

# ASSIGNMENT-10.5

## Lab 10 – Code Review and Quality: Using AI to Improve Code

### Quality and Readability

NAME: M.NISHTHA

HT NO:(2303A51540)

BTNO: 19

#### Task 1: Improving Variable & Function Names

Original Code

```
def f(a, b):  
    return a + b  
  
print(f(10, 20))
```

**CODE:**

```
def add_numbers(first_number, second_number):
```

"""

Returns the sum of two numbers.

"""

```
    return first_number + second_number
```

```
result = add_numbers(10, 20)
```

```
print(result)
```

```
Users > nishithamarepally > Desktop > 10.5_1py > ...
```

```
1 def add_numbers(first_number, second_number):  
2     """  
3     Returns the sum of two numbers.  
4     """  
5     return first_number + second_number  
6  
7  
8 result = add_numbers(10, 20)  
9 print(result)
```

## OUTPUT

```
● nishithamarepally@nishithas-MacBook-Air ~ % /usr/bin/python3 /Users/nishithamarepally/Desktop/10.5_1py  
30
```

### Task Description #2 – Missing Error Handling

Task: Use AI to add proper error handling.

Sample Input Code:

```
def divide(a, b):  
  
    return a / b  
  
print(divide(10, 0))
```

Expected Output:

- Code with exception handling and clear error messages

## CODE :

```
def divide_numbers(dividend, divisor):  
    """
```

Divides two numbers safely.

.....

try:

```
    return dividend / divisor
```

except ZeroDivisionError:

```
    return "Error: Division by zero is not allowed."
```

except TypeError:

```
    return "Error: Please enter valid numbers."
```

```
result = divide_numbers(10, 0)
```

```
print(result)
```

```
Users > nishithamarepally > Desktop > 10.5_2.py > ...
1  def divide_numbers(dividend, divisor):
2      """
3          Divides two numbers safely.
4      """
5      try:
6          return dividend / divisor
7      except ZeroDivisionError:
8          return "Division by zero is not allowed."
9      except TypeError:
10         return "Please enter valid numbers."
11
12
13 result = divide_numbers(10, 0)
14 print(result)
```

## OUTPUT:

```
● nishithamarepally@nishithas-MacBook-Air ~ % /usr/bin/python3 /Users/nishithamarepally/Desktop/10.5_2.py
Division by zero is not allowed.
○ nishithamarepally@nishithas-MacBook-Air ~ %
```

### **Task Description #3: Student Marks Processing System**

The following program calculates total, average, and grade of a student, but it has poor readability, style issues, and no error handling.

```
marks=[78,85,90,66,88]
```

```
t=0
```

```
for i in marks:
```

```
    t=t+i
```

```
a=t/len(marks)
```

```
if a>=90:
```

```
    print("A")
```

```
elif a>=75:
```

```
    print("B")
```

```
elif a>=60:
```

```
    print("C")
```

```
else:
```

```
    print("F")
```

Task:

- Use AI to refactor the code to follow PEP 8 standards.
- Add meaningful variable names, functions, and comments.
- Add basic input validation and documentation.

**CODE :**

```
def calculate_average(marks_list):
```

```
    """
```

Calculates the average of student marks.

Args:

marks\_list (list): List of student marks

Returns:

float: Average marks

"""

if not marks\_list:

    raise ValueError("Marks list cannot be empty.")

return sum(marks\_list) / len(marks\_list)

def determine\_grade(average\_marks):

"""

Determines grade based on average marks.

"""

if average\_marks >= 90:

    return "A"

elif average\_marks >= 75:

    return "B"

elif average\_marks >= 60:

    return "C"

else:

    return "F"

def main():

"""

Main function to process student marks.

\*\*\*\*

```
marks = [78, 85, 90, 66, 88]
```

```
try:
```

```
    average = calculate_average(marks)
```

```
    grade = determine_grade(average)
```

```
    print(f"Average Marks: {average:.2f}")
```

```
    print(f"Grade: {grade}")
```

```
except ValueError as error:
```

```
    print(f"Error: {error}")
```

```
if __name__ == "__main__":
```

```
    main()
```

Users > nishithamarepally > Desktop > 10.5\_3.py > main

```
1  def calculate_average(marks_list):
2      """
3          Calculates the average of student marks.
4
5          Args:
6              marks_list (list): List of student marks
7
8          Returns:
9              float: Average marks
10             """
11         if not marks_list:
12             raise ValueError("Marks list cannot be empty.")
13
14         return sum(marks_list) / len(marks_list)
15
16
17     def determine_grade(average_marks):
18         """
19             Determines grade based on average marks.
20             """
21         if average_marks >= 90:
22             return "A"
23         elif average_marks >= 75:
24             return "B"
25         elif average_marks >= 60:
26             return "C"
27         else:
28             return "F"
```

```
31 def main():
32     """
33     Main function to process student marks.
34     """
35     marks = [78, 85, 90, 66, 88]
36
37     try:
38         average = calculate_average(marks)
39         grade = determine_grade(average)
40
41         print(f"Average Marks: {average:.2f}")
42         print(f"Grade: {grade}")
43
44     except ValueError as error:
45         print(f"Error: {error}")
46
47
48 if __name__ == "__main__":
49     main()
```

## OUTPUT:

```
nishithamarepally@nishithas-MacBook-Air ~ % /usr/bin/python3 /Users/nishithamarepally/Desktop/1
0.5_3.py
Average Marks: 81.40
Grade: B
nishithamarepally@nishithas-MacBook-Air ~ % /usr/bin/python3 /Users/nishithamarepally/Desktop/1
```

## Task Description #4: Use AI to add docstrings and inline comments

to the following function.

```
def factorial(n):
```

```
    result = 1
```

```
    for i in range(1,n+1):
```

```
        result *= i
```

```
    return result
```

## CODE :

```
def factorial(number):
```

```
"""
```

```
Calculates the factorial of a given number.
```

```
"""
```

```
if number < 0:
```

```
    raise ValueError("Negative numbers not allowed.")
```

```
result = 1
```

```
for i in range(1, number + 1):
```

```
    result *= i
```

```
return result
```

```
if __name__ == "__main__":
```

```
    number = int(input("Enter a number: "))
```

```
    print("Factorial:", factorial(number))
```

```
Users > nishithamarepally > Desktop > 10.5_4.py > factorial
```

```
1 def factorial(number):
2     """
3     Calculates the factorial of a given number.
4     """
5     if number < 0:
6         raise ValueError("Negative numbers not allowed.")
7
8     result = 1
9
10    for i in range(1, number + 1):
11        result *= i
12
13    return result
14
15
16 if __name__ == "__main__":
17     number = int(input("Enter a number: "))
18     print("Factorial:", factorial(number))
```

## OUTPUT:

```
● nishithamarepally@nishithas-MacBook-Air ~ % /usr/bin/python3 /Users/nishithamarepally/Desktop/10.5_4.py
Enter a number: 7
Factorial: 5040
✗ nishithamarepally@nishithas-MacBook-Air ~ % 5
zsh: command not found: 5
● nishithamarepally@nishithas-MacBook-Air ~ % /usr/bin/python3 /Users/nishithamarepally/Desktop/10.5_4.py
Enter a number: 5
Factorial: 120
```

## Task Description #5: Password Validation System (Enhanced)

The following Python program validates a password using only a minimum length check, which is insufficient for real-world security requirements.

```
pwd = input("Enter password: ")

if len(pwd) >= 8:
```

```
print("Strong")
```

```
else:
```

```
print("Weak")
```

## CODE:

```
import re
```

```
def is_strong_password(password):
```

```
    """
```

```
    Checks whether the given password meets security standards.
```

Rules:

- Minimum 8 characters
- At least one uppercase letter
- At least one lowercase letter
- At least one digit
- At least one special character

Args:

password (str): User-entered password

Returns:

bool: True if strong, False otherwise

```
    """
```

```
if len(password) < 8:
```

```
    return False
```

```
if not re.search(r"[A-Z]", password):
    return False

if not re.search(r"[a-z]", password):
    return False

if not re.search(r"[0-9]", password):
    return False

if not re.search(r"[@#$%^&*()_+{}[\];<>,.?~\|-]", password):
    return False

return True
```

```
def main():
    """
    Main function for password validation.
    """

    user_password = input("Enter password: ")

    if is_strong_password(user_password):
        print("Password is Strong ")
    else:
        print("Password is Weak ")
        print("It must contain:")
        print("- At least 8 characters")
        print("- One uppercase letter")
```

```

print("- One lowercase letter")
print("- One digit")
print("- One special character")

if __name__ == "__main__":
    main()

```

Users > nishithamarepally > Desktop > 10.5\_5.py > main

```

1  import re
2
3
4  def is_strong_password(password):
5      """
6          Checks whether the given password meets security standards.
7
8          Rules:
9          - Minimum 8 characters
10         - At least one uppercase letter
11         - At least one lowercase letter
12         - At least one digit
13         - At least one special character
14
15        Args:
16            password (str): User-entered password
17
18        Returns:
19            bool: True if strong, False otherwise
20        """
21
22        if len(password) < 8:
23            return False
24
25        if not re.search(r"[A-Z]", password):
26            return False
27
28        if not re.search(r"[a-z]", password):
29            return False
30

```

```

31     if not re.search(r"[0-9]", password):
32         return False
33
34     if not re.search(r"[@#$%^&*()_+{}[\]:;<,.?~\\-]", password):
35         return False
36
37     return True
38
39
40 def main():
41     """
42     Main function for password validation.
43     """
44     user_password = input("Enter password: ")
45
46     if is_strong_password(user_password):
47         print("Password is Strong ✓")
48     else:
49         print("Password is Weak ✗")
50         print("It must contain:")
51         print("- At least 8 characters")
52         print("- One uppercase letter")
53         print("- One lowercase letter")
54         print("- One digit")
55         print("- One special character")
56
57
58 if __name__ == "__main__":
59     main()

```

## OUTPUT

```

● nishithamarepally@nishithas-MacBook-Air ~ % /usr/bin/python3 /Users/nishithamarepally/Desktop/1
0.5_5.py
Enter password: Nn@2005
Password is Weak ✗
It must contain:
- At least 8 characters
- One uppercase letter
- One lowercase letter
- One digit
- One special character
⊗ nishithamarepally@nishithas-MacBook-Air ~ % Nishi@2005
zsh: command not found: Nishi@2005
● nishithamarepally@nishithas-MacBook-Air ~ % /usr/bin/python3 /Users/nishithamarepally/Desktop/1
0.5_5.py
Enter password: Nishi@2005
Password is Strong ✓
● nishithamarepally@nishithas-MacBook-Air ~ % /usr/bin/python3 /Users/nishithamarepally/Desktop/1

```