Logo, company name

Description automatically generated

**ANL 252**

**Python for Data Analytics**

**Tutor-Marked Assignment (TMA01)**

**July 2023 Presentation**

**Tan Wee Huat (K2270304)**

**Question 1**

1. Plagiarism in general is simply the act of taking someone else's work or idea and taking credit for that. It can be quite an arduous task to produce a code or to find a suitable solution for any given problem. Thus, it is tempting to take existing solutions that solves the problem without having to put in much effort.

* Establish clear rules on academic integrity by adopting the perspective of collaboration instead of collusion. (Lee, 2023) This allows students to interact with the use of AI tools but within boundaries.
* Curate policies that govern the use of external code. (Lee, 2023) Similarly to citing from articles, the same should be done for codes to give due credit to the original owners.
* Seek the permission of the owners if you plan to use their code. (Mercado, 2022) This avoids any misunderstanding or potential prosecution by the other party.
* Always cite information from external sources and verify using a plagiarism checker. (Mercado, 2022) This ensures that your work is original and flags out segment of information that belongs to someone else.

In summary, plagiarism can happen anywhere but with proper ethics and a framework, we can detect and avoid such instances.

**(200 Words)**

**Question 1**

𝚒𝚖𝚙𝚘𝚛𝚝 𝚛𝚊𝚗𝚍𝚘𝚖

𝚍𝚎𝚏 𝚗𝚞𝚖𝚋𝚎𝚛\_𝚐𝚞𝚎𝚜𝚜𝚒𝚗𝚐():

𝚕𝚘𝚠𝚎𝚛 = 𝟷

𝚞𝚙𝚙𝚎𝚛 = 𝟷00

𝚊𝚗𝚜𝚠𝚎𝚛 = 𝚛𝚊𝚗𝚍𝚘𝚖.𝚛𝚊𝚗𝚍𝚒𝚗𝚝(𝚕𝚘𝚠𝚎𝚛, 𝚞𝚙𝚙𝚎𝚛)

𝚙𝚛𝚒𝚗𝚝(𝚏"𝙻𝚎𝚝'𝚜 𝚙𝚕𝚊𝚢 𝚐𝚞𝚎𝚜𝚜 𝚝𝚑𝚎 𝚗𝚞𝚖𝚋𝚎𝚛 𝚐𝚊𝚖𝚎!\𝚗 𝙸 𝚊𝚖 𝚝𝚑𝚒𝚗𝚔𝚒𝚗𝚐 𝚘𝚏 𝚊 𝚗𝚞𝚖𝚋𝚎𝚛 𝚝𝚑𝚊𝚝 𝚒𝚜 𝚋𝚎𝚝𝚠𝚎𝚎𝚗 0 𝚝𝚘 𝟷00")

𝚠𝚑𝚒𝚕𝚎 𝚃𝚛𝚞𝚎:

𝚐𝚞𝚎𝚜𝚜 = 𝚒𝚗𝚝(𝚒𝚗𝚙𝚞𝚝("𝚃𝚊𝚔𝚎 𝚊 𝚐𝚞𝚎𝚜𝚜: "))

𝚒𝚏 𝚐𝚞𝚎𝚜𝚜 < 𝚊𝚗𝚜𝚠𝚎𝚛:

𝚙𝚛𝚒𝚗𝚝("𝙶𝚞𝚎𝚜𝚜 𝚝𝚘𝚘 𝚕𝚘𝚠, 𝚝𝚛𝚢 𝚊𝚐𝚊𝚒𝚗!")

𝚎𝚕𝚒𝚏 𝚐𝚞𝚎𝚜𝚜 > 𝚊𝚗𝚜𝚠𝚎𝚛:

𝚙𝚛𝚒𝚗𝚝("𝙶𝚞𝚎𝚜𝚜 𝚝𝚘 𝚑𝚒𝚐𝚑, 𝚝𝚛𝚢 𝚊𝚐𝚊𝚒𝚗!")

𝚎𝚕𝚜𝚎:

𝚙𝚛𝚒𝚗𝚝(𝚏"𝙲𝚘𝚗𝚐𝚛𝚊𝚝𝚞𝚕𝚊𝚝𝚒𝚘𝚗𝚜 𝚢𝚘𝚞 𝚖𝚊𝚗𝚊𝚐𝚎𝚍 𝚝𝚘 𝚐𝚞𝚎𝚜𝚜 𝚝𝚑𝚎 𝚛𝚒𝚐𝚑𝚝 𝚗𝚞𝚖𝚋𝚎𝚛!")

𝚋𝚛𝚎𝚊𝚔

𝚒𝚏 \_\_𝚗𝚊𝚖𝚎\_\_ == "\_\_𝚖𝚊𝚒𝚗\_\_":

𝚗𝚞𝚖𝚋𝚎𝚛\_𝚐𝚞𝚎𝚜𝚜𝚒𝚗𝚐()

The code above mimics that of a random number generator between 0 to 100 and the user must guess what the hidden number is. (Parashar, 2022) As per the requirements given, the code must be between 15 to 30 lines and each line should serve a purpose. For this code, we must import the random library to be able to generate a randomized integer. A function is defined and assigned a set of instructions so when called upon, it can execute our instructions. This code first generates a random number between our defined parameters of 0 to 100. That number is then assigned to an integer variable to store it. Afterwards, it prompts the user to key in their guess. Depending on how high or low the guess is, it will prompt back the user hints to guide them towards the right answer. Since there are unlimited attempts, it allows the user to eventually get the answer with these hints. It is a relatively simple code that can be properly executed if the parameters are properly defined. I used if-elif-else condition block since there are only 3 possible outcomes: the guessed number is too high, too low or guessed correctly.

**(200 Words)**

**Question 1**

𝚒𝚖𝚙𝚘𝚛𝚝 𝚛𝚊𝚗𝚍𝚘𝚖

𝚍𝚎𝚏 𝚖𝚊𝚒𝚗():

𝚙𝚛𝚒𝚗𝚝("𝚆𝚎𝚕𝚌𝚘𝚖𝚎 𝚝𝚘 𝙶𝚞𝚎𝚜𝚜 𝚝𝚑𝚎 𝙽𝚞𝚖𝚋𝚎𝚛!")

𝚙𝚕𝚊𝚢\_𝚐𝚊𝚖𝚎()

𝚍𝚎𝚏 𝚙𝚕𝚊𝚢\_𝚐𝚊𝚖𝚎():

𝚕𝚘𝚠𝚎𝚛 = 𝟷

𝚞𝚙𝚙𝚎𝚛 = 𝟷00

𝚊𝚗𝚜𝚠𝚎𝚛 = 𝚛𝚊𝚗𝚍𝚘𝚖.𝚛𝚊𝚗𝚍𝚒𝚗𝚝(𝚕𝚘𝚠𝚎𝚛, 𝚞𝚙𝚙𝚎𝚛)

𝚊𝚝𝚝𝚎𝚖𝚙𝚝 = 0

𝚙𝚛𝚒𝚗𝚝(𝚏"𝙸'𝚖 𝚝𝚑𝚒𝚗𝚔𝚒𝚗𝚐 𝚘𝚏 𝚊 𝚗𝚞𝚖𝚋𝚎𝚛 𝚋𝚎𝚝𝚠𝚎𝚎𝚗 {𝚕𝚘𝚠𝚎𝚛} 𝚊𝚗𝚍 {𝚞𝚙𝚙𝚎𝚛}.")

𝚠𝚑𝚒𝚕𝚎 𝚃𝚛𝚞𝚎:

𝚐𝚞𝚎𝚜𝚜 = 𝚒𝚗𝚝(𝚒𝚗𝚙𝚞𝚝("𝙿𝚕𝚎𝚊𝚜𝚎 𝚒𝚗𝚙𝚞𝚝 𝚢𝚘𝚞𝚛 𝚐𝚞𝚎𝚜𝚜: "))

𝚊𝚝𝚝𝚎𝚖𝚙𝚝 += 𝟷

𝚒𝚏 𝚐𝚞𝚎𝚜𝚜 < 𝚊𝚗𝚜𝚠𝚎𝚛:

𝚙𝚛𝚒𝚗𝚝("𝚃𝚘𝚘 𝚕𝚘𝚠! 𝚃𝚛𝚢 𝚊𝚐𝚊𝚒𝚗.")

𝚎𝚕𝚒𝚏 𝚐𝚞𝚎𝚜𝚜 > 𝚊𝚗𝚜𝚠𝚎𝚛:

𝚙𝚛𝚒𝚗𝚝("𝚃𝚘𝚘 𝚑𝚒𝚐𝚑! 𝚃𝚛𝚢 𝚊𝚐𝚊𝚒𝚗.")

𝚎𝚕𝚜𝚎:

𝚙𝚛𝚒𝚗𝚝(𝚏"𝙲𝚘𝚗𝚐𝚛𝚊𝚝𝚞𝚕𝚊𝚝𝚒𝚘𝚗𝚜! 𝚈𝚘𝚞 𝚐𝚞𝚎𝚜𝚜𝚎𝚍 𝚝𝚑𝚎 𝚗𝚞𝚖𝚋𝚎𝚛 {𝚊𝚗𝚜𝚠𝚎𝚛} 𝚒𝚗 {𝚊𝚝𝚝𝚎𝚖𝚙𝚝} 𝚊𝚝𝚝𝚎𝚖𝚙𝚝𝚜.")

𝚋𝚛𝚎𝚊𝚔

𝚒𝚏 \_\_𝚗𝚊𝚖𝚎\_\_ == "\_\_𝚖𝚊𝚒𝚗\_\_":

𝚖𝚊𝚒𝚗()

The rationale of rewriting a code is analogous to drafting a continuous story; rewriting it allows one to “tidy up” their code by getting rid of unnecessary details or introducing new functions to create a more cohesive program. To enhance and introduce new functions, one must rewrite their code for proper integration. (Sharovar, 2022)

* Combination of pre-defined strings with values of expressions gives flexibility for updating the changes in values automatically. Previously, I manually input a string to print out the number range but now I replaced it with the expression value which gives the flexibility to update the string automatically when I change the limits.
* Introduced a new function to call upon running the program instead of the prompting the user with a blank output and having to call the right function to run the program. Previously, I had to input the function name to run the program but now I introduced an if condition that automatically calls the function whenever I run the program.
* Add a new variable to track the number of attempts the user took to guess the right number and automatically display that information inside a string. In this new program, a new variable “attempt” was declared and assigned with a 0 integer. Inside the while loop, for every attempt there would be an increment of +1 which will be stored inside “attempt”. Upon guessing the correct number, the program output’s a string with an expression value of “attempt” to display how many attempts the user took to guess the correct number.

Rewriting the program was to introduce new function to enable the program to be more autonomous and designing it to be more user-friendly. With the guidance of online program websites and the study materials provided, it was relatively straightforward to build the program.

**(300 Words)**

**Question 2**

Based on Appendix 1, first enhancement would be to create a loop to constantly prompt the user until a valid input has been received. I changed its while condition to true which allows it to loop indefinitely and replaced break with continue to allow the program to continue into the next iteration. This improves overall reliability as it allows the user to attempt until successful.

|  |
| --- |
| 𝚠𝚑𝚒𝚕𝚎 𝚃𝚛𝚞𝚎:  𝚒𝚝𝚎𝚖 = 𝚒𝚗𝚙𝚞𝚝("𝙷𝚎𝚕𝚕𝚘! 𝚆𝚑𝚊𝚝 𝚍𝚘 𝚢𝚘𝚞 𝚠𝚊𝚗𝚝 𝚝𝚘 𝚋𝚞𝚢?")  𝚒𝚏 𝚒𝚝𝚎𝚖 𝚗𝚘𝚝 𝚒𝚗 𝚙𝚛𝚘𝚍𝚞𝚌𝚝𝚜:  𝚙𝚛𝚒𝚗𝚝(𝚏'𝚆𝚛𝚘𝚗𝚐 𝚙𝚛𝚘𝚍𝚞𝚌𝚝! 𝙿𝚕𝚎𝚊𝚜𝚎 𝚝𝚛𝚢 𝚊𝚐𝚊𝚒𝚗.')  𝚌𝚘𝚗𝚝𝚒𝚗𝚞𝚎 |

The second enhancement have a more structured data storage to store the product name and its pricing. Originally it was stored in a list without including the price information. Instead, I introduced a dictionary to store the list of products as well as their pricing which makes it easier to access the data. This improves maintainability as the data is stored in a dictionary with multiple information.

|  |
| --- |
| 𝚙𝚛𝚘𝚍𝚞𝚌𝚝𝚜 = {  '𝚕𝚊𝚙𝚝𝚘𝚙': 𝟷000,  '𝚖𝚘𝚞𝚜𝚎': 𝟸0,  '𝚠𝚎𝚋𝚌𝚊𝚖': 𝟻0,  '𝚔𝚎𝚢𝚋𝚘𝚊𝚛𝚍': 𝟹0,  '𝚜𝚙𝚎𝚊𝚔𝚎𝚛': 𝟾0  } |

The third enhancement would be to convert the user input into a consistent string so that a valid input is recognised regardless of lower- or upper-case letters. (Team, 2020) The updated code can convert the user input into a string on lower-case letters and is able to recognise that input by looking into the product dictionary. This improves the readability of the program.

|  |
| --- |
| 𝚠𝚑𝚒𝚕𝚎 𝚃𝚛𝚞𝚎:  𝚒𝚝𝚎𝚖 = 𝚒𝚗𝚙𝚞𝚝("𝙷𝚎𝚕𝚕𝚘! 𝚆𝚑𝚊𝚝 𝚍𝚘 𝚢𝚘𝚞 𝚠𝚊𝚗𝚝 𝚝𝚘 𝚋𝚞𝚢?")  𝚕𝚘𝚠𝚎𝚛\_𝚌𝚊𝚜𝚎\_𝚒𝚝𝚎𝚖 = 𝚒𝚝𝚎𝚖.𝚕𝚘𝚠𝚎𝚛()    𝚒𝚏 𝚕𝚘𝚠𝚎𝚛\_𝚌𝚊𝚜𝚎\_𝚒𝚝𝚎𝚖 𝚗𝚘𝚝 𝚒𝚗 𝚙𝚛𝚘𝚍𝚞𝚌𝚝𝚜:  𝚙𝚛𝚒𝚗𝚝(𝚏'𝚆𝚛𝚘𝚗𝚐 𝚙𝚛𝚘𝚍𝚞𝚌𝚝! 𝙿𝚕𝚎𝚊𝚜𝚎 𝚝𝚛𝚢 𝚊𝚐𝚊𝚒𝚗.')  𝚌𝚘𝚗𝚝𝚒𝚗𝚞𝚎 |

**(195 Words)**

# References

Lee, C. (2023, August 22). *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>

Mercado, K. (2022, July 8). *Does Plagiarism Issue Apply To Programming?*. Kode Java. <https://kodejava.org/does-plagiarism-issue-apply-to-programming/>

Parashar, A. (2022, January 5). *10 python mini projects that everyone should build with code*. Medium. <https://towardsdatascience.com/10-python-mini-projects-that-everyone-should-build-with-code-769c6f1af0c4>

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

Team, D. (2020, June 25). *Lowercase in python tutorial*. <https://www.datacamp.com/tutorial/case-conversion-python>