**Course code: ANL252 Python for Data Analytic**

**Title: Tutor-Marked Assignment TMA 01**

**SUSS PI No.: Y2181412**

**Name: YIEW SHER KYM LYNETTE**

**Submission Date: 15 September 2023**

**Question 1**

(a) 188 words

As per Hattingh et al. (2020), plagiarism happens in coding due to the following reasons:

* It can be tempting to use the information and work available due to an ease of access to them. It is easy to find and pay individuals who are more capable in coding or locate the work of others online.
* The individual is being stretched thinly across multiple commitments resulting in little time left for doing the assignment after fulfilling said commitments.
* The individual does not understand coding and is unable to execute the tasks in the assignment. Some individuals take coding modules to meet graduation requirements instead of being sincere in acquiring the knowledge and skill.

As per ITBriefcase (2019), coding plagiarism can be avoided by:

* Adding comments to the line of codes where necessary as this differentiates the lines adopted from other sources and individual’s unique work.
* Using a plagiarism checking tool for the codes. Turnitin is one such tool to use in SUSS.
* Consulting the instructor so any assistance required to meet tasks objectives can be attained to do the work. This is especially helpful for students without programming background.

(b) 168 words

sec\_sib = True

while sec\_sib == True:

try:

y = int(input("Enter your age: "))

x = 2

if y < 0:

raise ValueError("You're probably not alive. Please be smart and enter an actual age.")

elif y > 100:

raise ValueError("Wow you're old. Skull emoji.")

else:

fir\_sib = y + x

print(f"Now that you are {y} years old, your sibling is {fir\_sib} years old.")

sec\_sib = False

except ValueError as err:

print(err)

This code was inspired by a Math trick question that is of interest to me. It goes like this: When you were 2 years old, your sibling was twice your age. How old is your sibling this year?

This code is able to let users test the question by keying in their age as “2” and getting an answer “4” through a sentence. This answers the question where the sibling, aged 4, is twice of being 2 years old. Users thereafter can try it with other numbers (y) and obtain the age of their sibling (fir\_sib). It is able to filter out the inappropriate values such as numbers outside of the stipulated range, symbols and letters. These are possible using the if-elif-else-condition and ValueError to filter out values and produce response outputs to reply to the inappropriate input. The code only works for numbers 0 to 100.

This code serves an edutainment purpose where users can apply mathematics concepts while being engaged by the output of the code.

(c) 242 words

Rewritten Python code (refer to the next line onwards)

flag = True

while flag:

x = 2

try:

y = int(input("Enter your age: "))

sibling = y + x

if y < 0 or y > 100:

print("That's rare.")

else:

print(f"Your sibling is {sibling} years old.")

flag = False

except ValueError:

print("This must to be an integer. What's your age?")

The output is as follow:

|  |
| --- |
|  |

Enter your age:

Once a number is entered e.g. 9, the output will then be:

Enter your age: 9

Your sibling is 11 years old.

If a negative number such as -5 is entered, the output will be:

Enter your age: -5

That’s rare.

|  |
| --- |
|  |

Enter your age:

If letters or symbols are entered, “This must be an integer. What’s your age?” will be prompted until a number fulfilling the criteria is entered:

Enter your age: hello

This must be an integer. What’s your age?

Enter your age: \*\*\*

This must be an integer. What’s your age?

Enter your age: 9

Your sibling is 11 years old.

As per Sharovar (2022), we need to rewrite codes because:

* it makes it easier for the code to be maintained, updated, or debugged.
* there is a more efficient way to code based on an updated understanding of the task. The code can be structured differently to achieve a similar, if not the same outcome in a more straightforward and succinct manner.
* it facilitates the addition of new features. Sometimes new features are incompatible with the existing code. Including it means the code has to be structured differently while still achieving the desired outputs.

As per Kadosh (2019), rewriting codes enables:

* it to be more legible. Code illegibility is a result of multiple refactors or sloppy initial work.
* new requirements to be met as the assumptions of what the code needs to do has changed.
* the code to be compatible with new technology to remain usable.

Rewriting of codes also allows for

* a review of individual’s understanding of the code.
* retaining only aspects of it that brings value.
* prevention of plagiarising off of the internet.

In summary, rewriting of codes allows for

* the code to be up to date with technology for the purpose of maintenance, updates and debugging.
* linking existing features with new features.
* simplification to ease comprehension of the code’s task.
* the demonstration of one’s understanding of the code’s function.
* code pruning where only useful functions, exceptions, and others of the code are retained in the rewriting.
* the avoidance of plagiarism.

**Question 2** (200 words)

Proposals:

1. Use Dictionary for product items and prices.
   1. **Reliability**: Dictionaries keys are immutable and unique, preventing product name’s accidental duplication or amendments.
   2. **Readability**: Values in key:value pairs are flexible to be integers. If codes require item prices e.g. for summation, the code works. Previously, if other codes uses price\_of\_item (str), there will be an error.
   3. **Maintainability**: Easier for updates.
   4. **Quality**: The code concise product list and self-input prices to Dictionary. Price list information is concisely consolidated.
2. Remove “break”.
   1. **Readability**: Codes without “break” are less complex, making it easier to read and understand.
   2. **Maintenance**: Eliminates the additional effort to identify trigger/s of “break”.
   3. **Reliability**: “break” hinders the code’s expected behaviour through termination. Without “break”, customers can continue choosing available products despite input invalidity.
   4. **Quality**: The code communicates clearly the conditions terminating the loop.
3. Use if-else block.
   1. **Readability**: The code’s intentions are clear.
   2. **Maintenance**: Issues are easier to identify and rectify. Improvements can be easily applied.
   3. **Reliability**: The code is responsive to inputs and runs efficiently. if-else blocks allows complex decision-making, if any, and the elimination of redundant codes. When “if” condition is unmet, other condition/s are tested.

Python code (refer to the next line onwards)

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

products\_price = {"laptop": 300, "mouse": 30, "webcam": 40, "keyboard": 15, "speaker": 35}

query = "yes"

updated\_items = []

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

while query == "yes":

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

if item not in products:

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

else:

price\_of\_item = products\_price[item]

print(f'It is SGD{price\_of\_item}.')

updated\_items.append(f'{item} at SGD{price\_of\_item}')

query = input("Do you still want to buy products? (yes/no): ")

print(f'You have selected the following items for purchase: {updated\_items}')

**References**

Hattingh, F., Buitendag, A., & Lall, M. (2020). Systematic literature review to identify and rank the most common reasons for plagiarism. *M. Jones (Ed.), Proceedings of InSITE 2020: Informing Science and Information Technology Education Conference*, 159-182. <https://doi.org/10.28945/4576>

TBriefcase. (2019, December 19). *Plagiarism and Programming: How to Code without Plagiarizing*. <https://www.itbriefcase.net/how-to-code-without-plagiarizing>

Kadosh, Y. (2019, September 1). *Why Rewriting Is Essential for Keeping Software Alive*. Better Programming. <https://betterprogramming.pub/why-code-rewriting-is-essential-for-keeping-software-alive-81c7307e7f6>

Sharovar, E. (2022, December 17). *Code rewriting: When and Why*. Waverly Software. <https://waverleysoftware.com/blog/code-rewriting-when-and-why/#:~:text=Here%20are%20some%20good%20reasons,set%20Continuous%20Integration%20or%20Deployment>.