Logo

Description automatically generated

**ANL 252**

**Python for Data Analytics**

**Tutor-Marked Assignment**

**(TMA01)**

Ong Sheng Da  
Z2281575

Prepared for: Professor Kumar Munish

Singapore University of Social Sciences

15 September 2023

Question 1 (a)

With the advent rise in coding plagiarism, we need to look at why it happens and how we can avoid it.

Why plagiarism happens:

* Lack of understanding (Lee, 2023), student may not be able to cope with the curriculum and find it difficult to understand complex topics which can pressure them into seeking others work.
* Pressure to succeed (Bailey, 2020), academic pressure and competition can tempt students to take shortcuts.
* Easy access (Lee, 2023), open-source code culture allows students to have easy access to a wide variety of solutions and codes.
* Fear of failure (Bailey, 2020), fear of failure or low grades can drive students to plagiarism as a way to guarantee a passing grade.

How to avoid plagiarism:

* Mastery of topic (Lee, 2023), students with a deep understanding of topic are more like to be capable of creating original content as they are more able to express their ideas.
* Cite the source (Linkedin, 2023), students should be taught how to properly site sources for which they reference their work from.
* Having very specific questions and problems to solve, the more specific the question the harder it is to find similar code online, decreasing the chances of students resorting to plagiarism.

(199 words, including in-text citations)

Question 1 (b)

#Expense tracker

#Create dictionary for the type of expenses

expenses = {'food': 0, 'transport': 0, 'others': 0}

#Select choice

while True:

print("Expense Tracker Menu:")

print("1. Add Expense")

print("2. View Expenses")

print("3. Exit")

choice = int(input("Enter your choice (1/2/3): "))

#Add expenese amount

if choice == 1:

type = input(f'Enter expense type {list(expenses.keys())}: ').lower()

if type in expenses:

try:

amount = float(input("Enter expense amount: "))

expenses[type] += amount

print(f"{amount} added to {type} expenses.")

except ValueError:

print("Invalid input. Please enter a valid amount.")

else:

print(f"Invalid type. Choose from {list(expenses.keys())}.")

#View summary of expense report

elif choice == 2:

print("Expense report:")

for type, amount in expenses.items():

print(f"{type.capitalize()}: ${amount:.2f}")

total = sum(expenses.values())

print(f"Total Expenses: ${total:.2f}")

#Exit expenses tracker

elif choice == 3:

print("Exiting Expense Tracker.")

break

else:

print("Invalid choice. Please choose from 1, 2, or 3.")

(29 lines, not including spaces and comments)

This code was written by me. The purpose of this code is to track your expenses. From the user’s perspective, you would first have to input 1,2 or 3 for adding new expenses, viewing expenses summary report or exiting the expense tracker respectively. If option 1 was to be chosen, you will need to input what type of expenses it is. I have included only 3 types of expenses namely “food”, “transport” and “others”. Afterwards, you will have to input the amount and the program will bring you back to the beginning where you are given the 3 options to add new expenses, view expenses summary report or exit. If at any point of time an invalid amount or expenses type was inputted, the program will bring you back to the beginning with where you will be given the 3 options.

If option 2 was chosen, a summary report of all the expenses you input previously will be generated. It will summarise the total amount in each of the three expenses type and sum of all three expenses.

Finally if option 3 was chosen, the program will simple end.

(183 words)

Question 1 (c)

#Expense tracker

#Create dictionary for the type of expenses

expenses = {'food': 0, 'transport': 0, 'others': 0}

#Select choice

while True:

print("Expense Tracker Menu:")

print("1. Add Expense")

print("2. View Expenses")

print("3. Exit")

choice = int(input("Enter your choice (1/2/3): "))

#Add expenese amount

if choice == 1:

type = input(f'Enter expense type {list(expenses.keys())}: ').lower()

if type in expenses:

try:

amount = float(input("Enter expense amount: "))

expenses[type] += amount

print(f"{amount} added to {type} expenses.")

except ValueError:

print("Invalid input. Please enter a valid amount.")

else:

print(f"Invalid type. Choose from {list(expenses.keys())}.")

#View summary of expense report

elif choice == 2:

print("Expense report:")

for type, amount in expenses.items():

print(f"{type.capitalize()}: ${amount:.2f}")

total = sum(expenses.values())

print(f"Total Expenses: ${total:.2f}")

#Exit expenses tracker

elif choice == 3:

print("Exiting Expense Tracker.")

break

else:

print("Invalid choice. Please choose from 1, 2, or 3.")

**Output:**

Expense Tracker Menu:

1. Add Expense

2. View Expenses

3. Exit

Enter your choice (1/2/3): 1

Enter expense type ['food', 'transport', 'others']: food

Enter expense amount: 5863.25

5863.25 added to food expenses.

Expense Tracker Menu:

1. Add Expense

2. View Expenses

3. Exit

Enter your choice (1/2/3): 1

Enter expense type ['food', 'transport', 'others']: others

Enter expense amount: 6589

6589.0 added to others expenses.

Expense Tracker Menu:

1. Add Expense

2. View Expenses

3. Exit

Enter your choice (1/2/3): 2

Expense report:

Food: $5863.25

Transport: $0.00

Others: $6589.00

Total Expenses: $12452.25

Expense Tracker Menu:

1. Add Expense

2. View Expenses

3. Exit

Enter your choice (1/2/3): 3

Exiting Expense Tracker.

Lines include comments and spaces.

Line 3: I used dictionary instead of list and tuples because I want to be able to categorise the different expenses and their total price.

Line 6: I used the **while True** loop because I want the user to have a more reliable way to exit the program rather than having a specific condition.

Line 16 and 23: I used to **if else** function in the scenario when the user input the wrong value, this will allow the user to try inputting again rather than ending the while loop.

Line 15 and 24: I want to make the question more dynamic but don’t want “dict\_keys” as part of the question so I converted it to a list when asking the question.

Line 17 to 22: I used the try block for the amount input, this way if a user enters any value that may cause on error, the users can try to input the value again.

Line 18: **float** was used in case the user wants to input a decimal amount.

Line 27: **elif** was used for the case where the user input choice 2 or 3.

Line 29: f**or** loop was used to generate a summary report that shows all the types of expense and their corresponding amounts.

Line 30 and 31: I capitalize the type, so that it would look nicer when it is printed, and I limited the amount to 2 decimal places.

Line 31: a sum of all the amount is included for the user.

Line 14 to 39: if else was in case the user enters anything besides choice 1, 2 or 3 to minimise chance of code failing due to user error.

283 words

Question 2



Appendix 1

With reference to Appendix 1, here are the 3 ways to improve the code’s reliability, readability, and maintainability.

Reliability:

* Use **.lower()** function at the end of string inputs, thus minimise the lucky hood that the program will stop if the user enters anything else besides ‘yes’ such as ‘Yes’.
* In line 10, instead of having **break** which will abruptly stop the program if the user input an invalid value, the program should try to loop back to the beginning of the code.
* When allowing user to re-enter their input, tell the users what they can input thus reducing the change of the code failing due to user error. Refer to new code below.

Readability:

* Standardise the use of (“) and (‘) to prevent confusion.
* Use descriptive variables so instead of “query” use “continue\_updating”.
* Remove unnecessary steps such as combining lines 12 and 13, refer to new code below.

Maintainability:

* Include comments to explain the purpose of each block of code (Ramos, 2023).
* Forsee any errors in the code and ensure that code can fail gracefully. For example, if user inputs a string for “price\_of\_item”.
* Instead of generic variable names such as “item”, you can use “selected\_item”.

(195 words, not including citations)

New code

# Create list for types of products

products = ['laptop','mouse','webcam','keyboard','speaker']

continue\_shopping = 'yes'

updated\_items = []

# Main code

while continue\_shopping == 'yes':

selected\_item = str(input(f'Hello! What do you want to buy{products}?')).lower()

# Determine if user input match products in list

if selected\_item not in products:

print(f'Invalid input please choose from: {products}.')

continue

# Get price of product

while True:

try:

item\_price = float(input(f'How much is the {selected\_item} (in SGD)?'))

break

except ValueError:

print("Please enter a valid amount.")

updated\_items.append([selected\_item, item\_price])

print(f'This is our updated shopping list: {updated\_items}')

continue\_shopping = str(input('Would you like to continue? (yes/no)')).lower()

(26 lines including spaces and comments)

Reference

Bailey, J. (2020). Why do students commit plagiarism? *Plagiarism Today*. https://www.plagiarismtoday.com/2020/07/09/why-do-students-commit-plagiarism/

Ramos, L. P. (2023). Python Constants: Improve your code’s maintainability. *realpython.com*. https://realpython.com/python-constants/

Lee, C. (2023, August 31). What is Programming Plagiarism? Why Is It on the Rise? *Turnitin*. https://www.turnitin.com/blog/what-is-programming-plagiarism-why-is-it-on-the-rise

Linkedin. (2023). What’s your process for avoiding plagiarism when using code from other sources? *www.linkedin.com*. https://www.linkedin.com/advice/1/whats-your-process-avoiding-plagiarism-when-using