

Name: M.Bharath

Roll no :2303A52060

Batch no:39

Prompt Engineering Task Report:

Experiment 6: AI-Based Code Completion: Working with suggestions for classes, loops, conditionals

# Task Description 1:

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

Prompt:

#Generate code for even numbers between 1 to n using loops in Python

#user defined input

Explanation :

As the prompt becomes more specific, the calculator logic improves by handling operators correctly and preventing division by zero.

Code:

```
def print_even_numbers(n):
    for i in range(1, n+1):
        if i % 2 == 0:
            print(i)
#user defined input
n = int(input("Enter a number: "))
print_even_numbers(n)
```

Output:

```
Enter a number: 12
2
4
6
8
10
12

c:\Users\dell\OneDrive\B-
```

## Task Description #2

### Prompt:

“Generate Python code to count how many numbers in a list are even and odd.”

### Explanation:

AI-generated code using loop and if condition.

- Correct count validation.
- Explanation of logic flow

### Code:

```
def count_even_odd(numbers):
    even_count = 0
    odd_count = 0
    for number in numbers:
        if number % 2 == 0:
            even_count += 1
        else:
            odd_count += 1
    return even_count, odd_count

# user defined input take only one number and print the output as oddnumbers and even numbers
n = int(input("Enter a number: "))
numbers = list(range(1, n+1))
even_count, odd_count = count_even_odd(numbers)
print(f"Even numbers count: {even_count}")
print(f"Odd numbers count: {odd_count}")
```

output:

```
DESKTOP\NILESH-DESKTOP-V1Z7A
Enter a number: 14
Even numbers count: 7
Odd numbers count: 7
```

### Task Description #3

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

Prompt:

“Generate a Python class User that validates age and email using conditional statements.”

Explanation:

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

Code:

```
class User:
    def __init__(self, age, email):
        self.age = age
        self.email = email

    def is_valid_age(self):
        return self.age >= 18

    def is_valid_email(self):
        return "@" in self.email and "." in self.email.split("@")[-1]
# user defined input
age = int(input("Enter your age: "))
email = input("Enter your email: ")
user = User(age, email)
if user.is_valid_age():
    print("Valid age")
else:
    print("Invalid age")
print("Valid email" if user.is_valid_email() else "Invalid email")
```

output:

```
C:\Users\bharath\OneDrive\Pictures\Desktop\AI Assistant\25
Enter your age: 18
Enter your email: bharath@gmail.com
Valid age
Valid email
```

Task4: Use an AI code completion tool to generate a Python class for managing student details.

Prompt:

```
#create a class of Student with attributes name, rollnumber, and marks. Implement a method to
display student details.
#use the conditional statements to check pass or fail
# calculate the all marks take the pass marks as 36 of user defined input
#take atleast 5 marks calculate total marks average marks
```

Explanation:

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

Code:

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

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.rollnumber}")
        print(f"Marks: {self.marks}")

    #use the conditional statements to check pass or fail
    def is_passed(self):
        return all(mark >= 36 for mark in self.marks)

# user defined input
name = input("Enter student name: ")
```

```

rollnumber = input("Enter roll number: ")
#take atleast 5 marks calculate total marks average marks
marks = []
for i in range(5):
    marks.append(int(input(f"Enter marks for subject {i+1}: ")))
student = Student(name, rollnumber, marks)
student.display_details()
if student.is_passed():
    print("Status: Passed")
else:
    print("Status: Failed")
total_marks = sum(marks)
average_marks = total_marks / len(marks)
print(f"Total Marks: {total_marks}")
print(f"Average Marks: {average_marks}")

```

output:

```

neDrive\Pictures\Desktop\Ai Assistant\2303A52060_Ai_Assistant_Ass_6.py"
Enter student name: bharath
Enter roll number: 2303A52060
Enter marks for subject 1: 40
Enter marks for subject 2: 50
Enter marks for subject 3: 90
Enter marks for subject 4: 100
Enter marks for subject 5: 80
Name: bharath
Roll Number: 2303A52060
Marks: [40, 50, 90, 100, 80]
Status: Passed
Total Marks: 360
Average Marks: 72.0

```

Task-5:

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

Prompt:

```
#Generate a python code for a simple bank account using class loops and conditional statements
```

Explanation:

Complete AI-generated program.

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

Code:

```
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(f"Deposited: {amount}")  
        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}")  
            else:  
                print("Insufficient balance")  
        else:  
            print("Withdrawal amount must be positive")  
  
    def display_balance(self):  
        print(f"Account Holder: {self.account_holder}")  
        print(f"Balance: {self.balance}")  
  
# user defined input  
account_holder = input("Enter account number: ")  
account = BankAccount(account_holder)  
while True:  
    print("\nOptions:")  
    print("1. Deposit")  
    print("2. Withdraw")  
    print("3. Display Balance")  
    print("4. Exit")  
    choice = input("Choose an option (1-4): ")  
    if choice == '1':  
        amount = float(input("Enter amount to deposit: "))  
        account.deposit(amount)  
    elif choice == '2':
```

```
amount = float(input("Enter amount to withdraw: "))
account.withdraw(amount)
elif choice == '3':
    account.display_balance()
elif choice == '4':
    print("Exiting...")
    break
else:
    print("Invalid option, please try again.")
```

output:

```
Options:
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Choose an option (1-4): 1
Enter amount to deposit: 1000
Deposited: 1000.0

Options:
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Choose an option (1-4): 2
Enter amount to withdraw: 800
Withdrew: 800.0

Options:
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
Choose an option (1-4): 3
Account Holder: 6448493
Balance: 200.0

Options:
1. Deposit
2. Withdraw
3. Display Balance
4. Exit
```