### ****Assignment 2****

**Name: Kamal**

**Student id: 100990948**

**Course Name: Scripting**

**Course Code: COSC 1104-01**

**Date: 15-11-2024**

**Professor: Sohaib Mohiuddin**

### ****Part 1: Identify the Problem****

**Problem 1:** Date-logging expense tracker   
  
Many people find it difficult to manage their money well, especially when it comes to monitoring their everyday expenses. One solution to this problem is to create a simple application for tracking spending that not only records expenditures but also the date they occurred. By integrating the datetime library, the application may instantly capture the current date when an expense is added.   
  
  
  
Usefulness: This course may be very beneficial for students, working professionals, and anybody else looking to enhance their financial management. Maintaining a date-based spending log enables consumers to:   
  
Analyze expenditure patterns over a specified period of time (weekly or monthly, for example).   
  
  
  
• To cut down on unnecessary spending, identify areas of excessive spending, and make informed decisions.   
  
  
  
• Produce accurate financial plans or budgets

Many people find it difficult to manage their money well, especially when it comes to monitoring their everyday expenses. One solution to this problem is to create a simple application for tracking spending that not only records expenditures but also the date they occurred. By integrating the datetime library, the application may instantly capture the current date when an expense is added.   
  
  
  
Usefulness: This course may be very beneficial for students, working professionals, and anybody else looking to enhance their financial management. Maintaining a date-based spending log enables consumers to:   
  
Analyze expenditure patterns over a specified period of time (weekly or monthly, for example).   
  
  
  
• To cut down on unnecessary spending, identify areas of excessive spending, and make informed decisions.   
  
  
  
• Produce accurate financial plans or budgets

**Python Libraries:**

* datetime: To record the date on which each expense is logged. This ensures accurate tracking of expenses over time without requiring manual date entry from the user.

#### ****Problem 2: Randomized Quiz Generator****

Learning through practice is an essential aspect of education, and quizzes are a popular tool for reinforcing knowledge. However, static quizzes can become repetitive and fail to engage users after a few attempts. A solution to this issue is a randomized quiz generator, which selects a subset of questions from a larger pool. Using the random library, the program can shuffle and present questions in a unique order each time it runs, ensuring a dynamic and engaging experience for users.

**Usefulness:**  
This program is valuable for educators, students, and anyone preparing for exams or interviews. The benefits include:

* **For educators:** They can quickly generate unique quizzes for students without manually creating multiple versions.
* **For students:** It provides a fresh learning experience by offering varied question sets each time, reducing monotony.
* **For self-learners:** It allows individuals to test their knowledge in an engaging way and identify areas for improvement.

For instance, a student preparing for a coding interview might use this program to practice multiple-choice questions about programming concepts. The randomness ensures that they are exposed to all possible questions in the pool over multiple quiz attempts.

**Difficulty:**  
This program involves several moderately challenging elements:

* Creating a question pool with options and correct answers.
* Randomly selecting and shuffling questions using the random library.
* Managing user inputs to ensure valid responses (e.g., selecting an option from 1 to 4).
* Keeping track of scores and providing feedback on user performance.

**Python Libraries:**

* random: To shuffle the questions and ensure that a different set of questions is presented in a unique order during each quiz session. This randomization keeps the program engaging and effective as a learning tool.

Part 2: Github link

<https://github.com/Kamal5755/Assignment-2>

### ****Part 3: Reflection****

### ****Reflection on Problem 1: Task Deadline Tracker****

**Was the problem suitably challenging?**  
  
Indeed, the challenge was appropriately difficult since it called for integrating several programming ideas, including basic data storage, date manipulation, and user input management. I was forced to consider how to efficiently organize the code when I designed a tool that lets users add tasks with deadlines, determine how much time is left for each work, and present the results in an easy-to-use style. Making the application user-friendly and accessible was another difficulty that complicated the issue. This task was interesting and instructive because it needed careful consideration of edge circumstances, such as overlapping dates or invalid inputs.

**What was the most challenging aspect?**  
Managing incorrect date formats and making sure the application could gracefully handle user failures were the most difficult parts. For instance, users may enter dates in forms "2024/11/15" rather than "15-11-2024" that do not follow the intended syntax. It was challenging to implement strong validation logic that would identify and alert users to invalid formats without causing the program to fail. Managing situations where a task's deadline has already gone and making sure the application informed the user of this explicitly was another difficulty. Last but not least, arranging the deadlines chronologically for display called for more reasoning, but it was worthwhile to improve the program's usefulness.

**What did you learn?**  
I now have a better understanding of the datetime module, particularly how to format, parse, and compare dates, thanks to this assignment. I discovered that organizing a program into several functions made it simpler to comprehend, test, and maintain the code. I also learned from this experience how crucial error-handling and user-friendly design are to producing dependable software. I also came to see how crucial it is to predict user behavior and build the program to elegantly handle mistakes or unforeseen inputs.

**How confident are you that it works reliably?**  
I used a range of inputs to test the program, such as tasks with overlapping deadlines, valid and invalid dates, and edge cases such tasks that were due on the current date. The program worked as planned in every situation, giving the user clear feedback and precise computations. I am confident in its dependability because of the thorough testing. I do admit, though, that extra testing with a bigger dataset or odd inputs can turn up more edge cases that could be fixed in later versions. All things considered, I have no doubt that the program is dependable and operates effectively within the anticipated range of use.