SR UNIVERSITY
AI ASSISTED CODING
ASSIGNMENT-04

NAME:SHAIK FAHEEM
HTNO: 2503A51L39

TASK1

TASK1 DESCRIPTION :- Auto-Complete a Python Class for Bank Account • Write a class definition comment and start the constructor for a class called Bank Account with account_holder and balance attributes. Use GitHub Copilot to auto-complete the rest of the class, including methods to deposit, withdraw, and display balance.

PROMPT:-Write a Python class called BankAccount with a class definition comment, a constructor that takes account_holder and balance, and methods deposit(self, amount), withdraw(self, amount) with error handling for insufficient funds, and display_balance(self); provide the complete implementation with example.

SR UNIVERSITY
AI ASSISTED CODING
ASSIGNMENT-04

NAME:SHAIK FAHEEM
HTNO: 2503A51L39

CODE

aiii.py

C: > Users > Shaik Faheem > Desktop > aiii.py > ...

```python
class BankAccount:
    def __init__(self, account_holder, balance=0.0):
        """
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        """Deposit money into the account."""
        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):
        """Withdraw money from the account if funds are sufficient."""
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew: {amount}. New balance: {self.balance}")
        else:
            print("Insufficient funds or invalid withdrawal amount.")

    def display_balance(self):
        """Display the current balance."""
        print(f"Account Holder: {self.account_holder}, Balance: {self.balanc


# ----------- Execution Part -----------
# Create a BankAccount object
account = BankAccount("Alice", 1000)

# Perform some operations
account.display_balance()   # Show initial balance
account.deposit(500)        # Deposit 500
account.withdraw(200)       # Withdraw 200
account.withdraw(2000)      # Try withdrawing too much
```

SR UNIVERSITY
AI ASSISTED CODING
ASSIGNMENT-04

NAME:SHAIK FAHEEM
HTNO: 2503A51L39

OUTPUT

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    ...    Filter    Code

[Done] exited with code=0 in 0.13 seconds

[Running] python -u "c:\Users\Shaik Faheem\Desktop\aiii.py"
Account Holder: Alice, Balance: 1000
Deposited: 500. New balance: 1500
Withdrew: 200. New balance: 1300
Insufficient funds or invalid withdrawal amount.
Account Holder: Alice, Balance: 1300

[Done] exited with code=0 in 0.124 seconds
```

TASK2

TASK2 DESCRIPTION:- Auto-Complete a For Loop to Sum Even Numbers in a List Write a comment and the initial line of a loop to iterate over a list. Allow GitHub Copilot to complete the logic to sum all even numbers in the list.

PROMPT:-Write a comment and the initial line of a Python loop to iterate over a list, then let GitHub Copilot auto-complete the logic to sum all even numbers in the list and implement with example**.**

CODE

```
C: > Users > Shaik Faheem > Desktop > aiii.py > ...
# Loop through the list to check and sum all even numbers
numbers = [1, 2, 3, 4, 5, 6, 10, 13]
total = 0
for num in numbers:
    if num % 2 == 0:
        total += num

print("Sum of even numbers:", total)
```

SR UNIVERSITY
AI ASSISTED CODING
ASSIGNMENT-04

NAME:SHAIK FAHEEM
HTNO: 2503A51L39

OUTPUT

```
[Running] python -u "c:\Users\Shaik Faheem\Desktop\aiii.py"
Sum of even numbers: 22
```

TASK3

TASK3 DESCRIPTION:- Auto-Complete Conditional Logic to Check Age Group Start a function that takes age as input and returns whether the person is a child, teenager,adult, or senior using if-elif-else. Use Copilot to complete the conditionals

PROMPT:-Generate a python function that takes age as input and returns whether the person is a child, teenager,adult, or senior using if-elif-else.implement with clear example.

SR UNIVERSITY
AI ASSISTED CODING
ASSIGNMENT-04

NAME:SHAIK FAHEEM
HTNO: 2503A51L39

CODE

aiii.py

C: > Users > Shaik Faheem > Desktop > aiii.py > categorize_age

```python
def categorize_age(age):
    if age < 13:
        return "Child"
    elif age < 20:
        return "Teenager"
    elif age < 60:
        return "Adult"
    else:
        return "Senior"


# Example list of ages
ages = [10, 16, 30, 70, 12, 18, 45, 65, 8]

# Dictionary to count categories
counts = {"Child": 0, "Teenager": 0, "Adult": 0, "Senior": 0}

for age in ages:
    category = categorize_age(age)
    counts[category] += 1

# Print result
for category, count in counts.items():
    print(f"{category}: {count} members")

```

OUTPUT

```
[Running] python -u "c:\Users\Shaik Faheem\Desktop\aiii.py"
Child: 3 members
Teenager: 2 members
Adult: 2 members
```

SR UNIVERSITY
AI ASSISTED CODING
ASSIGNMENT-04

NAME:SHAIK FAHEEM
HTNO: 2503A51L39

TASK4

TASK4 DESCRIPTION:- Auto-Complete a While Loop to Reverse Digits of a Number Write a comment and start a while loop to reverse the digits of a number. Let Copilot complete the loop logic.

PROMPT:-Generate a Python comment and start a while loop to reverse the digits of a number. Then use GitHub Copilot to auto-complete the loop logic and implement it with example

CODE

C: > Users > Shaik Faheem > Desktop > aiii.py > ...

```python
# Program to reverse the digits of a number using a while loop

num = 12345
reversed_num = 0

# Start while loop to extract digits
while num > 0:
    digit = num % 10          # Get the last digit
    reversed_num = reversed_num * 10 + digit  # Append digit to reversed num
    num = num // 10           # Remove the last digit

print("Reversed number:", reversed_num)
```

OUTPUT

SR UNIVERSITY
AI ASSISTED CODING
ASSIGNMENT-04

NAME:SHAIK FAHEEM
HTNO: 2503A51L39

```
Done] exited with code=0 in 0.189 seconds

Running] python -u "c:\Users\Shaik Faheem\Desktop\aiii.py"
Reversed number: 54321
```

TASK5

TASK5 DESCRIPTION:- Auto-Complete Class with Inheritance (Employee → Manager) Begin a class Employee with attributes name and salary. Then, start a derived class Manager that inherits from Employee and adds a department. Let GitHub Copilot complete the methods and constructor chaining

PROMPT:- Write the start of a Python class Employee with attributes name and salary, then begin a derived class Manager that inherits from Employee and adds a department attribute. Let GitHub Copilot auto-complete the constructors and any necessary methods, including proper constructor chaining implement with example.

SR UNIVERSITY
AI ASSISTED CODING
ASSIGNMENT-04

NAME:SHAIK FAHEEM
HTNO: 2503A51L39

CODE

C: > Users > Shaik Faheem > Desktop > aiii.py > ...

```python
# Base class Employee
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary

    def display_info(self):
        print(f"Employee Name: {self.name}")
        print(f"Salary: {self.salary}")


# Derived class Manager (inherits from Employee)
class Manager(Employee):
    def __init__(self, name, salary, department):
        # Constructor chaining using super()
        super().__init__(name, salary)
        self.department = department

    def display_info(self):
        # Call parent method first
        super().display_info()
        print(f"Department: {self.department}")


# Example usage
emp1 = Employee("Alice", 50000)
emp1.display_info()

print("\n--- Manager Info ---")
mgr1 = Manager("Bob", 75000, "IT")
mgr1.display_info()
```

SR UNIVERSITY
AI ASSISTED CODING
ASSIGNMENT-04

NAME:SHAIK FAHEEM
HTNO: 2503A51L39

OUTPUT

```
[Running] python -u "c:\Users\Shaik Faheem\Desktop\ai11.py"
Employee Name: Alice
Salary: 50000

--- Manager Info ---
Employee Name: Bob
Salary: 75000
Department: IT
```

OBSERVATION:- I observed how GitHub Copilot can be effectively used to auto-complete Python code when given the correct prompts. By providing only the initial structure such as a class definition, function header, or loop starter, Copilot was able to generate complete implementations with logical flow.

- **In Task 1**, I noticed that Copilot could generate a full Python class with constructor, methods, and proper error handling just from a descriptive prompt.

- **In Task 2**, it correctly completed the loop logic to filter and sum even numbers, showing its ability to understand conditional iteration.

- **In Task 3**, the function for age classification highlighted how Copilot handles nested conditionals and returns meaningful results.

- **In Task 4,** the while loop to reverse digits showed Copilot's capability to handle mathematical logic inside loops.

- **In Task 5,** I observed how Copilot implements object-oriented concepts like inheritance and constructor chaining effectively.