Emmanuel Cabili Cuyugan
387094
29/01/2024

In Partial Fulfilment of The University of Law’s:
MSc - Computer Science - Software Development - Online FT - Sep 23 Start - October 2023
Assessment 1: Coursework Portfolio

Project Title: Python Budget Tracker

Abstract:

The Python Budget Tracker is an expense management application, functioning as a comprehensive demonstration of the developer’s skill in Python programming, problem-solving ability, and knack for producing functional software. Through the use of a modular and stylized approach, the application streamlines the process of entering and categorizing expenses and is able to summarize large quantities of data, providing a real-time view of a user’s financial standing.

Key features include an intuitive method for categorizing expenses, robust error handling to ensure input is correctly validated, and a complex budget analysis that provides a total expense, remaining budget, and average daily spend. A number of software versioning tools are also being used to demonstrate proper collaborative development. This project is indicative of their passion for creating a financial software that is user-centric and a balance of technical mastery and utility.

Introduction

The Python Budget Tracker is an expense management application developed by myself, Emmanuel Cuyugan, as a way to create a way to track spending, to stay within budget, and make informed financial decisions. This functional software solution also serves as a project submission to showcase my proficiency in programming and to demonstrate his problem solving skills.

Requirements:

This project seeks to assess proficiency in Python programming, problem-solving skills, and the ability to create a functional software solution. Key Requirements identified and attempted include:

Displaying Development Environment Competence:

This program has been built using a development environment to implement functioning code. Writing on VSCode, I have attempted to use the environment to ensure code modularity, readability, and scalability within the program for a seamless development experience.

Utilization of Software Versioning Tools:

GitHub has been selected as a software versioning tool to showcase proficiency and understanding in deployment and version control. This project has been deployed to GitHub, accessible through my, Emmanuel's, GitHub account, @emmcygn

Code Modularity:

In the writing of this program, a modular approach to writing the program was selected as in the development process, it was found that writing and testing individual functions part by part was the most effective way to ensure code quality and scalability. This program also utilises modular libraries such as calendar and datetime and implements stylistic elements to increase text readability within the program.

While writing this code, I found that attempting to build the whole application from "top to bottom" wasn't the best approach. Frequent dead ends were met as I tried to wrangle multiple portions of the code at once. It was only when I built an outline of the different parts that I was able to build parts out based on the understanding gained from the portions built out.

User Input and Error Handling:

I anticipated errors in user input within the application. As a result, some loops have been built in to prompt for a correct input until the criteria has been satisfied. For instance, when entering expense amounts, the system uses a loop to continuously prompt the user until a valid float is provided. This ensures that the application does not crash due to invalid input.

Some difficulty was met in this portion which taught me the importance of appropriate indentation. Some tools that helped me was actually using VScodes built in debugging suggestions.

Expense Categorization and Tracking:

While there may be different categories for expenses for each user, simplicity in options would be the best way to proceed. In this application, users are prompted to select a category for their expenses from a predefined list. The code incorporates nested loops and exception handling to ensure that users input valid category numbers. The selected expense is then saved to a file, maintaining a record of all user transactions. If a user should wish to edit the categories, one can simply enter the code and change the entry names in lines **58-62** of the expensetracker.py file.

The categories I selected are what I believe are most appropriate for the time being. Important to note is that this code can be modified to accept more categories by adding to the list in lines 58-62.

Expense Summarization and Budget Analysis:

By reading the expense file, the program then summarizes expenses by category and provides an overview of spending and the budget.

Included in the summary are total expenses, remaining budget, average daily spend, and days remaining in the month. This was built in to help the user get a clearer picture of their financial position relative to their budget for that month. A month's timeline was selected as it is the most commonly used timeframe for budgeting. This code will be improved in the future with a weekly budget planner or a customisable timeframe planner.

Using the calendar and datetime modules were crucial to this portion, and basic mathematical equations were used in conjunction to provide the crucial user information.

User Prompted Rerun Option:

After the pplication processes the data, it then prompts users if they would like to add more expenses to the tracker. The loop function enables the user to rerun the application as many times as needed to log in the additional expenses, leading to a user-friendly and interactive experience.

This was the last portion of the code that I wrote, and gave me a better appreciation and understanding of how to loop entire functions and programs.

Data Storage:

In order to store user data, the application creates a file named expenses.csv within the same folder as the application file. The application then writes to this csv file after input. The advantages of using a CSV file are that it can be easily exported and formatted to be viewed on other platforms and formats.

For data deletion, simply delete an entry within the CSV file for individual lines and delete the file to clear the program.

Testing

Phase	**Input**	**Intended Output**	**Actual Output**
Load Libraries and Style Functions	Run application	Calendars load, Font colours and styles load	Calendars load, Font colours and styles load *1
Input expense name	Type in expense name	Proceeds to next step	Proceeds to next step
Input expense amount	Type in expense amount	Proceeds to next step	Proceeds to next step
Select category	Select category by typing integer between 1-5	Proceeds to next step	Proceeds to next step
Create new expense object	Input from select category step	Create object	Create object
Write object to csv	Create object	Write object to csv	Write object to csv
Read Lines	Input from select category step and object creation	Read objects in CSV	Read objects in CSV
Summarise Expenses	Read objects in CSV	Summarise contents of CSV	Summarise contents of CSV
Print Summary	Read and summarise objects in CSV finish	Print Summary in terminal	Print Summary in terminal
Rerun/Terminate Loop	Input either yes or no answer to prompt asking to rerun script	Terminate or loop function, depending on input	Terminate or loop function, depending on input

***appendix items**

Conclusion

This was the first application of this scale that I wrote, and I am very pleased with how I got to combine a lot of the Python basics I learned over the course of this term. Future improvements to the code will be a way to rewrite the categories within the terminal and to delete/modify entries. Further development will be integrations via API to web based data storage.