CA278 Python Project

Money Manager App

Group Members
Jonathan Clear - 18432192
Shane Duffy- 18371411

Problem Description

Money Manager is run through the command line. In advance to running the program, the user should make a text file to a name they wish and a list of shops they've shopped in and what they've spent in each relevant shop i.e.
[expenses.txt]

lidl 2.1
dominos 50.13
irish rail 9.6
caffe nero 3.4
wetherspoons 12.10
tesco 34.90
bombay pantry 23.5

Once this file has been created the user is ready to run the program. They will be prompted to enter their name and their salary. The program will check the user has entered their salary correctly by checking if it is an integer. If the program detects it's not an integer the user will again be asked to enter his/her salary. They will then enter the name of the file they have filled in their expenses. The program will then read in this data and use it for further analysis. The program shops.py contains a mini database with all the different outlets i.e groceries, travel, takeaways, restaurants with all the different types.

The program will be able to read in the data and detect the most expensive shop, the total money spent, the cheapest shop and the different amounts spent in each different business. I.e You spent 120.30 in groceries, you spent 80.20 in takeaways etc.
This will be outputted on the command line but also a file called money_manager.txt will also be created so all of this data will stay on record for future reference. Each day the program the data will be appended to the file so it will not corrupt previous data.
This will make it extremely easy for the user to keep on track with their spending as they can easily see if they have to spend less or more than the previous months

Algorithm

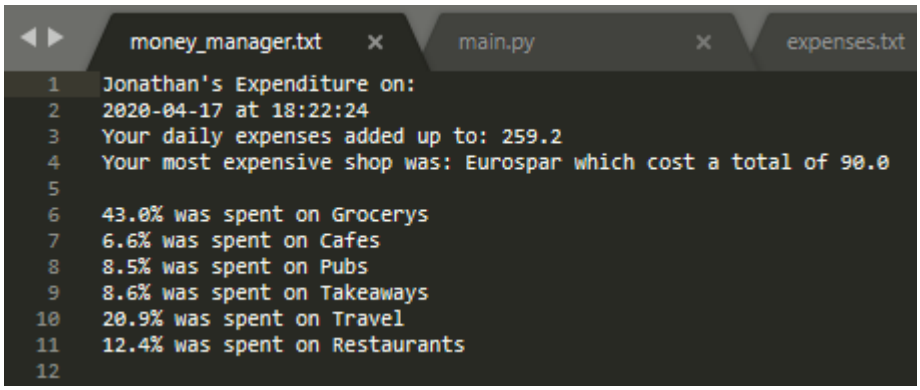
The algorithm goes as follows:
Gets users name
Asks for the users' salary
If salary is an integer:
    The program continues
Else:
    Tells the user that the inputted value is not valid and asks them to enter the value again
Asks the user the name of the file that contains their expenses
If file exists
    The file is then imported
Else:
    Returns error message and program ends

The program then opens the file and converts all numbers into floats
Six dictionaries are created: grocerys, cafes, pubs, takeaways, travel, restaurants to calculate the sum of money spent in each category
Shops_dic is imported from the shops.py which contains a mini database of shops
A key is used to sort the shops into their relevant dictionary
The shop where the most money spent is calculated
The total money spent for that day is calculated
The percentage of money spent in each category is calculated
The file money_manager.txt is opened and the following is written
    The users' name
    The days' date
    Total of that days' spending
    The most expensive shop and what was spent there
    The percentages of what categories where money was spent in
That file is then closed and stored
The following is then printed to the screen:
    Whether the user is over or under budget
    The percentages of what category the most money was spent in
    Total of that days' spending
    The most expensive shop and what was spent there

Program Design

We decided to split the program into a number of separate sections. The first is main.py which is where all the calculations and formatting will be done.
Shops.py is a separate file which will provide a mini database with the different types of shops and their names, this will be in the form of a dictionary and will be quiet long so it would clean up the main.py if we put it into a separate file and could then import it when needed. T

There is then a text file the user creates them self, this can be called whatever they wish however they must input the shops and the price they spent in the same row. The user will be prompted to enter in the name they called this file and main.py will import and use that data provided for further analysis.



The output of these calculations made in main.py will be outputted to the command line for the user to see instantly however they will also be written and backed up to a file called money_manager.txt. This will be a great tool for the user to review their spending at the end of the week, month, year etc.

Because this program is heavily reliant on the user's input being entered incorrectly, we decided rather than the program failing on errors to include try and except function. This is used for the salary function if the user mistypes the program will be able to detect this and ask the user to enter in the correct format (i.e an integer). This makes the program as user friendly as possible as mistakes are likely to be made

Individual Member Contribution
Together - Brainstorming ideas, concluding final project, delegation of workload

Jonathan Clear – Importing shops.py and converting into separate dictionaries, implanted try and except function, program design in report file

Shane Duffy – Testing; making sure the output was to the format required, taking in user input, creating shops.py, algorithm in report file