

**ANL252**

**Python for Data Analytics**

**Tutor-Marked Assignment (TMA01)**

**July 2023 Presentation**

NAME: NUR AMEERAH BINTE NASSIRUDIN

SUSS PI NO: Y2170573

SUBMISSION DATE: 10 SEPTEMBER 2023

**Question 1a)**

Plagiarism in coding occurs when a programmer copies or closely imitates someone else's code without proper attribution or consent.

Factors as to why plagiarism in coding happens are due to:

1. Having a lack of understanding of coding concepts and resorts to copying code to complete assignments or projects.
2. Pressures to meet deadlines result in programmers taking shortcuts and plagiarizing code to save time. This is common in fast-paced environments where time constraints are a constant challenge.
3. Lack of knowledge of the ethical and professional standards related to code plagiarism, especially if they have not received proper education or guidance.

Code plagiarism can be avoided by:

1. Thoroughly understanding the code that the individual is working on. Help can be sought out via independent research or consulting mentors or teachers.
2. Ensuring that proper attribution is included by citing the original author or source. Any licensing agreements or requirements associated with the code need to be followed as well.
3. Consulting documentation such as textbooks, and online resources for guidance and reference.
4. Collaborating with peers and engaging in code reviews will help to identify potential issues and ensure that no one is inadvertently plagiarizing code.

**Question 1b)**

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

# Input

num = 5

# Calculate factorial

result = factorial(num)

# Output

print(f"The factorial of {num} is {result}")

The factorial of 5 is 3628800

This code computes and prints the factorial of the input number. The factorial of a non-negative integer n is the product of all positive integers from 1 to n. In this example, 10! is calculated as 10 x 9 x 8 x 7 x 6 x 5 x 4 x 3 x 2 x 1, which equals 3,628,800.

**Question 1c)**

# Recursive function to calculate factorial

def factorial(n):

return 1 if n == 0 else n \* factorial(n - 1)

# Input

num = 10

# Calculate factorial

result = factorial(num)

# Output

print(f"Factorial of {num}: {result}")

Factorial of 10: 3628800

The initial code in 1b uses an ‘if-else’ statement for the factorial equation which was then simplified to a more concise ternary expression of (‘1 if n == 0 else n \* factorial (n - 1)’) to reduce code complexity.

**Question 2)**

Functions of modularity can be used to break down the code into functions to improve maintainability and readability. Create functions for adding items to the shopping list, checking if the entered item is valid, and handling user input for continuing shopping.

Error handling can be implemented to handle exceptions that might occur during user input, such as invalid price inputs. By using a try-except block to catch and handle exceptions gracefully, providing informative error messages to the user.

Using meaningful variable names conveys the purpose of each variable more clearly.

Hence, the improved code is

def add\_item\_to\_list(products, updated\_items):

while True:

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

if item not in products:

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

continue

while True:

try:

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

break

except ValueError:

print('Invalid input for price. Please enter a valid number.')

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

updated\_items.append(entered\_input)

continue\_shopping = input("Would you like to continue shopping? (yes/no) ").lower()

if continue\_shopping != 'yes':

break

def main():

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

updated\_items = []

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

add\_item\_to\_list(products, updated\_items)

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

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

main()

**References:**

1. Cosma, G. [Georgina], Joy, M. [Mike], May 2008, Plagiarism Online, UK Academics

<http://web.science.mq.edu.au/~mtaylor/ponline/index.php?id=source-code-plagiarism#:~:text=Source%2Dcode%20plagiarism%20in%20programming,as%20his%2Fher%20own%20work>.

1. Maryono, D., Yuana, R. A., Hatta, P., 2019, 2018 International Conference of Computer and Informatics Engineering (IC2IE), IOP Publishing,

<https://iopscience.iop.org/article/10.1088/1742-6596/1193/1/012027/pdf#:~:text=Typical%20examples%20of%20plagiarism%20which,merging%20declaration%20variable%20(L4)>.

1. Rai, A. 7 March 2023, 7 Steps to Improve Code Quality, Stackify

<https://stackify.com/7-steps-to-improve-code-quality/>

1. Bellairs, R., 11 July 2019, What Is Code Quality? Overview + How to Improve Code Quality, Perforce

<https://www.perforce.com/blog/sca/what-code-quality-overview-how-improve-code-quality>

1. Vishwakarma, S., 15 May 2023, Improve Code Quality with These Tips and Best Practices, Dev

<https://dev.to/documatic/improve-code-quality-with-these-tips-and-best-practices-2mh2>

1. Mashurihammad, n. d., Modularity and Its Properties, Geeks for Geeks

<https://www.geeksforgeeks.org/modularity-and-its-properties/>

1. Google Developers, n.d., General Error Handling Rules, Google

<https://developers.google.com/tech-writing/error-messages/error-handling>

1. Rouse, M., 1 May 2017, Error Handling, Techopedia

<https://www.techopedia.com/definition/16626/error-handling>