\PYTHON TRA>

Explanation:

1. The user enters how many elements the list will contain.
2. A for loop collects elements dynamically into a list.
3. Another for loop checks each number using an if-else condition.
4. Even and odd counters are updated accordingly.
5. Final counts are displayed.

Loop and Conditional Used

Loop: for loop

Conditional: if-else

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 the user

Expected Output:

- AI-generated class with validation logic.
- Verification of condition handling.
- Test cases for valid and invalid inputs.

Prompt:

"Create a Python class `User` that validates user age and email using conditional statements. Include validation logic and test cases for both valid and invalid inputs."

Code and Output:

```
hpc.py > ...
1 class User:
2     def __init__(self, age, email):
3         self.age = age
4         self.email = email
5
6     def validate_age(self):
7         return self.age >= 18
8
9     def validate_email(self):
10        return "@" in self.email and "." in self.email
11
12    def validate_user(self):
13        return self.validate_age(), self.validate_email()
14
15    age = int(input("Enter age: "))
16    email = input("Enter email: ")
17
18    user = User(age, email)
19    age_valid, email_valid = user.validate_user()
20
21    print("\nValidation Results:")
22    print("Age valid:", age_valid)
23    print("Email valid:", email_valid)
24
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```
sktop\PYTHON TRA\hpc.py'
PS C:\Users\manis\OneDrive\Desktop\PYTHON TRA> c::; cd 'c:\Users\manis\OneDrive\Desktop\PYTHON TRA\Programs\Python\Python313\python.exe' 'c:\Users\manis\.vscode\extensions\ms-python.debugpy-2022.12.0\py\launcher' '61629' '--' 'C:\Users\manis\OneDrive\Desktop\PYTHON TRA\hpc.py'
Enter age: 21
Enter email: pavani@1322@gmail.com

Validation Results:
Age valid: True
Email valid: True
PS C:\Users\manis\OneDrive\Desktop\PYTHON TRA>
```

Explanation:

1. The User class stores age and email as attributes.
2. Separate methods validate age and email using conditional statements.
3. Test cases demonstrate correct handling of valid and invalid inputs.

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.

Expected Output:

- AI-generated class code.
- Verification of correctness and completeness of the class structure.
- Minor manual improvements (if needed) with justification.

Prompt:

"Create a Python class `Student` with attributes `name`, `roll_number`, and `marks`. Implement methods to calculate total marks and average marks, and include a short test example to verify correctness."

Code and Output:

```
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

    def calculate_total(self):
        return sum(self.marks)

    def calculate_average(self):
        if len(self.marks) == 0:
            return 0
        return self.calculate_total() / len(self.marks)

name = input("Enter student name: ")
roll_number = input("Enter roll number: ")

n = int(input("Enter number of subjects: "))
marks = []

for i in range(n):
    mark = float(input(f"Enter marks for subject {i + 1}: "))
    marks.append(mark)

student = Student(name, roll_number, marks)

print("\nStudent Details")
print("Name:", student.name)
print("Roll Number:", student.roll_number)
print("Total Marks:", student.calculate_total())
print("Average Marks:", student.calculate_average())
```

```

C:\Users\manis\OneDrive\Desktop\PYTHON TRA\ipet.py
Enter student name: pavani
Enter roll number: 1494
Enter number of subjects: 3
Enter marks for subject 1: 98
Enter marks for subject 2: 95
Enter marks for subject 3: 94

Student Details
Name: pavani
Roll Number: 1494
Total Marks: 287.0
Average Marks: 95.66666666666667
PS C:\Users\manis\OneDrive\Desktop\PYTHON TRA>

```

Explanation:

- The Student class stores student name, roll number, and marks.
- User inputs details and marks dynamically using a loop.
- `calculate_total()` adds all marks.
- `calculate_average()` computes the average safely.
- The program displays total and average marks correctly.

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.

Expected Output:

- Complete AI-generated program.
- Identification of strengths and limitations of AI suggestions.
- Reflection on how AI assisted coding productivity.

Prompt:

Prompt:

"Create a Python program for a simple bank account system using classes, loops, and conditional statements."

Code and Output:

```
py > ...
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print("Amount deposited successfully.")
        else:
            print("Invalid deposit amount.")

    def withdraw(self, amount):
        if amount <= self.balance and amount > 0:
            self.balance -= amount
            print("Withdrawal successful.")
        else:
            print("Insufficient balance or invalid amount.")

    def check_balance(self):
        print("Current Balance:", self.balance)

name = input("Enter account holder name: ")
account = BankAccount(name)

while True:
    print("\n--- Bank Menu ---")
    print("1. Deposit")
    print("2. Withdraw")
    print("3. Check Balance")
    print("4. Exit")
```



```
33
34     if choice == "1":
35         amount = float(input("Enter deposit amount: "))
36         account.deposit(amount)
37
38     elif choice == "2":
39         amount = float(input("Enter withdrawal amount: "))
40         account.withdraw(amount)
41
42     elif choice == "3":
43         account.check_balance()
44
45     elif choice == "4":
46         print("Thank you for using the bank system.")
47         break
48
49     else:
50         print("Invalid choice. Please try again.")
51
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python De

```
PS C:\Users\manis\OneDrive\Desktop\PYTHON TRA> c:: cd 'c:\Users\manis\OneDrive\Desktop\PYTHON TRA'; & 'c:\Users\ma
Python313\python.exe' 'c:\Users\manis\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugp
s\manis\OneDrive\Desktop\PYTHON TRA\hpc.py'
2. Withdraw
3. Check Balance
4. Exit
Enter your choice: 1
Enter deposit amount: 5000
Amount deposited successfully.

--- Bank Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Enter your choice: 4
Thank you for using the bank system.
Exiting completed.
```

Explanation:

- The program uses a BankAccount class to store account details and balance.
- Conditional statements handle deposit, withdrawal, and validation.
- A loop creates a menu-driven system for repeated user actions.
- AI assistance helped quickly build the program structure and logic.
- Human review is needed to improve security and error handling.