Final Project

ATM Simulator (40%)

DUE DATE

Session 15

OBJECTIVE

The main objectives of this project are to:

- Use the Python syntax to read user input, process information and display output.
- Declare and initialize variables using the Python variable naming conventions and keywords.
- Understand the different data types used with the Python programming language.
- Code decision-making structures and Boolean logic using Python.
- Code repetition structures using Python.
- Code selection and sequence structures using Python.
- Define, declare and call void and value returning functions.
- Pass arguments to functions.
- Implement file input and output functions.
- Use loops to process content of files.
- Implement exception-handling constructs in Python.
- Perform operations on sequences, lists and tuples.
- Perform basic string operations using built-in string functions.
- Test, search and manipulate strings.
- Demonstrate and understanding of dictionaries and sets as Python objects.
- Apply object oriented programming concepts to Python programming.
- Use the Python functions, features and libraries to create graphical user interfaces.

DESCRIPTION

By the time you begin this project, you should have sufficient background in Python and programming concepts to be able to work through this project with relative ease. This project will allow you to apply all that you have learned in the Python course by implementing the skills within the context of developing a typical program in Python.

The focus is to apply Python programming skills regardless of the context. For this reason, you will be building yet another ATM system. Because this is a familiar scenario, it is a good premise to use this since it will allow you to focus on the python implementation itself as opposed to having to re-think the entire logic.

You can complete this project in many ways. It will be up to you to decide how you want to implement the functionality. For example, you can use list and tuples to store information of the users and accounts, or you can store the information in files that can store the updated information for future use. You can have the program set-up as a command line setup or as a GUI interface. The functionality should be the same regardless of how you implement the program.

The project development may follow two different formats, Basic Functionality and Extended Functionality. If you select the Basic Functionality model, your project will be graded over a maximum of 65/100 points. (Maximum you can obtain is 65%). If you take the Extended Functionality option, then your project will be graded against the maximum 100% grade as you will be including program modularization, functions, classes, GUI, etc.

A Basic Functionality model for this project is one where everything is programmed within the same project file, there are no functions called, or returning values, the entire program is executed using the basic programming constructs of loops, selections and decisions, local and global variables and lists. User information and account information and balances are stored in volatile structures such as lists and tuples. This means that every time the program simulator runs, it starts with the same original data that is built into the program when it is first created. This model allows you to test the ATM simulator basic functionality and the basic Python syntax.

If you opt to work with an Extended Functionality model, you will still need to respect the same requirements as above. However, you will implement the application using functions, classes (Object Oriented Programming), and the data is stored in files that can be saved and updated for future use. For example, if a user withdraws $60 from an account that has $1000, then the next time the program runs, the balance for that user should be $940. Otherwise, using local lists or tuples, the next time the program runs, it would revert back to the original programmed values. This would also allow for additional functionality that would otherwise not be available with the Basic Functionality model. For example, the ability to have an administrator add new users and accounts.

If you adopt the basic model, but add *certain features* of the extended model, such as implementing the use of files instead of lists, which would allow you save data, and thus also implement fully the change PIN functionality, you will be given the credit for it, and potentially have a higher grade than the maximum 65. The decision is yours as to how you implement your program solution. If you plan your time effectively, you should have enough time to create a fully functional program that implements all the necessary features for the maximum grade.

Time Required

You will be provided 16 hours in class to complete this project.

Required Material

You will need the following material to complete this project:

- Python IDLE Editor

INSTRUCTIONS

Key Functions

The main functions of the ATM simulator may be summarized as follows:

Before performing any transaction, the user must enter his or her username and PIN (personal identification number) at an input screen/prompt (GUI or otherwise).

The system must validate the user information against the information that is pre-built into the system. For the purposes of this project, the system will have the following three username: PIN pairings to start.

User1: 1234
User2: 2222
User3: 3333

These can be stored in volatile (temporary) storage or in a file or series of files depending on whether you opt for the Basic Functionality or the Extended Functionality.

The prompt for the user information should allow for three attempts. If the user does not enter a valid username:PIN combination, the system must display an appropriate error message after each attempt and re-prompt. After the third invalid attempt, the system simply ends the program.

Basic Functionality	Extended Functionality
• Use Lists to store Usernames and PIN pairings	• Use files to store Usernames and PINS • Must have an administrator login feature

The names and PINs of users must be validated using data contained in the appropriate lists or files using the following criteria:

Username (String)
PIN (String – 4 characters)

Once access has been authorized, the main transaction screen of the application should allow the user to carry out one of the following transactions:

Basic Functionality	Extended Functionality

• Account Balance • Withdraw	If user is administrator Menu options include :
• Deposit • Change PIN • Exit	• Add new user • Delete user • Plot Account Balances • Exit If the user is a regular user The Menu options are • Account Balance • Withdraw • Deposit • Change PIN • Exit

For the purposes of this project, all transactions will be carried out on a single account per user. There is no distinction between a chequing and a savings account.

After each transaction, with the exception of Exit, the system must return to the menu display and prompt the user for another transaction choice. In the case of the Administrator, upon exiting, the system should prompt for a new user and PIN

Deposit,

- The user must enter the amount to be deposited into his or her account. The amount of the deposit must be added to the current balance into the account. Appropriate confirmation messages must be provided indicating the amount of the deposit and the new current balance. Proper data input validation to ensure that numerical data is input must be used along with error handling.
  Keep in mind that if you are working with the Basic Functionality model, the balances are stored in the list, which will be lost once the program ends. If you use files to store the information, then the new balances should be saved and available when the program runs subsequent times.

Withdrawal,

- The user must enter the amount to withdraw. Each withdrawal transaction is subject to a maximum of $1,000 per transaction. If the user enters an amount greater than $1000, the system must present a message indicating that $1000 is the maximum per withdrawal. The ATM accepts only transactions for which the amount entered is a multiple of $10. There is no daily maximum amount apart from the user's account balance in which case the system must also inform the user if the account balance is less than the transaction amount.

Account Balance

- When the user selects this option, the system simply checks the data for that user and displays the current balance. Checking the account balance does not alter the balance itself.

Changing the PIN

- Changing the PIN is available to all users. The difference in how this feature operates depends on how you set up the program. If you are using the Basic Functionality model, using lists to store user information, the change in PIN will only be effective during the current running of the program. Once you quit the program and start it again all data values return to the original values. However if you are working with the Extended Functionality model and you are using files to store the information, the change in PIN will be committed to the file and be available for subsequent accesses using this new value.

Administrator Functions (Extended Functionality)

The following functions concern the ATM's functioning with respect to the system administrator and the internal mechanisms of the ATM, not the user.

- The system administrator, as any other user, must enter his or her name and PIN (personal identification number) on the same input screen. The system administrator may perform only system transactions (he or she has no personal account).
- The System Administrator must login with the unique User Name of SysAdmin and the PIN 1357. When these credentials are entered, the system must automatically display only the Administrator screen with its options
    - ❖ Add new user
    - ❖ Delete user
    - ❖ Exit

Add a new user

The administrator is the only user wo can create new user accounts. When this option is selected, the system must create a file or files for the new user to store the user information and account transactions and balances. There are many ways that this can be setup. You can use a single file to store all usernames and PINS followed by individual files for each user's account details (hint, name each user's account file using the username as the file name). Alternatively, you can create a single file per user which stores the username, PIN and account balances. There is no right or wrong way, it is up to you to decide how you want to set it up.

Hint: This feature should be one of the first functions that you create so that you can then use it to create your user files with the three initial users indicated in the first section of the specifications. You have to determine how you will store the information (Single file containing all users, and account information, single file for users and PINS with individual user account files, or a series of individual user

files that house all user and account details for that user.)

Delete user

When the administrator selects this option, they should be able to simply prevent the user in question from logging in. This could be accomplished by deleting the file(s) associated with that user, or, by changing the PIN number for the user to a pre-determined value that would remain unknown to the user.

Plot Account Balances

This feature of the Administrator menu is not something that is realistic, however it is used in this project for the sole purpose of having you implement the use of Python's graph and plotting feature.

When the administrator selects this option, the program will work through the user accounts and plot the current account balances for each user (you can use the type of plot of your choice: pie, bar etc) Keep this to a maximum of 10 users. The plot should indicate the user and corresponding account balance.

Exit

When the administrator selects this option, the system should return to prompting for a new username and PIN.

General Design Features Basic Functionality Model

One thing to consider if you opt for the Basic model of the program is that the data will never be saved for future use since everything is stored in lists and other volatile structures. You program will be contained in a single file (not modularized) and will use the basic programming constructs to control the flow of the program. This is not the ideal programming approach but for the purposes of applying basic Python programming, it is sufficient. This does not mean that you cannot implement more advanced features into this model. You can opt for the basic model but you can also replace lists with files to store information for future use. This would allow for additional points as per the marking scheme.

Extended Functionality Model

If you decide to create a more robust application that goes beyond the basic use of Python syntax, there are a number of considerations that you need to account for. You will use modularization, and Object Oriented programming, with files to store the data. That in itself will provide a more robust program design. In addition to that, you will need to consider what information to store in the file(s) and how you will store it. You must consider all the requirements above for regular users and administrators and decide how you will implement them. For example, when plotting the account balances under the administrator feature, you must make sure that you have the data available in a suitable format.

Error and exception handling

Regardless of the model you use, error and exception handling must be implemented wherever possible. Data validation must also be implemented. For example, when the system requires the user to

enter a dollar amount, the system should make sure that the input is numeric and not alphabetic.

General functional Considerations

- The application must check the account balance before doing a transaction. Any transaction that would result in a negative balance must be rejected.
- The balance of the account affected by a transaction should be updated and displayed after each transaction.
- The user should be able to do as many transactions as he or she would like to do before leaving the ATM.

SUBMISSION INSTRUCTIONS

Your assignment must include the following:

All source code, as well as all other files required for the proper functioning of your programs. We strongly recommend that you verify, using anti-virus software, that your submission contains no viruses.

- A project report containing:
  - ✓ a title page with your name, the submission date and your instructor's name
  - ✓ a table of contents
  - ✓ the project specifications
  - ✓ all relevant documentation including diagrams, flowcharts, pseudocode
  - ✓ the user manual (instruction guide for your instructor, This must include any test data that you use to test your application)
  - ✓ explanations of any additions that you made (where applicable)
- Place all files in a single folder
- Zip your folder into a single file and follow the naming convention described below.

Work must be submitted in the correct file type and be properly labelled as per the College naming convention:
NAME_COURSE_ASSIGNMENT. E.g. XuXiaLing_FM50D_A01.

GRADING CRITERIA

Assignment Value: **40%**

Grading Criteria	Basic Functionality Grading	Extended Functionality Grading
Appropriate program flow and proper use of syntax, import statements etc.	5	5
Program fully documented with appropriate comments	5	5
User ability to log in and display of appropriate messages if unsuccessful	5	5
Deposit feature uses appropriate prompt and displays appropriate messages	5	5
Withdrawal feature uses appropriate prompt and displays appropriate messages	5	5
Change PIN functionality	2	5
Transactions are validated to make sure that the account balance is sufficient to cover the transaction	5	5
Use of appropriate mathematical functions and methods	5	5
Incorporation of the necessary if, if/else, else if and switch statements	10	10
Adding of the appropriate exception handling	10	10
Input validation and error handling	5	5
Use of lists*, files** and other storage structures	3*	5**
Addition of and correct functioning the Administrator features	N/A	5
Modularization of program	N/A	5
Use of Object Oriented programming features	N/A	5
Plotting of user balances	N/A	5
Implementation of GUI interface	N/A	10
TOTAL	**65**	**100**

Penalties

- A penalty of 5% will be deducted from your mark for each day your project is late.
- Any project submitted more than 3 calendar days late will receive a maximum mark of 60%.
- Any project containing a virus must be resubmitted and will receive a maximum mark of 60%.

Special Note for grading

A student that adopts the basic non-modular approach to this program will be graded out of the maximum of 65 points if he or she creates the program in the simplest manner as per scheme above. However, if the student maintains the basic model, but uses or implements certain features of the Extended Model, you can give him or her the extra points brining the grade potentially higher than the 65%.

Example:

A student creates the basic non-modular program. However, instead of using lists to store the user information (that gets reset to the original values each time the program runs), the student uses files which can save the updated data for future use. In this case, the student would benefit from the 10 marks allocated for the Extended model, instead of the 5 marks allocated to the Basic Model.

The same logic applies then to the marks allocated to the Change PIN feature. If the student uses a list to store the user information, then the Change PIN feature is also temporary will be allocated the marks based on the Basic Model. However, is the student uses files as described above, then the Change PIN feature can also be allocated the marks given to the Extended Model since the update in PIN will be committed to file and the new value will be used the next time the program runs.

Obviously, if the student implements the general modularized design, then it is expected that the entire program will be evaluated against the Extended Model Grading Scheme.