A logo with blue and red letters

Description automatically generated

A white paper with blue text

Description automatically generated

**Course Code: ANL252 Python for Data Analytics**

**TMAO1**

**Name: Ris Nurish Iszryfinna**

**PI Number: Y2270086**

**Submission Date: 15th September 2023**

Contents

[**1a)** 4](#_Toc145629249)

[**1b)** 4](#_Toc145629250)

[**1c)** 5](#_Toc145629251)

[**2)** 7](#_Toc145629252)

# **1a)**

Plagiarism in coding occurs due to various reasons, but it can be avoided through several key practices:

1. Lack of Understanding: Some developers copy code because they don't fully understand the concepts or algorithms involved.

2. Time Pressure: Tight deadlines may lead to copying code to save time, instead of writing it from scratch.

3. Inadequate Training: Developers who haven't been educated about plagiarism and its consequences may inadvertently engage in it.

4. Cut-and-Paste Mentality: A quick internet search can provide ready-made solutions, tempting developers to copy without comprehension.

5. Competition: Fierce competition in the tech industry may push individuals to take shortcuts, including plagiarism.

To avoid plagiarism:

1. Understand the Code: Comprehend the code you're using to ensure you can explain and modify it if needed.

2. Use Proper Attribution: Give credit to the original source through comments or proper citations.

3. Write from scratch: Whenever possible, write your code independently, even if it takes more time.

4. Educate and Train: Promote awareness of plagiarism among team members and provide training on ethical coding practices.

5. Use Plagiarism Detection Tools: Employ code similarity detection tools to identify potential instances of plagiarism.

By understanding why plagiarism occurs and implementing these preventive measures, developers can maintain ethical coding practices.

# **1b)**

Python Code:

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

try:

num = int(input("Enter a positive integer: "))

if num < 0:

print("Please enter a positive integer.")

else:

result = factorial(num)

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

except ValueError:

print("Invalid input. Please enter a positive integer.")

Explanation:

This Python code calculates the factorial of a positive integer entered by the user using a recursive function. It prompts the user to enter a positive integer, checks if the input is valid, and then calculates the factorial. The `factorial` function recursively multiplies the number by the factorial of the number minus one until it reaches the base case (0 factorial is defined as 1). It then prints the result.

**Example Output:**

Enter a positive integer: 5

The factorial of 5 is 120.

This code illustrates the use of recursion to solve a mathematical problem and demonstrates user input handling.

# **1c)**

def factorial(n):

result = 1

while n > 0:

result \*= n

n -= 1

return result

try:

num = int(input("Enter a positive integer: "))

if num < 0:

print("Please enter a positive integer.")

else:

result = factorial(num)

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

except ValueError:

print("Invalid input. Please enter a positive integer.")

**Rationale for Rewriting:**

**1.Recursion to Iteration:** The original code used recursion to calculate the factorial. Rewriting it using a while loop provides an alternative approach and helps learners understand both methods.

**2. Enhanced Readability:** Using a while loop simplifies the code and makes it more accessible for individuals who may not be familiar with recursive functions.

**3.Improved Efficiency:** The iterative solution is generally more efficient for calculating factorials as it avoids the function call overhead associated with recursion.

**Example Output:**

Enter a positive integer: 5

The factorial of 5 is 120.

This code maintains the same functionality as the original but presents an alternative way to solve the problem while improving clarity and efficiency.

# **2)**

To improve the code in Appendix 1, consider the following enhancements for reliability, readability, maintainability, and overall quality:

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

updated\_items = []

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

while True:

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

if item.lower() not in products:

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

else:

price\_of\_item = input(f"How much is {item} (in SGD)? ").strip()

try:

price\_of\_item = float(price\_of\_item)

updated\_items.append([item, price\_of\_item])

except ValueError:

print("Invalid price. Please enter a valid number.")

query = input("Would you like to continue? (yes/no) ").strip().lower()

if query != 'yes':

break

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

for item, price in updated\_items:

print(f'{item}: SGD {price:.2f}')

**Enhancements:**

**1. Input Validation:** The code now validates the price input, ensuring it's a valid number.

**2. User-Friendly**: Added clearer prompts for input and error messages to enhance user experience.

**3. Consistency:** Ensured consistent formatting, indentation, and use of lowercase for comparisons, improving code readability and reliability.