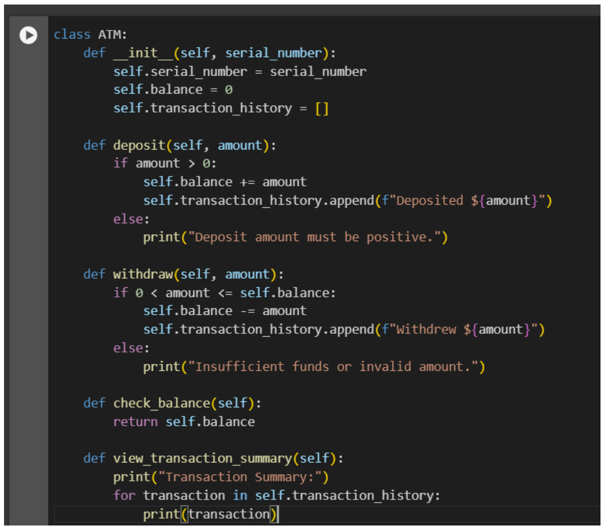
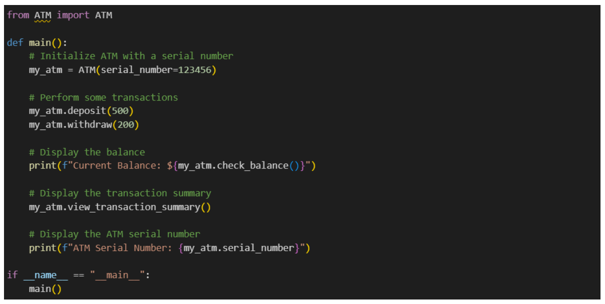
Activity Name #1 - Introduction to Object-Oriented Programming	
Cabilan, Cyruz Peter, C	09/16/2024
CPE 009B/CPE21S4	Ms.Maria Rizette Sayo

1. Modify the ATM.py program and add the constructor function.



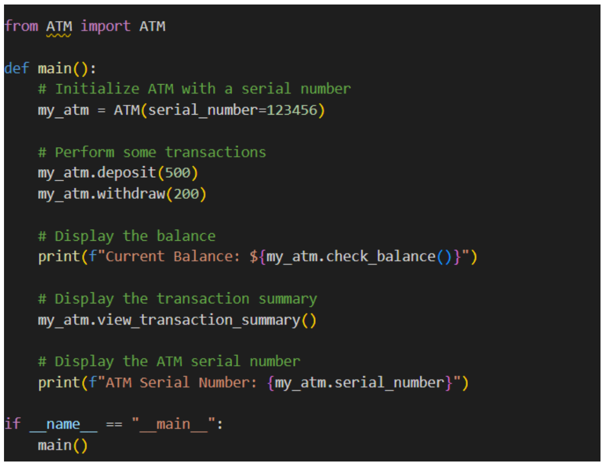
```python
class ATM:
    def __init__(self, serial_number):
        self.serial_number = serial_number
        self.balance = 0
        self.transaction_history = []

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            self.transaction_history.append(f"Deposited ${amount}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            self.transaction_history.append(f"Withdrew ${amount}")
        else:
            print("Insufficient funds or invalid amount.")

    def check_balance(self):
        return self.balance

    def view_transaction_summary(self):
        print("Transaction Summary:")
        for transaction in self.transaction_history:
            print(transaction)
```

2. Modify the main.py program and initialize the ATM machine with any integer serial number combination and display
the serial number at the end of the program



```python
from ATM import ATM

def main():
    # Initialize ATM with a serial number
    my_atm = ATM(serial_number=123456)

    # Perform some transactions
    my_atm.deposit(500)
    my_atm.withdraw(200)

    # Display the balance
    print(f"Current Balance: ${my_atm.check_balance()}")

    # Display the transaction summary
    my_atm.view_transaction_summary()

    # Display the ATM serial number
    print(f"ATM Serial Number: {my_atm.serial_number}")

if __name__ == "__main__":
    main()
```

3.Modify the ATM.py program and add the view_transactionsummary() method. The method should display all the
transaction made in the ATM object.



```python
from ATM import ATM

def main():
    # Initialize ATM with a serial number
    my_atm = ATM(serial_number=123456)

    # Perform some transactions
    my_atm.deposit(500)
    my_atm.withdraw(200)

    # Display the balance
    print(f"Current Balance: ${my_atm.check_balance()}")

    # Display the transaction summary
    my_atm.view_transaction_summary()

    # Display the ATM serial number
    print(f"ATM Serial Number: {my_atm.serial_number}")

if __name__ == "__main__":
    main()
```

Questions

1. What is a class in Object-Oriented Programming?

In Object-Oriented Programming, a class is like a blueprint for creating objects. It defines what data the objects will hold and what actions they can perform. By using classes, you can keep your code organized and reusable, making it easier to build and manage complex systems.

2. Why do you think classes are being implemented in certain programs while some are

sequential(line-by-line)?

Classes are like blueprints that help structure and organize complex code, making it easier to understand and reuse across different parts of a project. They are particularly useful for larger programs because they keep code modular and maintainable. For simpler tasks or quick solutions, though, sequential programming can be more straightforward and efficient.

3. How is it that there are variables of the same name such account_firstname and account_lastname that exist but have different values?

Variables with the same name can have different values if they are in different parts of the code, like inside different functions or classes. Each object in a program can have its own version of the variable, so the same name can mean different things in different objects. Also, variables can have different values if they're used in separate sections of the code, even if they share the same name.

4. Explain the constructor functions role in initializing the attributes of the class? When does the Constructor function execute or when is the constructor function called?

The constructor function, **__init__** sets up the initial values for an object's attributes when you create a new instance of a class. It runs automatically as soon as the object is created, so you don’t need to start it yourself. This helps ensure that each new object begins with the right setup.

5. Explain the benefits of using Constructors over initializing the variables one by one in the main

program?

Constructors automatically set up all the necessary attributes when you create a new object, so you don’t have to manually assign each one in the main program. This keeps your main code tidy and focused on the bigger picture. Plus, if you need to change how things are set up, you only need to update the constructor, not every place where the object is used.

Conclusion:

In Object-Oriented Programming, classes act like blueprints for creating and managing objects bundling data and functions together. They are great for larger programs because they help keep things organized and make it easier to reuse code Variables with the same name can have different values depending on where they are used, like in different functions or objects. The __init__ constructor makes setting up new objects easier by automatically initializing their values so you don’t have to do it manually. Overall constructors keep your code cleaner and more manageable, making it simpler to update and maintain.