CSE 1321L: Programming and Problem Solving I Lab

Assignment 4

Module 3

What students will learn

- Problem Solving.
- Basic Program Structure.
- Input and Output with the user.
- Write code that includes while/for loop and nested loops logic.
- Structure program to include methods.

Content

- Overview
- Assignment4A: Pascal Case Conversion
- Assignment4B: Password Check
- Assignment4C: ATM Simulator
    - A4C_Functions: Helper functions file for assignment 4 (c)

Overview:

For this assignment, you're going to practice making logic in your code. It will include loops and nested loops. In practical terms, this means you're going to expand on the concepts from previous assignments, but also include things like while loop, for loop statements. Again, start early, practice, and ask a lot of questions.

Final note: *Do not cheat*

If your temptation is to look online, don't. Come see us instead and ask questions – we are here to help. Remember, you are going to have to write codes in your future job interviews, so learn it now to secure a high-paying job later.

Assignment4A: Pascal Case Conversion

Write a Python program that converts a given text into **Pascal Case** by implementing two functions:

format_word(word) – This function should take a single word as input, capitalize its first letter, and convert the rest of the letters to lowercase.Finally this function will return the converted/formatted word.

convert_to_pascal_case(text) – This function should iterate through the entire string. When it encounters a space character, it should treat the collected word as complete and call another function to convert that word to Pascal Case. Once the full string is processed, it should return the final Pascal Case string.

Requirements:

- Write main function to take user input as text and in this main function you will be just calling another function i.e. convert_to_pascal and will display the returned cases in main function.
- The program should handle multiple spaces between words.
- The final output should not contain any space.
- You are not allowed to use arrays/lists.
- You can use str.capitalize() to convert first letter to capital.

Example runs are shown below. The user input is shown in **red**.

Sample Output #1:

```
Enter a string: this is pascal case
Pascal Case: ThisIsPascalCase
```

Sample Output #2:

```
Enter a string: convert this to pascal case
Pascal Case: ConvertThisToPascalCase
```

Sample Output #3:

```
Enter a string: first letter of each word is captital
Pascal Case: FirstLetterOfEachWordIsCaptital
```

Sample Output #4:

```
Enter a string: FiRST letter OF Each WORD IS CAPITAL
Pascal Case: FirstLetterOfEachWordIsCapital
```

Assignment4B: Password Check

Write a Python program that checks whether a password meets security requirements. The program should **continuously prompt the user** until they enter a strong password.
You must **write three functions**, each checking a specific requirement:

- **`check_length(password)`** – This function should check if the password is **at least 8 characters long**.
- **`check_upper_lower(password)`** – This function should check if the password **contains both uppercase and lowercase letters**.
- **`check_special_character(password)`** – This function should check if the password **includes at least one special character (`!`, `@`, `#`)** using selection statements (`if` conditions).

Requirements:

- The program should keep asking for a password until all conditions are met.
- You cannot use lists or arrays.
- In the main function, take user input for the password, call the three functions, and display appropriate messages.
- If the password meets all requirements, print `"Password is strong!"` and stop asking for input.
- If the password **fails any requirement**, display the missing criteria and ask the user to enter a new password.

Example runs are shown below. The user input is shown in **red**.

Sample Output #1:

```
Enter a password: Hello123
Password does not meet the requirements: Must include at least one special character (!, @, #).
Enter a password: 123
Password does not meet the requirements: Must be at least 8 characters long. Must contain both uppercase and lowercase letters. Must include at least one special character (!, @, #).
Enter a password: hello123@
Password does not meet the requirements: Must contain both uppercase and lowercase letters.
Enter a password: Hello123@
Password is strong!
```

Sample Output 2:

```
Enter a password: Strong@
Password does not meet the requirements: Must be at least 8 characters long.
Enter a password: strong@1
Password does not meet the requirements: Must contain both uppercase and lowercase letters.
Enter a password: Strong@2
Password is strong!
```

Assignment4C: ATM Simulator

Write a Python program to simulate an ATM system using multiple functions. The program should start by asking the user for their name and an initial balance. It should then display a main menu with the following options:

1. **Deposit** – The user can enter an amount to deposit, which will be added to their balance and updated balance will be displayed.
2. **Withdraw** – The user can enter an amount to withdraw. If the balance is sufficient, deduct the amount; otherwise, display an insufficient funds message.
3. **Check Balance** – Display the current balance.
4. **Exit** – Exit the program.

The program should continue running in a loop until the user chooses to exit. Implement separate functions for each operation (Deposit, Withdraw and balance check) and ensure proper handling of invalid inputs.

For this task:

You will be writing the following functions into a separate file named **A4C_Functions.py**

- **Display_Main_Menu()-** This function will display main menu with 4 options described above. Take the input from user about choice and return the user choice to main function.
- **Deposit(balance)-** This function will take current balance as input and ask about the amount to be added/deposited and will return updated balance. **You need to update the account balance with this returned balance in main function.**
- **Withdraw(balance)-** This function will take current balance as input and inside function ask the user about the amount to be withdrawn. If the amount is greater than the current balance display a message insufficient balance and if amount is smaller than current balance withdraw/minus the amount of current balance. Finally, this function will return updated balance.
- **Check_balance(balance)-** Use this function to display current balance which is passed as argument

All of these functions will be in a separate file named A4C_Functions.py and in Assignemnt4C.py file you will have main function. In the start take input of user name and initial balance and after that you will call display menu function first and get the user choice. Further, based on this user choice you will be calling respective functions. You will keep displaying menu until user choose to exit.

Example runs are shown below. The user input is shown in **red.**

Sample Output #1:

```
Welcome to the ATM!
Please enter your name: Usman
Enter your initial balance: $1500
ATM Main Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Please choose an option (1-4): 1
Enter the amount to deposit: $50
Deposited $50.0. New balance: $1550.0

ATM Main Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Please choose an option (1-4): 8
Invalid choice. Please try again.

ATM Main Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Please choose an option (1-4): 2
Enter the amount to withdraw: $200
Withdrew $200.0. New balance: $1350.0

ATM Main Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Please choose an option (1-4): 3
Your current balance is: $1350.0

ATM Main Menu:
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Please choose an option (1-4): 4
Goodbye, Usman! Thank you for using the ATM.
```

Submission Instructions:

- Programs must follow the output format provided. This includes each blank line, colons (:), and other symbols.
- Programs must be working correctly.
- Programs must be written in Python.
- Programs must be submitted with the correct **.py** format.
- Programs must be saved in files with the correct file name:
  - Assignment4A.py
  - Assignment4B.py
  - Assignment4C.py
    - A4C_Functions.py
- Programs (source code files) must be uploaded to Gradescope by the due date.