Activity Name #1 - Introduction to Object-Oriented Programming	
Magistrado, Aira Pauleen M.	09/15/2024
BSCPE/CPE21S4	Maria Rizette Sayo

Creating classes:

```
"""
Accounts.py
"""

class Accounts():
    account_number = 0
    account_firstname = ""
    account_lastname = ""
    current_balance = 0.0
    address = ""
    email = ""

    def update_address(self, new_address):
        self.address = new_address

    def update_email(self, new_email):
        self.email = new_email
```

```
"""
ATM.py
"""
class ATM():
    serial_number = 0

    def deposit(self, account, amount):
        account.current_balance = account.current_balance + amount
        print("Deposit Complete")

    def withdraw(self,account,amount):
        account.current_balance = account.current_balance - amount
        print("Withdraw Complete")

    def check_currentbalance(self, account):
        print(account.current_balance)
```

Creating Instances of Classes:

```python
"""
main.py
"""
import Accounts
import ATM

Account1 = Accounts.Accounts()

print("Account 1")
Account1.account_firstname = "Royce"
Account1.account_lastname = "Chua"
Account1.current_balance = 1000
Account1.address = "Silver Street Quezon City"
Account1.email = "roycechua123@gmail.com"

print(Account1.account_firstname)
print(Account1.account_lastname)
print(Account1.current_balance)
print(Account1.address)
print(Account1.email)

print()

Account2 = Accounts.Accounts()
Account2.account_firstname = "John"
Account2.account_lastname = "Doe"
Account2.current_balance = 2000
Account2.address = "Gold Street Quezon City"
Account2.email = "johndoe@gmail.com"

print("Account 2")
print(Account2.account_firstname)
print(Account2.account_lastname)
print(Account2.current_balance)
print(Account2.address)
print(Account2.email)
print()

ATM1 = ATM.ATM()
ATM1.deposit(Account1, 500)
ATM1.check_currentbalance(Account1)

ATM1.deposit(Account2, 300)
ATM1.check_currentbalance(Account2)
```

Create the Constructor in each Class:

```
Account 1
Royce
Chua
1000
Silver Street Quezon City
roycechua123@gmail.com

Account 2
John
Doe
2000
Gold Street Quezon City
johndoe@gmail.com

Deposit Complete
1500
Deposit Complete
2300
```

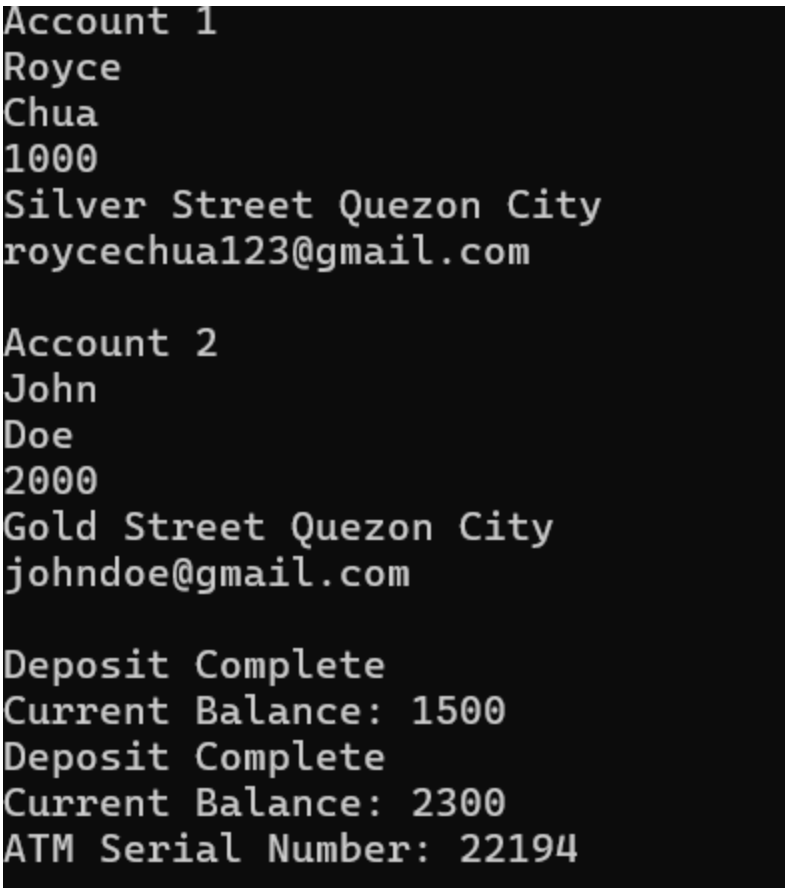
Tasks

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

```
class ATM:
    def __init__(self, serial_number):
        self.serial_number = serial_number

    def deposit(self, account, amount):
        account.current_balance = account.current_balance + amount
        print("Deposit Complete")

    def withdraw(self, account, amount):
        account.current_balance = account.current_balance - amount
        print("Withdraw Complete")

    def check_currentbalance(self, account):
        print("Current Balance: {account.current_balance}")
```

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
import Accounts
import ATM

Account1 = Accounts.Accounts(
    account_number=123456,
    account_firstname="Royce",
    account_lastname="Chua",
    current_balance=1000,
    address="Silver Street Quezon City",
    email="roycechua123@gmail.com",
)

print("Account 1")
print(Account1.account_firstname)
print(Account1.account_lastname)
print(Account1.current_balance)
print(Account1.address)
print(Account1.email)
print()

Account2 = Accounts.Accounts(
    account_number=654321,
    account_firstname="John",
    account_lastname="Doe",
    current_balance=2000,
    address="Gold Street Quezon City",
    email="johndoe@gmail.com",
)

print("Account 2")
print(Account2.account_firstname)
print(Account2.account_lastname)
print(Account2.current_balance)
print(Account2.address)
print(Account2.email)
print()

ATM1 = ATM.ATM(serial_number=22194)
ATM1.deposit(Account1, 500)
ATM1.check_currentbalance(Account1)

ATM1.deposit(Account2, 300)
ATM1.check_currentbalance(Account2)

print(f"ATM Serial Number: {ATM1.serial_number}")
```



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

```
import Accounts
import ATM

Account1 = Accounts.Accounts(
    account_number=123456,
    account_firstname="Royce",
    account_lastname="Chua",
    current_balance=1000,
    address="Silver Street Quezon City",
    email="roycechua123@gmail.com",
)

print("Account 1")
print(Account1.account_firstname)
print(Account1.account_lastname)
print(Account1.current_balance)
print(Account1.address)
print(Account1.email)
print()

Account2 = Accounts.Accounts(
    account_number=654321,
    account_firstname="John",
    account_lastname="Doe",
    current_balance=2000,
    address="Gold Street Quezon City",
    email="johndoe@gmail.com",
)

print("Account 2")
print(Account2.account_firstname)
print(Account2.account_lastname)
print(Account2.current_balance)
print(Account2.address)
print(Account2.email)
print()

ATM1 = ATM.ATM(serial_number=22194)
ATM1.deposit(Account1, 500)
ATM1.check_currentbalance(Account1)

ATM1.deposit(Account2, 300)
ATM1.check_currentbalance(Account2)

ATM1.view_transactionsummary()

print(f"ATM Serial Number: {ATM1.serial_number}")
```

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

    def deposit(self, account, amount):
        account.current_balance += amount
        transaction = f"Deposited money: {amount} | New Balance is: {account.current_balance}"
        self.transactions.append(transaction)
        print("Deposit Complete")

    def withdraw(self, account, amount):
        account.current_balance -= amount
        transaction = f"Withdrawed money: {amount} | New Balance is: {account.current_balance}"
        self.transactions.append(transaction)
        print("Withdraw Complete")

    def check_currentbalance(self, account):
        print(f"Current Balance: {account.current_balance}")

    def view_transactionsummary(self):
        print("Transaction Summary:")
        for transaction in self.transactions:
            print(transaction)
```

```
Account 1
Royce
Chua
1000
Silver Street Quezon City
roycechua123@gmail.com

Account 2
John
Doe
2000
Gold Street Quezon City
johndoe@gmail.com

Deposit Complete
Current Balance: 1500
Deposit Complete
Current Balance: 2300
Transaction Summary:
Deposited money: 500 | New Balance is: 1500
Deposited money: 300 | New Balance is: 2300
ATM Serial Number: 22194
```

Questions:

1. What is a class in Object-Oriented Programming?
   A class in Object-Oriented Programming(OOP) is often compared to a blueprint or building block. It is a detailed description for creating objects and attributes.
2. Why do you think classes are being implemented in certain programs while some are sequential(line-by-line)?
   Classes are used for more complex programs that can be reused and are more organized when compared to sequential/line-by-line, which is used for simpler programs and tasks.
3. How is it that there are variables of the same name such account_firstname and account_lastname that exist but have different values?
   They may have the same attribute, but the values they hold differ because the variables stored in them are different.
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 is called __init__. It assigns values to an object's attributes and ensures that new objects have the required data when they are created. It is called automatically whenever a new class is created.
5. Explain the benefits of using Constructors over initializing the variables one by one in the main program?
   Using constructors on the Accounts and ATM helped to organize the code making it easier to read and understand. It also sets up all of the necessary details when a new object is created..

Conclusion:
Classes are the logical grouping of data and functions. Constructors improve the code's organization and readability. When you create an object, they ensure that the attributes are correctly initialized from the beginning.