Python For Data Analytics (ANL252)

JULY SEMESTER 2023

TMA01

Adam Abdullah Ang

PI No.: H2210484

Submission Date: 14 SEPTEMBER 2023

1.a)

* Functions like mathematical formulas are commonly used in programs. In order to simplify the process of coding, libraries were created to shorten the time for this process so they would not need to type the same code and could just call a function.

* Online forums which share codes, from the standpoint of learning and teaching, students will be bound to copy the code. To avoid these, students need to understand how the code functions and how to attune it to their program. These changes will help them avoid plagiarism, but at the same time, they are required to cite and give credit to the source of the code.

* We can promote self-discovery by encouraging the students to make errors and guiding them to solve the issue uniquely. It will enhance the students' understanding and develop their style of writing codes that might be innovative in their future projects. (163 Words)

b)

tasks = []

while True:

print("\nOptions:")

print("1. Add Task")

print("2. View Tasks")

print("3. Quit")

choice = input("Enter your choice: ")

if choice == "1":

task = input("Enter the task: ")

tasks.append(task)

print(f"Added task: {task}")

elif choice == "2":

print("To-Do List:")

for i, task in enumerate(tasks, start=1):

print(f"{i}. {task}")

elif choice == "3":

print("Exiting the to-do list application. Goodbye!")

break

else:

print("Invalid choice. Please select a valid option.")

This code does a simple task of making a to do list. I found this short code useful and every day we would have a to do list. Whether for work or for grocery shopping, its usability is for all. And from this short program we could input more variables and change the program to further help us, depending on our needs such as adding a time counter or deadline reminder.

c)

# Initialize an empty list for tasks

tasks = []

task\_time = []

while True:

print("\nOptions:")

print("1. Add Task")

print("2. View Tasks")

print("3. Remove Task")

print("4. Quit")

decision = input("\nPlease select the number for action: ")

#Input task you want to add

if decision == "1":

task = input("Enter the task: ")

tasks.append(task)

task\_time = input("Enter the task Time: ")

print(f"Added task: {task} @ {task\_time}")

#View To-do list

elif decision == "2":

print("\nTo-Do List:")

if len(tasks) == 0: #To check if To-Do list is empty

print("\n\*\*To-Do List is Empty\*\*")

else:

for i, task in enumerate(tasks, start=1):

print(f"{i}. {task} @ {task\_time}")

#Remove items from the To-Do List

elif decision == "3":

task\_index = int(input("\nWhich task do you want remove from To-Do List: "))

if 1 <= task\_index <= len(tasks): #To check if there is any task on the list

removed\_task = tasks.pop(task\_index - 1)

print(f"Removed task: {removed\_task} @ {task\_time} from To-Do List")

else:

print("\*\*Task number does not exist\*\*")

#Exit Loop and to check if Invalid number for action was made

elif decision == "4":

print("\nExiting the To-Do list application. Goodbye!")

break

else:

print("Invalid choice. Please select a valid option.")

**Output**

Select the number for action:

Options:

1. Add Task

2. View Tasks

3. Remove Task

4. Quit

* The choice variables were changed to decisions instead. It was appropriate because it prompts the user to choose which action to execute. I amended the output to "Please select the number for action:" to make it easier for the user to understand. Some might not understand "Enter your choice" to make it more straightforward so all users can understand.
* Time was added so that the user can be more specific at completion. The user can track the task better and the deadlines. Adding time allows the user to manage things efficiently, improving productivity.
* Changed the code for "2. View Task" and added an if condition to check whether the To-Do List was empty. The previous version did not address the matter and will prompt the same question without informing the user whether the list was empty. Hence, I wanted the user to be more aware and know there is no task in the To-Do list.
* Added a remove task function so that users can remove the task they have completed or might have missed keyed. The previous program did not allow users to amend or change the input they had keyed. Adding this feature will allow the user to remove the task they have completed at any given time. (209 Words)

2.

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

products\_str = ', '.join(products)

updated\_items = []

while True:

print(f'We have a list of products here: {products\_str}.')

item = str(input("Hello! What do you want to buy? "))

if item not in products:

print(f'Wrong product! Please try again.')

else:

price\_of\_item = input('How much is it (in SGD)? ')

entered\_input = [item, price\_of\_item]

query = input('Would you like to continue? (yes/no)')

if query == 'no':

updated\_items.append(entered\_input)

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

break

elif query == 'yes':

updated\_items.append(entered\_input)

else:

print('Invalid Response')

1. To improve readability and maintainability, it should be consistent throughout, and I am referring to the double and single quotations used in the program. It might not be a significant issue, but there should be a consistent use throughout your program. If you open a string with a single quotation, it should be the case throughout.
2. Single quotation marks on the different products that can be bought were removed, and the ‘f’ for strings that have expressions as it is redundant for better readability.
3. Restructure the while loop to make it loop infinitely and remove the condition that was in place. The reason is that I want the program to run till the user selects no. The idea of this program is to approach it like a checkout feature. The query declared above was removed as we should not declare ‘yes’ when the user is prompted to input.
4. Improving maintainability by segmenting the conditions needed, for example, when prompting to continue whether ‘yes’ or ‘no’. Also, for error handling like mismatch characters, which are not what should be inputted so that the program would not just end. (191 Words)

**References**

1. *The python tutorial. Python documentation. (n.d.-b).* [*https://docs.python.org/3/tutorial/index.html*](https://docs.python.org/3/tutorial/index.html)
2. *Python tutorial. (n.d.).* [*https://www.w3schools.com/python/*](https://www.w3schools.com/python/)
3. *Wu, K. Y., & Zhu, S. (2023). ANL252 Python for data analytics. Singapore University of Social Sciences*.