Logo, company name

Description automatically generated

**ANL252  
Python for Data Analytics**

**Tutor-Marked Assignment**

**July 2023 Presentation**

**Submitted by:**

|  |  |
| --- | --- |
| **Name** | **PI No.** |
| **Muhammad Yusuf Bin Zainudin** | **Q2111756** |

**Tutorial Group : ­­­­­­­­­­­­T03**

**Instructor’s Name : Mr. Munish Kumar**

**Submission Date : 15/09/2023, 2355 hrs**

1.

a)

* Plagiarism in coding happens for a number of reasons. A reason could be due to the ease of having access to codes from online sources. Examples of these sources are Stack Overflow and GitHub (Copyleaks, 2021).
* Another reason that could cause a person to copy codes is due to laziness and time constraints. By copying codes, the individual can save a lot of time without having to code from scratch and spend a lot of time debugging their own codes.
* One way to avoid plagiarism is by using plagiarism checkers (Copyleaks, 2021). Plagiarism checkers are able to look for partial or even true plagiarism in codes by looking for similarities (Copyleaks, 2021).
* Another way to avoid plagiarism is to either directly ask for permission from the programmer whose code you intend to use or to cite the source in which you are referencing from. You can do so by adding comments in your code, including a README text file or a reference list (LinkedIn, n.d.).

b)

print("""You enter a dark room with two doors.

Do you go through door #1 or door #2?""")

door = input("> ")

if door == "1":

print("There's a giant bear here eating a cheese cake.")

print("What do you do?")

print("1. Take the cake.")

print("2. Scream at the bear.")

bear = input("> ")

if bear == "1":

print("The bear eats your face off. Good job!")

elif bear == "2":

print("The bear eats your legs off. Good job!")

else:

print(f"Well, doing {bear} is probably better.")

print("Bear runs away.")

elif door == "2":

print("You stare into the endless abyss at Cthulhu's retina.")

print("1. Blueberries.")

print("2. Yellow jacket clothespins.")

print("3. Understanding revolvers yelling melodies.")

insanity = input("> ")

if insanity == "1" or insanity == "2":

print("Your body survives powered by a mind of jello.")

print("Good job!")

else:

print("The insanity rots your eyes into a pool of muck.")

print("Good job!")

else:

print("You stumble around and fall on a knife and die. Good job!")

(Shaw, 2017, p. 108)

The above code was taken off the course textbook “Learn Python 3 The Hard Way” from page 108. The code is made to ask the user questions and make decisions based on their inputs by making use of if, else and elif conditions. (Shaw, 2017, p. 108). This code can be used in various situations where the user is having difficulty coming to a decision and thus, can rely on the code to come to a decision by prompting multiple questions that can lead to many different decisions. The idea behind this code can be applied in various real life situations. An example would be tutorials/troubleshooting where users can input what type of issue they are facing and what steps they have taken and what steps could be taken next. Another example could be personality tests where a user’s answers will lead to specific types of questions and therefore, different results. Another example could also be a diagnosis for an illness by prompting questions regarding what the user is feeling and facing. Such automation could increase efficiency and reduce time spent on questions that are frequently asked in any given field such as medicine, finance and more.

(197 words)

c)

def main():

print("You enter a dark room with two doors.\n"

"Do you go through door #1 or door #2?")

door = input("> ")

if door == "1":

bear\_room()

elif door == "2":

trippy\_room()

else:

death()

def present\_choices(prompt, choices):

print(prompt)

for i, choice in enumerate(choices, start=1):

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

user\_choice = input("> ")

return user\_choice

def bear\_room():

prompt = "You find yourself in a room with a giant bear eating a cheese cake."

choices = [

"Take the cake.",

"Scream at the bear",

]

consequences = [

"The bear eats your face off. Good job!",

"The bear eats your legs off. Good job!",

]

user\_choice = present\_choices(prompt, choices)

choice\_index = int(user\_choice) - 1

if 0 <= choice\_index < len(consequences):

print(consequences[choice\_index])

else:

print(f"Well, doing {user\_choice} is probably better.\n"

"The bear runs away.")

def trippy\_room():

prompt = "You stare into the endless abyss at Cthulhu's retina."

choices = ["Blueberries.",

"Yellow jacket clothespins.",

"Understanding revolvers yelling melodies",

]

consequences = [

"Your body survives powered by a mind of jello. Good job!",

"Your body survives powered by a mind of jello. Good job!",

"The insanity rots your eyes into a pool of muck. Good job!",

]

user\_choice = present\_choices(prompt, choices)

choice\_index = int(user\_choice) - 1

if 0 <= choice\_index < len(consequences):

print(consequences[choice\_index])

else:

print("The insanity rots your eyes into a pool of muck. Good job!")

def death():

print("You stumble around and fall on a knife and die. Good job!")

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output 1:**

You enter a dark room with two doors.

Do you go through door #1 or door #2?

> 1

You find yourself in a room with a giant bear eating a cheese cake.

1. Take the cake.

2. Scream at the bear

> 1

The bear eats your face off. Good job!

**Output 2:**

You enter a dark room with two doors.

Do you go through door #1 or door #2?

> 1

You find yourself in a room with a giant bear eating a cheese cake.

1. Take the cake.

2. Scream at the bear

> 2

The bear eats your legs off. Good job!

**Output 3:**

You enter a dark room with two doors.

Do you go through door #1 or door #2?

> 1

You find yourself in a room with a giant bear eating a cheese cake.

1. Take the cake.

2. Scream at the bear

> 3

Well, doing 3 is probably better.

The bear runs away.

**Output 4:**

You enter a dark room with two doors.

Do you go through door #1 or door #2?

> 2

You stare into the endless abyss at Cthulhu's retina.

1. Blueberries.

2. Yellow jacket clothespins.

3. Understanding revolvers yelling melodies

> 1

Your body survives powered by a mind of jello. Good job!

**Output 5:**

You enter a dark room with two doors.

Do you go through door #1 or door #2?

> 2

You stare into the endless abyss at Cthulhu's retina.

1. Blueberries.

2. Yellow jacket clothespins.

3. Understanding revolvers yelling melodies

> 2

Your body survives powered by a mind of jello. Good job!

**Output 6:**

You enter a dark room with two doors.

Do you go through door #1 or door #2?

> 2

You stare into the endless abyss at Cthulhu's retina.

1. Blueberries.

2. Yellow jacket clothespins.

3. Understanding revolvers yelling melodies

> 3

The insanity rots your eyes into a pool of muck. Good job!

**Output 7:**

You enter a dark room with two doors.

Do you go through door #1 or door #2?

> 2

You stare into the endless abyss at Cthulhu's retina.

1. Blueberries.

2. Yellow jacket clothespins.

3. Understanding revolvers yelling melodies

> 4

The insanity rots your eyes into a pool of muck. Good job!

**Output 8:**

You enter a dark room with two doors.

Do you go through door #1 or door #2?

> 0

You stumble around and fall on a knife and die. Good job!

* Firstly, I have rewritten the code to be more flexible by making it easier for the programmer to include more “choices” and their corresponding “consequences”. This is done by making “choices” and “consequences” into lists that are neater, easier to read and easier to add on to. This improves scalability, readability and maintainability.
* Secondly, despite the code being slightly longer than the initial code, the changes implemented improve efficiency tremendously. For example, in the initial code, the print() function is being repeated multiple times and this will worsen should the programmer need to add more options. However, in the new code, the print() function is used less times despite being longer.
* Thirdly, I have rewritten the code to use multiple functions that can be reused within the code without typing everything out again. Examples are “def present\_choices,” “def bear\_room” and “def trippy\_room”. Reusable functions make the code neater and easier to scale, maintain or edit.
* Fourthly, the changes promote flow and tidiness. For example, if the user picks “1” from the very first prompt, the subsequent “consequences” and prompts are coded within “def bear\_room”, separated from the “consequences” and prompts from picking “2” at the first prompt, which are coded within “def trippy\_room” instead.

(208 words)

2.

Firstly, the output of the code is messy, affecting the overall quality and readability.

For example,

print(f"We have a list of products here: {products}.")

outputs to,

We have a list of products here: ['laptop', 'mouse', 'webcam', 'keyboard', 'speaker'].

We can improve the code by using,

print(f"We have a list of products here:")

for product in products:

print(product)

which outputs to,

We have a list of products here:

laptop

mouse

webcam

keyboard

speaker

This change in code improves the readability and overall quality, especially if there is going to be addition of more products.

Secondly, if the user inputs an incorrect item, the code ends by showing the shopping list. This is not reliable and not user-friendly as it does not allow any mistakes. For example,

Hello! What do you want to buy?chair

Wrong product! Please try again.

This is our updated shopping list: []

We can improve the code by prompting the user to re-input a correct item. We set up a loop to prompt an input again. Additionally, the user can input ‘done’ if they want to end by going to shopping list. This improves the quality of the code as well as make it more user-friendly.

while query == 'yes':

item = input("Hello! What do you want to buy?\n"

"or enter 'done' to view your shopping list:\n"

)

if item == 'done':

break

if item not in products:

print(f"Invalid product! Please try again.")

continue

Lastly, the original code accepts non-numbers as price, resulting in an inaccurate shopping list. For example,

We have a list of products here: ['laptop', 'mouse', 'webcam', 'keyboard', 'speaker'].

Hello! What do you want to buy?laptop

How much is it (in SGD)?not-a-number

Would you like to continue? (yes/no)no

This is our updated shopping list: [['laptop', 'not-a-number']]

We can improve the code with,

while True:

try:

price\_of\_item = float(input("How much is it (in SGD)?\n"))

break

except ValueError:

print("Invalid input! Please enter numbers only.")

We set up a loop to check if the user keyed in a float. If it is a non-float, the code prompts to re-enter the correct price. This improves accuracy and allows users to correct their mistakes without having to restart the code.

(200 words, excluding lines and outputs in red)

## This is my improved code

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

query = 'yes'

updated\_items = []

print(f"We have a list of products here:")

for product in products:

print(product)

while query == 'yes':

item = input("Hello! What do you want to buy?\n"

"or enter 'done' to view your shopping list:\n"

)

if item == 'done':

break

if item not in products:

print(f"Invalid product! Please try again.")

continue

if item == 'done':

break

while True:

try:

price\_of\_item = float(input("How much is it (in SGD)?\n"))

break

except ValueError:

print("Invalid input! Please enter numbers only.")

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

updated\_items.append(entered\_input)

while True:

query = input("Would you like to continue?\n"

"Yes/No\n"

)

if query in ['yes', 'no']:

break

else:

print("Please input 'yes' or 'no'")

print(f"This is our updated shopping list:")

for item in updated\_items:

print(f"{item[0]} - SGD {item[1]}")

# References

Copyleaks. (2021, July 9). *All You Need to Know About Plagiarism in Coding*. Retrieved September 15, 2023, from Copyleaks: https://copyleaks.com/blog/all-you-need-to-know-about-plagiarism-in-coding

LinkedIn. (n.d.). *What's your process for avoiding plagiarism when using code from other sources?* Retrieved September 15, 2023, from LinkedIn: https://www.linkedin.com/advice/1/whats-your-process-avoiding-plagiarism-when-using

Shaw, Z. A. (2017). *Learn Python 3 The Hard Way.* United States of America: Addison-Wesley.