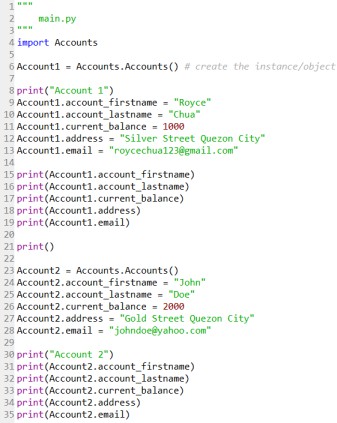
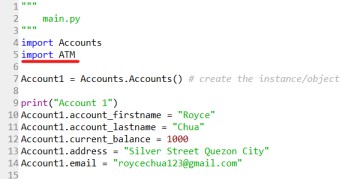
|  |  |
| --- | --- |
| **Laboratory Activity No. 3.1** | |
| **Introduction to Object-Oriented Programming** | |
| **Course Code:** CPE103 | **Program:** BSCPE |
| **Course Title:** Object-Oriented Programming | **Date Performed: 25/01/25** |
| **Section: 1A** | **Date Submitted: 01/02/25** |
| **Name: Disomnong, Jalilah M.** | **Instructor: Engr. Maria Rizette H. 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:   * 1. Identify the possible attributes and methods of a given object   2. Create a class using the Python language   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/Python Colab 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: |

|  |
| --- |
| 1. Modify the Accounts.py and add ***self,*** before the new\_address and new\_email. 2. Create a new file named ATM.py and copy the code shown below:     **Creating Instances of Classes**   1. Create a new file named main.py and copy the code shown below: |

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.

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

|  |
| --- |
| 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   1. Modify the main.py and change the following codes with the red line. Do not remove the other codes in the program. |

|  |
| --- |
| 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.   refers to this link <https://colab.research.google.com/drive/1oK7RYovZJzN5khZmzgnOZPESXH02HVir#scrollTo=dvjQ2E5qAjMV&uniqifier=2>  **Questions**   1. What is a class in Object-Oriented Programming?   In Object-Oriented Programming, Class is a blueprint of an object. It is a fundamental concept that allows for ordered and efficient program development. It includes data properties and methods, acting as a template for defining object structure and behavior.   1. Why do you think classes are being implemented in certain programs while some are sequential(line-by-line)?   The purpose of Object-Oriented Programming is to simplify and streamline processes, enhancing accuracy and precision using nearly identical syntax.   1. How is it that there are variables of the same name such account\_firstname and account\_lastname that exist but have different values?   Because there are different instances in different variables, their data are unique to each instance. |

|  |
| --- |
| 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 role of the constructor function is to assign values to the object of the class. The constructor  automatically executes whenever you create an object |

|  |
| --- |
| 5. Explain the benefits of using Constructors over initializing the variables one by one in the main program?  One advantage of utilizing constructor functions is the ability to assign various values immediately after  creating an object. This approach streamlines your code, eliminating the need to call functions to  initialize each variable individually. |
| **7. Conclusion:** |
| In this laboratory, we engaged in practical exploration of the essential principles of Object-Oriented Programming (OOP), with a particular emphasis on classes, attributes, methods, and constructors. We developed a basic banking system by designing Python classes that represented both accounts and ATM machines. By applying the concepts we learned—such as defining attributes and methods within classes, creating instances, and initializing attributes through constructors—we were able to observe the real-world applications of OOP in structuring and streamlining complex systems.  Utilizing constructors allowed us to effectively initialize object attributes, minimizing code redundancy and improving the program's clarity. Furthermore, our work with instances and the realization that identical attribute names can possess different values across various objects enriched our comprehension of object-oriented concepts. Overall, this exercise highlighted the effectiveness of OOP in crafting clean, modular, and maintainable code. |
| **8. Assessment Rubric:** |