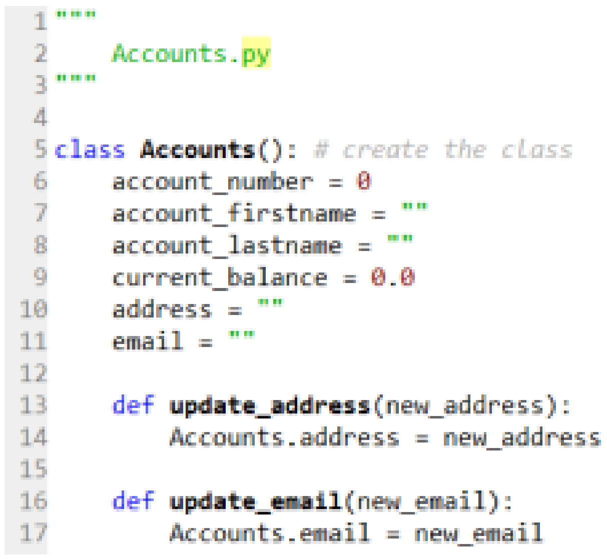
Laboratory Activity No. 1
Introduction to Object-Oriented Programming
Course Code: CPE009B **Program:** BSCPE
Course Title: Object-Oriented Programming **Date Performed:**
Section: Date Submitted: Aug 28, 2024
Name: Instructor: Engr. Maria Rizette Sayo
1. Objective(s):
This activity aims to familiarize students with the concepts of Object-Oriented Programming
2. Intended Learning Outcomes (ILOs):
The students should be able to: 2.1 Identify the possible attributes and methods of a given object 2.2 Create a class using the Python language 2.3 Create and modify the instances and the attributes in the instance.
3. Discussion:
Object-Oriented Programming (OOP) is an approach to programming that views the world and systems as consisting of objects that relate and interact with each other. This involves identifying the characteristics that describe the object which are known as the Attributes of the object. Furthermore, it also deals with identifying the possible capabilities or actions that an object is able to do which are called Methods. An object is simply composed of Attributes and Methods wherein Attributes are variables that hold the information describing the object and Methods are functions which allow the object to perform its defined capabilities/actions. A UML Class Diagram is used to formally represent the collection of Attributes and Methods. An example is given below considering a simple banking system. Accounts ATM + account_number: int + serial_number: int + account_firstname: string + account_lastname: string + current_balance: float + address: string + deposit(account: Accounts, amount: int) + email: string + widthdraw(account: Accounts, amount: int) + update_address(new_address: string) + check_currentbalance(account: Accounts) + update_email(new_email: string) + view_transactionsummary()
4. Materials and Equipment:

Desktop Computer with Anaconda Python
Windows Operating System

5. Procedure:

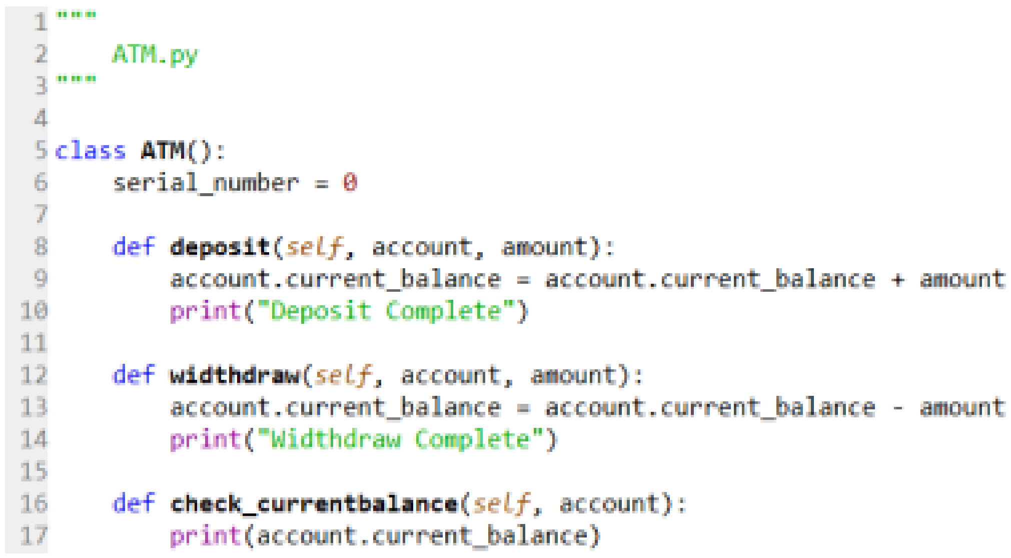
Creating Classes

1. Create a folder named **OOPIntro_LastName**
2. Create a Python file inside the **OOPIntro_LastName** folder named **Accounts.py** and copy the code shown below:



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

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

    def update_address(new_address):
        Accounts.address = new_address

    def update_email(new_email):
        Accounts.email = new_email
```

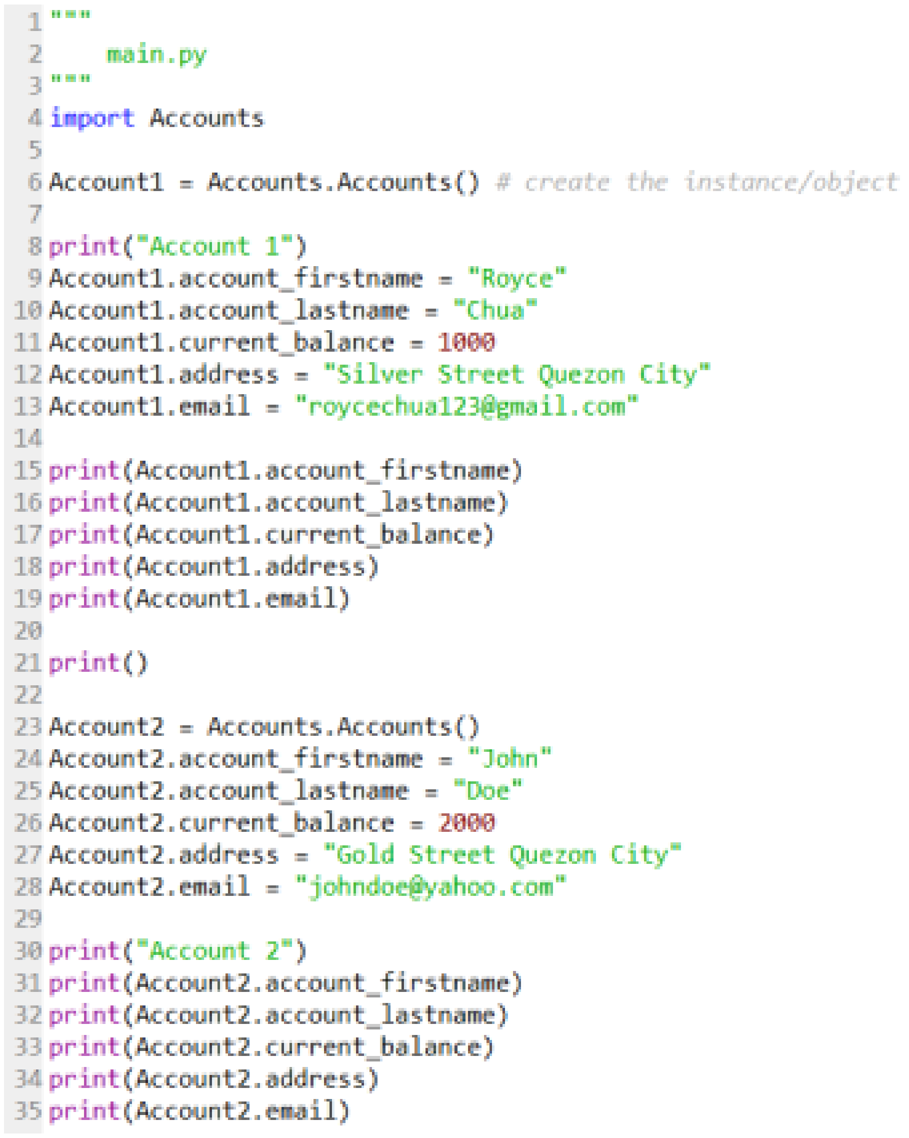
3. Modify the Accounts.py and add ***self,*** before the new_address and new_email.
4. Create a new file named ATM.py and copy the code shown below:



```python
"""
    ATM.py
"""

class ATM():
    serial_number = 0

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

    def widthdraw(self, account, amount):
        account.current_balance = account.current_balance - amount
        print("Widthdraw Complete")

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

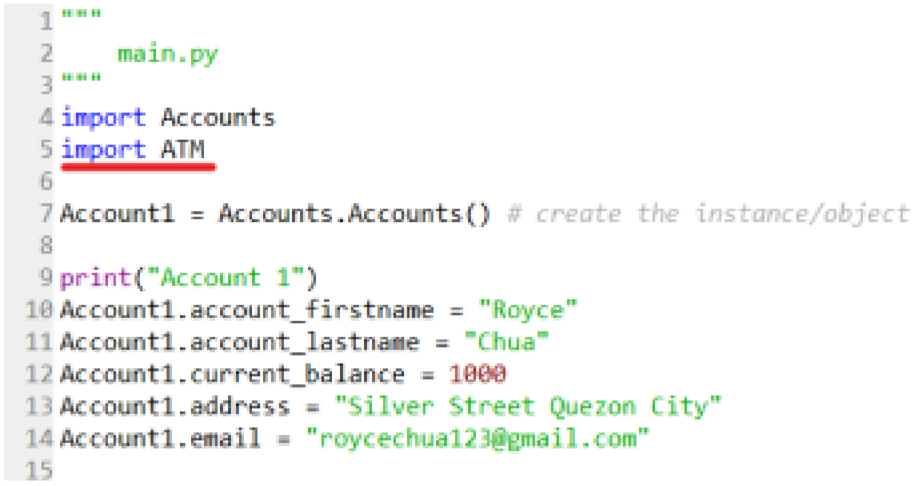
Creating Instances of Classes

5. Create a new file named main.py and copy the code shown below:



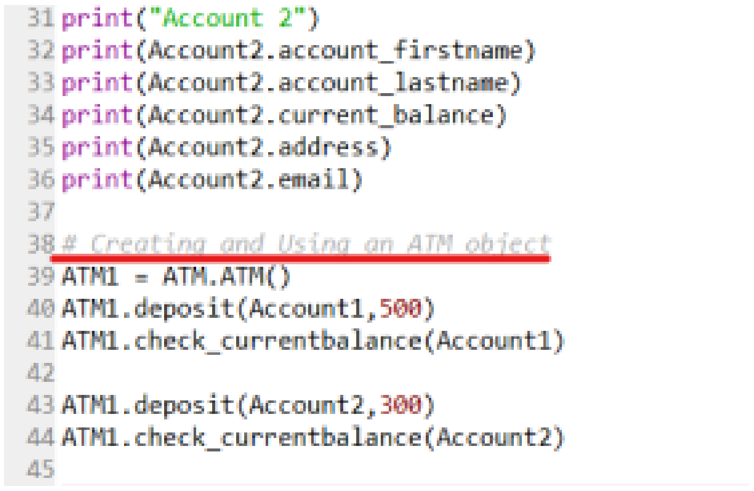
```python
"""
    main.py
"""
import Accounts

Account1 = Accounts.Accounts() # create the instance/object

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@yahoo.com"

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

6. Run the main.py program and observe the output. Observe the variables names account_firstname, account_lastname as well as other variables being used in the Account1 and Account2. 7. Modify the main.py program and add the code underlined in red.



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

Account1 = Accounts.Accounts() # create the instance/object

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"

```

8. Modify the main.py program and add the code below line 38.



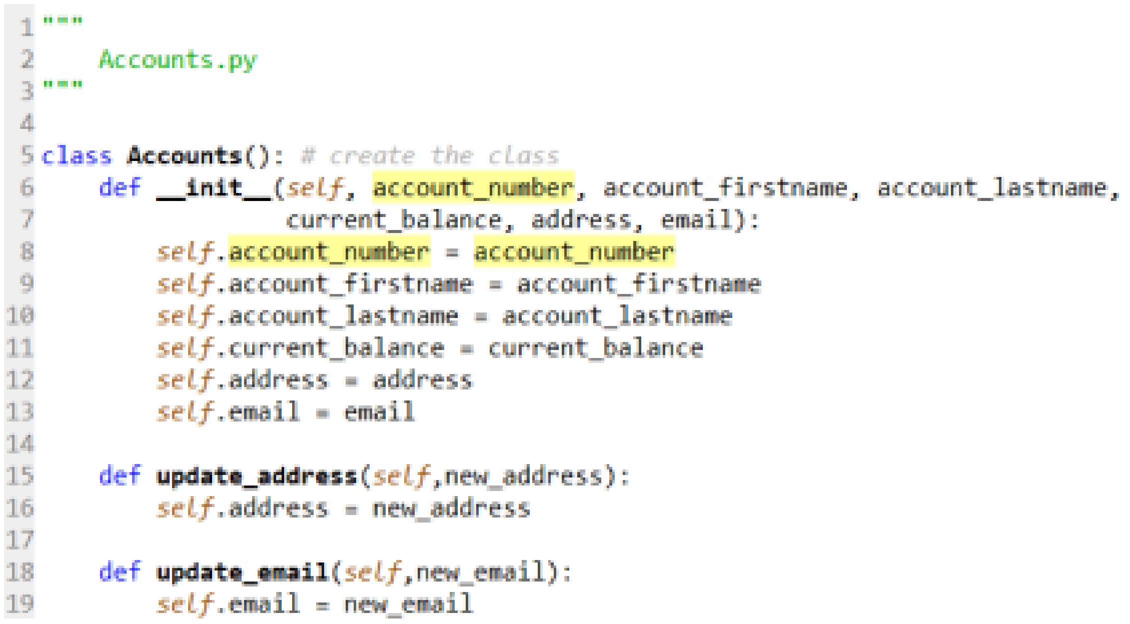
```python
print("Account 2")
print(Account2.account_firstname)
print(Account2.account_lastname)
print(Account2.current_balance)
print(Account2.address)
print(Account2.email)

# Creating and Using an ATM object
ATM1 = ATM.ATM()
ATM1.deposit(Account1,500)
ATM1.check_currentbalance(Account1)

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

```

9. Run the main.py program.

Create the Constructor in each Class

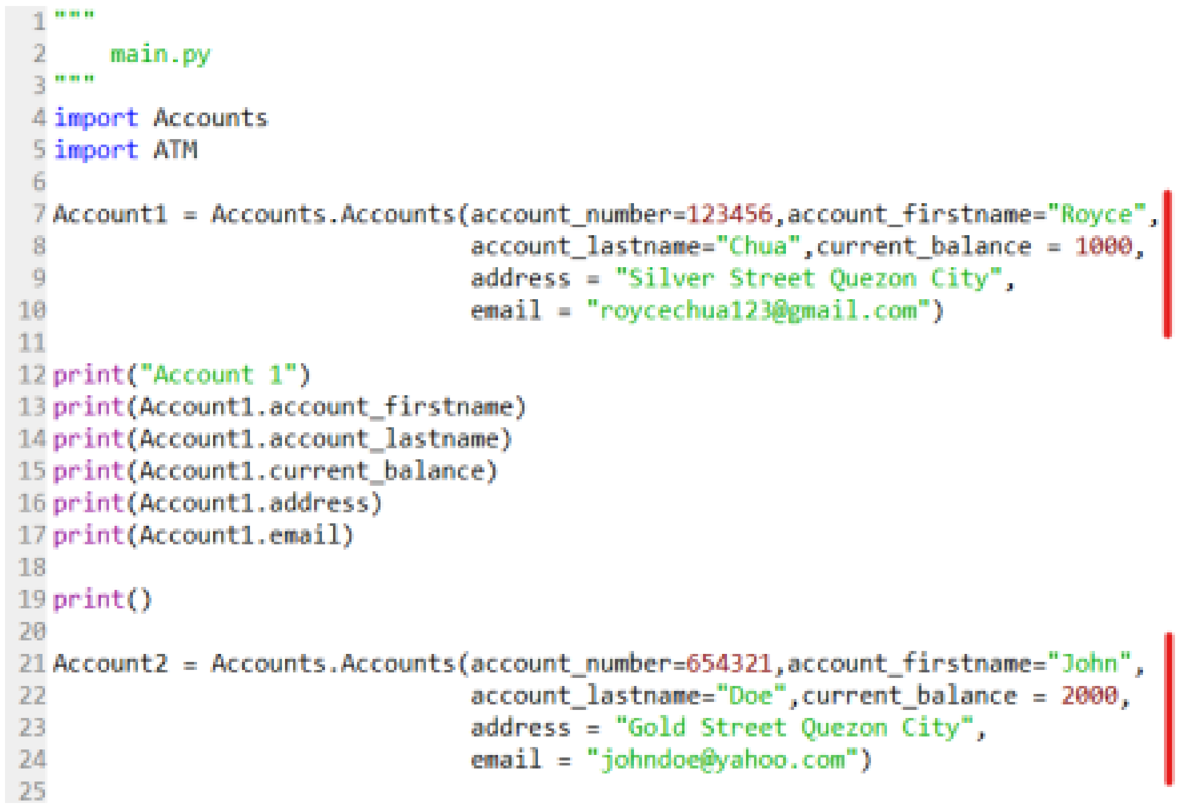
1. Modify the Accounts.py with the following code:

   Reminder: def __init__(): is also known as the constructor class



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

class Accounts(): # create the class
    def __init__(self, account_number, account_firstname, account_lastname,
                 current_balance, address, email):
        self.account_number = account_number
        self.account_firstname = account_firstname
        self.account_lastname = account_lastname
        self.current_balance = current_balance
        self.address = address
        self.email = email

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

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

2. Modify the main.py and change the following codes with the red line. Do not remove the other codes in the program.



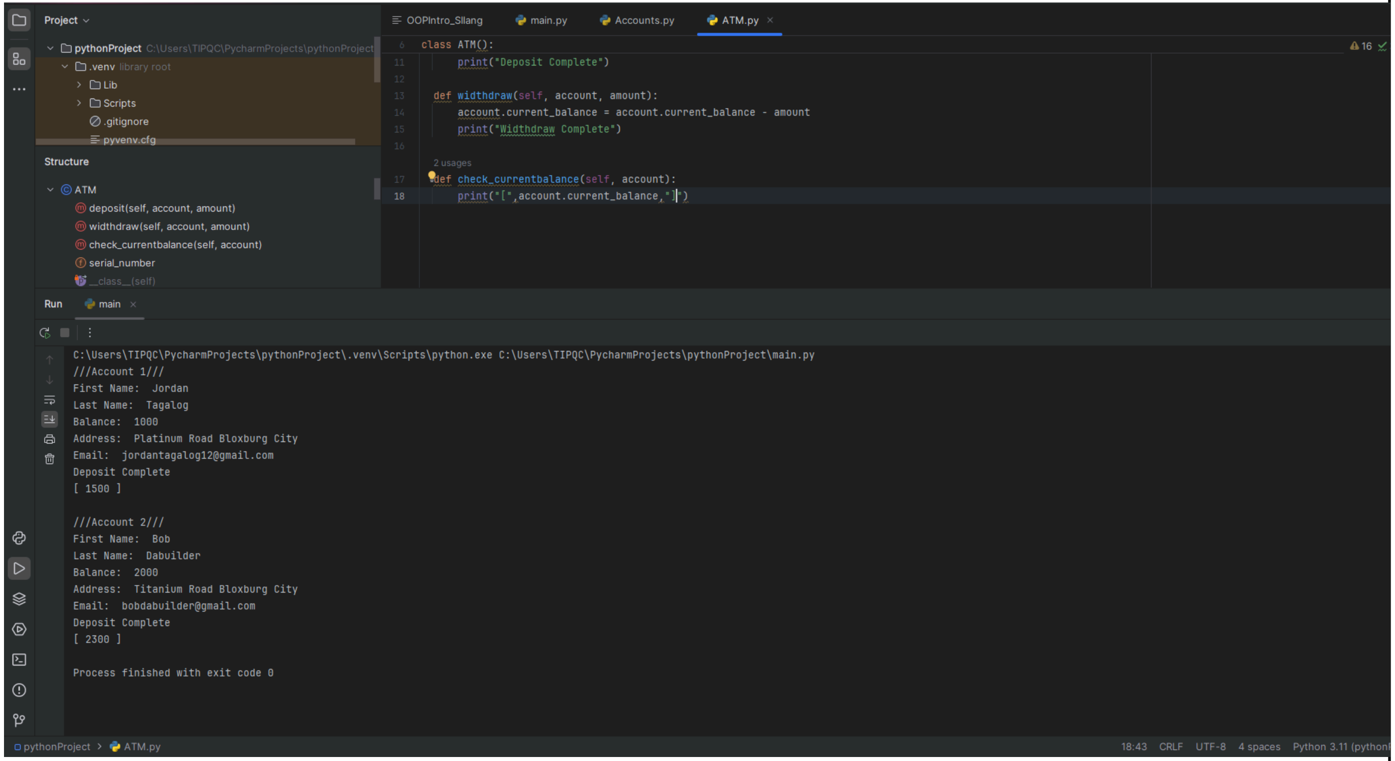
```python
"""
    main.py
"""
import Accounts
import ATM

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@yahoo.com")

```

3. Run the main.py program again and run the output.

6. Supplementary Activity:

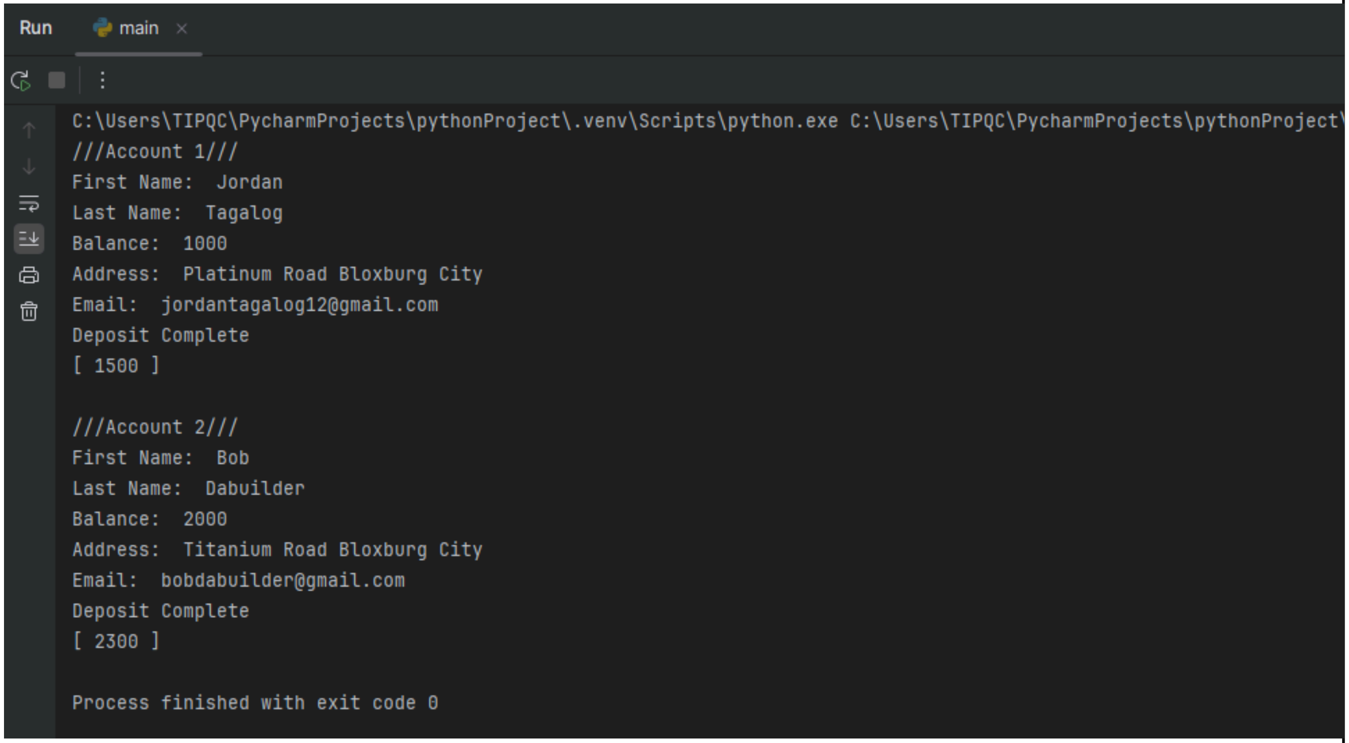
Tasks

1. Modify the ATM.py program and add the constructor function.
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.
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
class ATM():
        print("Deposit Complete")

    def widthdraw(self, account, amount):
        account.current_balance = account.current_balance - amount
        print("Widthdraw Complete")

    2 usages
    def check_currentbalance(self, account):
        print("[",account.current_balance,"]")
```

```
C:\Users\TIPQC\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\TIPQC\PycharmProjects\pythonProject\main.py
///Account 1///
First Name:  Jordan
Last Name:  Tagalog
Balance:  1000
Address:  Platinum Road Bloxburg City
Email:  jordantagalog12@gmail.com
Deposit Complete
[ 1500 ]

///Account 2///
First Name:  Bob
Last Name:  Dabuilder
Balance:  2000
Address:  Titanium Road Bloxburg City
Email:  bobdabuilder@gmail.com
Deposit Complete
[ 2300 ]

Process finished with exit code 0
```



```
C:\Users\TIPQC\PycharmProjects\pythonProject\.venv\Scripts\python.exe C:\Users\TIPQC\PycharmProjects\pythonProject\
///Account 1///
First Name:  Jordan
Last Name:  Tagalog
Balance:  1000
Address:  Platinum Road Bloxburg City
Email:  jordantagalog12@gmail.com
Deposit Complete
[ 1500 ]

///Account 2///
First Name:  Bob
Last Name:  Dabuilder
Balance:  2000
Address:  Titanium Road Bloxburg City
Email:  bobdabuilder@gmail.com
Deposit Complete
[ 2300 ]

Process finished with exit code 0
```

Questions

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

A class constructs objects.

2. Why do you think classes are being implemented in certain programs while some are sequential(line-by-line)?

The use of class is much more organized and you are able to hide the code by making it simpler with the use of an interface for users or clients. That is why it is suitable for real-world works or businesses.

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

The reason is that it is easier to read if you ever wanted to go back and analyze your code or maybe if there will be errors it is much easier to navigate and understand the code.

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?

A constructor initializes the objects created. It automatically executes when a new class is created. This means that every time you create an object from a class, the constructor organizes the attributes of the object created.

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

It is cleaner and more organized. This eases the work of the coder since it will be easy to read and analyze. This is also much more flexible. It has less coding if ever you want to create more objects since that it is already structured so you will only just add.

7. Conclusion:

In this lab activity I have learned that class really does make the work clean. It is easier to navigate and more neat to work on. If i ever want to add a function it is not as hard as sequential does since using class only needs less codes. Using this method will make work convenient and more user friendly.

8. Assessment Rubric: