22603VIC – Certificate IV in Cyber Security

ICTPRG435 – Write Scripts for Software Applications

Assessment Task 2 – Practical: Develop Scripts

Student Name	**Princedhorn Wongsuphap**
Student ID	**8106485**

Answer all questions in the boxes provided. Each box will expand to fit your answer.

Lab Guide
To complete this assessment, you will need to access a computing system with a Python installation and suitable Integrated development environment. The current version of Python and Visual Studio Code are the suggested tools to use. Your own system is likely to be suitable for this. However, a system can be provided to you if needed. Discuss this with your teacher and a Azure lab registration link can be provided to you. The following credentials can be used to access the lab. Ask your teacher for assistance if you are having difficulty accessing the lab.

Username	Student
Password	!nf0tecH

Assessment scenario – Password Manager

You work as a programmer for a small digital development agency "Apps2U". Apps2U have asked you to develop a script on behalf of their customer "DigiCore".

DigiCore have requested a simple password manager because their staff are resorting to insecure practices to record login credentials such as writing them down on sticky notes. The password manager will be used to store and retrieve credentials for websites and other login services used by employees.

Script Requirements Your code will be written to the following requirements	
	Done
Include an options menu for the user allowing them to carry out the following actions • Add stored credentials and related resources (username, password and URL/resource) • View stored credentials • Exit the script	**Done**
Return to the menu after each action has completed	**Done**
Create a text file for credential storage if a text file does not already exist	**Done**

Append new records to the text file without overwriting previous entries	**Done**
Display the text file contents in a visually presentable way including spacing and headings	**Done**
Handle any input from the user and carry out actions, without errors	**Done**
Include embedded explanatory comments (#) to clarify the meaning of the code where necessary	**Done**
Structure code with the layout, naming conventions, white space and comments recommendations in the PEP 8 Style guide for Python code	**Done**
Include at the top of the script, the file name, author's name, script purpose and, the date development began and ended	**Done**

Task 1.1 Identify and discuss script requirements

Discuss the script requirements above with your instructor who is acting as your work supervisor. This is your opportunity to ask any questions to ensure you fully understand the requirements. Provide brief notes or dot points of the discussion that show your attempts to clarify and understanding the script requirements. *Note that your instructor may work with you individually, in small groups or as one large group for this task*

Enter brief notes or dot points of the discussion

1. **Add Stored Credentials**
   - The script should allow users to add and store credentials for websites or services
   - Need input for URL name, username, and password
   - The credentials will be stored in a dictionary, and each entry will contain URL name, username, and password.
   - These entries will be appended to a text file for storage
2. **View Stored Credentials**
   - The script should allow users to view the stored credentials
   - If no credentials exist, the script will display a message indicating that
   - The stored credentials will be read from the text file and displayed in a visually presentable way with headings
3. **Exit the Script**
   - Users should be able to exit the script by choosing the “Exit” option
   - We will provide a user-friendly exit message
4. **Create Text File for Credential Storage**
   - If the credential text file does not already exist, the script should create it. ‘os.path.exist()’ function will be used to check for the file’s existence.
5. **Append New Records Without Overwriting**
   - New records should be appended to the text file, ensuring that existing entries are not overwritten.
6. **Display Text File Contents**
   - When viewing stored credentials, the contents of the text file will be display in a visually presentable way, including headings and proper spacing.
7. **User Input and Actions**

- Validate user input correctly and ensuring that they are within expected ranges and format, such as menu options, URLs, usernames, passwords). **8. Comments and PEP 8 style guide** - Explanatory comments(#) will be included in the code to clarify its functionality - The code will adhere to the PEP 8 style guide and ensures proper naming conventions, whitespace, and code layout. **9. Header Information** - The script will include a header with the file name, author's name, script purpose, and development dates.

Task 1.2 Identify and discuss script outcomes

Discuss the following software outcomes with your instructor who is acting as your supervisor. Complete the table rows by answering the questions.

Script outcome discussion
What is the benefit of using the script for the client?
- The script provides a secure and convenient way for the client, DigiCore, to manage and store login credentials - It helps eliminate the risk of insecure practices such as writing passwords on sticky notes. - By centralising credential storage, it enhances security and reduces the chances of credentials being lost or compromised. - The script is user-friendly, making it easy for employees to use and navigate
What are the features and functions of the script? ***What is it that the script does overall, what are its individual functions, how will a user navigate the script, what are the inputs, what are the outputs, are there any special features?***
- The script has a main menu with options to add, view, and exit. - "Add stored credentials" function lets users input and store URL name, username, and password. - "View stored credentials" function displays stored credentials - The script appends new entries to a text file without overwriting previous ones. - User navigation is menu-driven, with numeric choices (1/2/3) to select actions - Inputs include URL names, usernames, passwords, and menu choices. - Outputs are displayed on the console and include success messages and stored credentials
Appearance ***How will the user interface look? How will any outputs from the script look?***
- The user interface is text-based and runs in the console or terminal - It displays a menu with numbered options - Outputs appear as text messages and displayed credential information
When is the script due for completion?
13/10/2023 to 24/10/2023

What questions do you have about the script or the development process?
The questions about the script and client requirements will depend on the client's needs and preferences. These are questions that might consider asking clients: - Do they have any specific security requirements for credential storage? - Is there a need for additional features or security measures such as encryption? - Are there any specific formatting or display preferences for stored credentials? - Do they have a deadline or specific timeline for completing the script's development?

Task 1.3 Algorithm Design

Produce a design of your intended script using pseudo-code or a flowchart with explanatory comments. Use the script requirements to guide the design of your algorithm/s

Contact your instructor for constructive and corrective feedback when complete. Review the algorithm design based on the feedback and, provide brief notes or bullet points of the review discussion. *Note that your instructor may choose to provide feedback individually, in small groups or as a large group.*

Enter your reviewed pseudo code or flowchart:

I also have the attached pseudo code file to the dropbox submission.

```
# Password Manager Pseudocode
# Author: Princedhorn Wongsuphap
# Purpose: To store and retrieve login credentials securely.
# Date Started: 13/10/2023
# Date Ended: 25/10/2023

# Initialize an empty dictionary for credential storage
credentials = {}

# Define the file path for storing credentials
file_path = "credentials.txt"

# Check if the credential file already exists, and if not, create it
if the file at file_path does not exist:
    create an empty file at file_path

# Function to add credentials to the dictionary and file
Function add_credentials():
    Display("Enter URL/resource name:")
    URL_name = Input()
    Display("Enter username:")
    username = Input()
    Display("Enter password:")
    password = Input()

    # Add the credential to the dictionary
    credentials[URL_name] = {"username": username, "password": password}

    # Append the credential to the file
    Open the file at file_path in append mode
    Write "URL: " + URL_name + "\n" to the file
    Write "Username: " + username + "\n" to the file
    Write "Password: " + password + "\n" to the file
    Write a newline character to the file  # Add spacing between entries
    Close the file

    Display("Credential added successfully!")

# Main menu loop
While True:
    # Display menu options
    Display("Password Manager Menu:")
    Display("1. Add stored credentials")
    Display("2. View stored credentials")
    Display("3. Exit")

    # Get user's choice
    choice = Input("Enter your choice (1/2/3):")

    # Handle user's choice
    If choice is "1":
        # Add stored credentials
        add_credentials()

    ElIf choice is "2":
        # View stored credentials
        If the credentials dictionary is empty:
            Display("No credentials stored yet.")
        Else:
            Display("\nStored Credentials:")
            Open the file at file_path in read mode
            Display the contents of the file
            Close the file

    ElIf choice is "3":
        # Exit the script
        Display("Exiting the Password Manager. Goodbye!")
        Break  # Exit the loop

    Else:
        # Invalid choice
        Display("Invalid choice. Please select 1, 2, or 3.")
```

Enter brief notes or bullet points of discussion during review:

- Use file functions to read
- Write credentials in the pseudo code

Task 1.4 Coding and internal documentation

Translate your pseudo code into a Python3 script that meets the script requirements and outcomes.

I have attached files to the dropbox with my submission:

1. passwordmanager.py

2. credentials.txt

Task 1.5 Debugging

- Provide a short Panopto video (3 minutes maximum) with vocal commentary, showing an IDE debugger in action, including a variables contents changing, stopping at a breakpoint, stepping over a function and, stepping into a function
  Be brief by not including any more content than is required for the task

Upload the Panopto video to the assessment dropbox for this unit. Also, enter the hyperlink to the Panopto video here and ensure that your instructor has read access to the video I have attached a video file to the assessment dropbox with all my other submission files.

- Provide a list of three semantic/logic errors you have encountered and how you rectified them. *Note that syntax errors are not acceptable. The errors must be caused through incorrect logic*

Error	Rectification
e.g. Code to view the text file is not being ran	*The menu option was capitalised but user entry was in lower case. So, added code to accept both upper and lower case* *if option == 'a' or option == 'A'*
In the "View stored credentials" section, the code may not properly format or display the stored credentials	Ensure that the code reads the credential entries from the file properly and format correctly for display. So, might use a loop to read and display credentials one by one
If a user tries to add credentials for a service that already exists in the dictionary, the code may not handle this case properly.	Add logic check whether the service name already exists in the dictionary and ask the user if they want to update the credentials or skip.
The code doesn't include data validation for inputs like URL/resource name, username, and password, which can lead to unexpected input.	Include data validation checks to ensure that inputs meet certain criteria, such as URL format, password strength, before accepting and storing them.

Task 1.6 Testing and testing documentation

Develop a list of test cases to confirm the code meets the script requirements and outcomes as well as testing for error conditions. List ALL the test cases that need to be run.

Brief Description: ***(what is being tested)***	**Brief Description:** ***(what is being tested)***	**Brief Description:** ***(what is being tested)***	**Brief Description:** ***(what is being tested)***
e.g Information message is generated when attempting to view the database file if the database file does not exist.	Enter valid URL name, username, and password then verify that the credential is added to the credentials dictionary and the file	Attempt to view stored credentials when none exist. View stored credentials after adding multiple credentials	Choose the "Exit" option from the menu and being able to exit.
Add a credential with less than 8 characters URL name and password to see if the script can handle the restriction properly.	Enter an invalid menu option, such as "4" or "abc" then ensure that the script is able to provide an error message	Delete the credential file and run the script and verify that the new credential is appended to the file and doesn't overwrite the previous entry	Add a credential with same URL name as an existing one and verify that the new credential is appended to the file and doesn't overwrite the previous entry
Delete the credential file while the script is running and verify that the script can handle the missing file without error.	Add a large number of credentials to test the script's performance	Test each menu option by providing incomplete input such as service name only, missing username, or missing password.	Add a credential with an unusually long URL name and password to see if the script can handle and display long credentials properly.

Record two of your listed test cases as well as any subsequent code modification that occurred if any.

EXAMPLE

Title: Data added to the file is correctly formatted

Preconditions: database exists, menu option has been chosen

Steps	**Expected response or output**	**Actual response or output**
Add username: user1 Add password: pass1 Add related resource: URL1	Information message: "Your data has been saved"	Information message: "Your data has been saved"
Open database with text editor	Database file contents should contain the typed credentials and resource in the required format username: user1 password: pass1 URL: url1	username: user1 pass1 url1 password: URL:
Code modification	Modify code writing to file to place credentials on individual lines	

Title: Enter an invalid menu option, such as “4” or “abc” then ensure that the script is able to provide an error message **Preconditions:** database exists, menu options exists		
Steps	**Expected response or output**	**Actual response or output**
Enter your choice (1/2/3): abc	Invalid Choice. Please select 1, 2, or 3	Invalid Choice. Please select 1, 2, or 3
Code modification	No code modification required	

Title: Add a credential with less than 8 characters username and password to see if the script can handle the restriction properly. **Preconditions:** database exists, menu options exists.		
Steps	**Expected response or output**	**Actual response or output**
Enter URL/resource name: VU Enter username: Pipe Enter password: Wong	Information message “Input length for username and password must be at least 8 characters. Please try again”	Information message “Credential added successfully!”
Open data base with text editor	There shouldn’t be any data recorded as any input of usernames and passwords with less than 8 characters shouldn’t be a successful recorded.	URL: VU Username: Pipe Password: Wong
Code modification	Add # Check if inputs are at least 8 characters	

	```
	if len(username) < 8 or len(password) < 8:
	    print("Input length for usernames and passwords must be at least 8 characters. Please try again.")
	    continue  # Continue to the menu
	```

Task 1.7 Confirm completion

- Contact your instructor, who is acting as your supervisor, when your script is complete to confirm that the script requirements and outcomes have been met. Make adjustments to the code if requested by your supervisor. Document this discussion via bullet points or brief notes below.

Enter notes of

The discussion with your supervisor regarding script requirements, outcomes and any required modifications

The overall script is good and meet all script requirements. Thus, there is no need for further required modification

However, beyond the script requirements, the script could have possible improvements on:

1. Implementing encryption to protect passwords as the current script does not provide any encryption for credentials
2. Implementing more error handling, such as handle file-related errors and input validation more thoroughly
3. Implementing a more permanent data storage solution, as currently, all data is lost when the script is closed
4. Implement security policies for the password manager, such as password complexity requirements, or the ability to change or delete stored credentials.