

Assignment Report

Course: Ai Assisted Coding

- **Student Name:** B.yashwanmth
- **Roll Number / ID:** 2403A51266
- **Batch :** 24BTCAICSB12

Date of Submission: 21st August 2025

Table of Contents

1. **Task 1** – Bank Account Class
 2. **Task 2** – For Loop to Sum Even Numbers
 3. **Task 3** – Age Group Classification
 4. **Task 4** – While Loop to Reverse Digits
 5. **Task 5** – Employee → Manager Inheritance
 6. **Conclusion**
-

Task 1: Auto-Complete a Python Class for Bank Account

Task Description:

Create a BankAccount class with attributes `account_holder` and `balance`. Add methods to deposit, withdraw, and display the balance.

◇ Python Code

✓ Task 1: Auto-Complete a Python Class for Bank Account

```
# This class represents a bank account
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        """Add money to the account"""
        self.balance += amount
        print(f"Deposited: {amount}")

    def withdraw(self, amount):
        """Withdraw money if sufficient balance exists"""
        if amount > self.balance:
            print("Insufficient funds!")
        else:
            self.balance -= amount
            print(f"Withdrew: {amount}")

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

# Sample Usage for Task 1
my_account = BankAccount("Alice", 1000)
my_account.display_balance()
my_account.deposit(500)
my_account.display_balance()
my_account.withdraw(200)
my_account.display_balance()
my_account.withdraw(2000)
my_account.display_balance()
```

◇ Output

```
my_account.display_balance()

⇒ Account Holder: Alice, Balance: 1000
   Deposited: 500
   Account Holder: Alice, Balance: 1500
   Withdrew: 200
   Account Holder: Alice, Balance: 1300
   Insufficient funds!
   Account Holder: Alice, Balance: 1300
```

Task 2: Auto-Complete a For Loop to Sum Even Numbers in a List

Task Description:

Iterate a list, check if the number is even, and calculate total of even numbers.

◇ Python Code

✓ Task 2: Auto-Complete a For Loop to Sum Even Numbers in a List

```
[5] # Sum all even numbers in a list
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum = 0
for number in numbers:
    if number % 2 == 0:
        even_sum += number

print(f"The list is: {numbers}")
print(f"The sum of even numbers is: {even_sum}")
```

◇ Explanation

- Loops over the list.
- Checks `num % 2 == 0`.
- Adds even numbers to the accumulator.

◇ Output

```
➡ The list is: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
   The sum of even numbers is: 30
```

Task 3: Auto-Complete Conditional Logic to Check Age Group

Task Description:

Design a function to classify age into categories.

◇ Python Code

✓ Task 3: Auto-Complete Conditional Logic to Check Age Group

```
▶ # Check the age group of a person
def age_group(age):
    if age < 13:
        return "Child"
    elif 13 <= age < 20:
        return "Teenager"
    elif 20 <= age < 65:
        return "Adult"
    else:
        return "Senior"

# Sample Usage for Task 3
print(f"Age 5: {age_group(5)}")
print(f"Age 15: {age_group(15)}")
print(f"Age 45: {age_group(45)}")
print(f"Age 70: {age_group(70)}")
```

◇ Explanation

- if-elif-else structure handles classification.
- Ranges: <13, <20, <60, else Senior.

◇ Output

```
➡ Age 5: Child
   Age 15: Teenager
   Age 45: Adult
   Age 70: Senior
```

Task 4: Auto-Complete a While Loop to Reverse Digits of a Number

Task Description:

Reverse an integer using while loop.

◇ Python Code

✓ Task 4: Auto-Complete a While Loop to Reverse Digits of a Number

```
[ ] # Reverse the digits of a number
    number = 1234
    reversed_number = 0
    original_number = number # Store the original number for printing

    while number > 0:
        digit = number % 10
        reversed_number = reversed_number * 10 + digit
        number //= 10

    print(f"Original number: {original_number}")
    print(f"Reversed number: {reversed_number}")
```

◇ Explanation

- % 10 extracts last digit.
- Builds reversed number iteratively.
- Number shrinks using // 10.

◇ Output

```
➡ Original number: 1234
   Reversed number: 4321
```

Task 5: Auto-Complete Class with Inheritance (Employee → Manager)

Task Description:

Implement class inheritance where Manager extends Employee.

◇ Python Code

Task 5: Auto-Complete Class with Inheritance (Employee → Manager)

```
[8] # Base class for Employee
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary

    def display_employee(self):
        print(f>Name: {self.name}, Salary: {self.salary}")

# Derived class for Manager, inheriting from Employee
class Manager(Employee):
    def __init__(self, name, salary, department):
        super().__init__(name, salary)
        self.department = department

    def display_manager(self):
        print(f>Name: {self.name}, Salary: {self.salary}, Dept: {self.department}")

# Sample Usage for Task 5
emp = Employee("Jane", 60000)
emp.display_employee()

mgr = Manager("John", 50000, "IT")
mgr.display_manager()
```

◇ Explanation

- Manager reuses Employee constructor using super().
- Adds department.
- Overrides display() for extended output.

◇ Output

```
→ Name: Jane, Salary: 60000
   Name: John, Salary: 50000, Dept: IT
```

+