**Logo

Description automatically generated**

Course code : **ANL252**

Course title : **Python for Data Analytics**

Assessment title : **Tutor-Marked Assignment 01 (TMA01)**

**July 2023 Presentation**

Student name **: Lee Jay Hoon Jenny**

Student PI number : **Y2070655**

Submission date **: 15 September 2023**

**Question 1 (a)**

Plagiarism in coding happens in academic environments is a serious issue and we shall discuss reasons for one to use existing coding without mentioning the source and how to avoid such actions:

Why plagiarism happens:

* **Availability & accessibility of online resources:** Open-source platforms and online resources allow the easy sharing of codes and solutions. This accessibility can tempt individuals into copying and pasting someone else's codes into their own projects.
* **Lack of programming knowledge:** When facing difficulties in coding a particular programming concept, one may resort to simply copying code from the internet without understanding how it works.
* **Inadequate knowledge of plagiarism policies:** Individuals may be unaware of the obligations outlined in the various software licenses, and hence fail to comply with them (e.g. not crediting the original author).

Preventing plagiarism:

* **Understanding the source material:** Instead of copying and pasting the source material, individuals should seek to comprehend them, and then implement the code in their own way.
* **Using a plagiarism checker:** There are solutions available to detect coding plagiarism and identify any unintentional similarities between their work and existing code.
* **Understanding the software licensing terms:** Reading up on a source material's software licensing terms can help an individual take steps to avoid violating them.

* **Coding originally:** Writing original variable names alongside code commenting can demonstrate an in-depth comprehension of the code that was written and distinguish original work.

[196 words]

**Question 1 (b)**

The following piece of code is a simple program that outputs the factorial of a non-negative integer provided by the user.

|  |
| --- |
| def factorial(n):     if n == 0:         return 1     else:         return n \* factorial(n-1)  valid = False while valid == False:     user\_input=input("Please provide a non-negative integer: ")     if(user\_input.isnumeric()):         valid=True     else:         print("Please enter a valid non-negative integer")         valid=False result = factorial(int(user\_input)) print(f"The factorial of {user\_input} is {result}") |

[16 lines]

The “factorial” function takes in a parameter “n” , which represents the non-negative integer for which the factorial is to be calculated. The internal working of the function is to first check if the value of “n” is 0 (base case), in which case it will return the value 1. Otherwise, it will recursively multiply the value of “n” by the factorial of “n-1”.

The main program uses a while loop to repeatedly prompt the user for a valid non-negative integer value. The validation of the user input is achieved via python-in-built “isnumeric” function (n.d.,n.d.) which checks if the string provided by the user contains all numeric characters.

The “isnumeric” function also prevents users from inputting negative and floating values. Specifically, it does not accept “-” and “.” characters. Once a valid input is obtained, it calculates the factorial of the input using the “factorial” function and prints the result.

[176 words]

**Question 1 (c)**

The rationale for rewriting:

1. **Better code readability**. The program has been rewritten with a much more descriptive function and variable name. In addition, comments have been added to understand the program better.
2. **Better code maintainability.** The new code encapsulates the main program, factorial, and the retrieval of user input into its own dedicated function, hence achieving separation of concern. This improves code maintainability as bug fixes or changes in individual sections can be made without breaking other parts of the code.
3. **Added error handling**. A try-except block has been added in the main function to handle any unexpected errors gracefully.
4. **Code Optimisation**. The base case of the factorial function has been optimised to just check if the variable “number” is less than or equal to 1, in which case it will return the value 1. In the original code, the base case checks if the variable “number” is equal to 0 before returning 1. This results in an additional recursion loop. Hence, this new base case allows the code to be more efficient and optimised.

[175 words]

|  |
| --- |
| *# Function to get the factorial of the number provided* def factorial(number):     *# Stop the recursion if value reaches <=1 , as the function would return 1 once value drop below 1*     if number <= 1:         return 1     else:         return number \* factorial(number - 1)   *# Function to get user's input* def getUserInput():     valid = False  *# Determining factor to stop while loop*     while valid == False:         *# Get User Input*         user\_input = input("Please provide a non-negative integer: ")         *# Validation*         valid = user\_input.isnumeric()     *# Return the value in integer format*     return int(user\_input)   *# Main function* def main():     try:         user\_input = getUserInput()         result = factorial(user\_input)         print(f"The factorial of {user\_input} is {result}")     except Exception as error:         print("An exception occurred:", error) main() |

**Question 2**

**Overall code quality**

* For line 4, instead of simply printing the raw output of the product list, iterate through it and print-format it nicely.
* Change the original ‘product’ list to a dictionary, which is a better data type for representing the user’s input and uses one less variable, ‘updated\_items’. Each item will be a dict key, and the dict value is an empty list that will represent the inputted price.

**Reliability**

* Lines 8 to 10 prompt the user to try again if an invalid product is entered. However, the 'break' statement on line 10 causes the program to prematurely exit. To fix this, change 'break' to 'continue'.
* The query on line 14 can be case-insensitive for a better user experience.
* Each of the program’s outputs, including the user prompts, can be formatted to look prettier. For example, line 15.
* Validation should be performed on the price as well.

**Readability**

* Comments should be added for better code-readability.
* The program should use meaningful variable names and function names.

**Maintainability**

* Parts of the program can be broken down into reusable functions. (e.g. a function to validate item prices is meaningful and highly reusable).

[199 words]

|  |
| --- |
| *# validate the item price input by the user.* def is\_price\_valid(item\_price):     if not item\_price.isnumeric() or int(item\_price)<0:         print(f'[-]Invalid value entered! Please try again.')         return False     return True  def main():     product\_dict = {'laptop':[],'mouse':[],'webcam':[],'keyboard':[],'speaker':[]}     query = 'yes'          print(f'Our list of products:')     for product in product\_dict:         print(f'[\*]{product}')              while query == 'yes':         *#prompt user for item and perform validation*         item = str(input("Hello! What do you want to buy?\r\n[?] Enter item name:")).lower()         if item not in product\_dict:             print(f'[-] Wrong product! Please try again.')             continue                      *#prompt user for price and perform validation*         while True:             item\_price = input("How much is it (in SGD)\r\n[?] Enter price:")             if not is\_price\_valid(item\_price):                 continue                   *# break out of the loop on validation success*             break                      product\_dict[item].append(item\_price)         query = str(input("[\*] Would you like to continue?\r\n(yes/no): ")).lower()                  *# print out shopping list*     print(f'This is our updated shopping list:')     for item, shopping\_list in product\_dict.items():         for price in shopping\_list:             print(f'\* {item} -> {str(price)}')  main() |

[40 lines]

**Reference**

Wu, K. Y., & Zhu, S. (2023). *ANL252 Python for data analytics*. Singapore University of

Social Sciences.

Release V1.2.0

Build S1.0.15, T1.7.2

n.d. (n.d.). *Python String isnumeric() method*. Retrieved from w3schools: https://www.w3schools.com/python/ref\_string\_isnumeric.asp