

AI ASSISTED CODING

Assignment – 10.5

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

Quality and Readability

Hall ticket No:2303A51734

Batch-11


Task Description #1 – Variable Naming Issues

Task: Use AI to improve unclear variable names.

Code:

```
C: > Users > RANJITH > OneDrive > Desktop > AI Assistant coding > 📁 vote eligible.py > ...
1  #optimize the below code that defines a function to add two numbers and prints the result
2  #add the readable comments to the code
3  # This function takes two numbers as input and returns their sum
4  def add_numbers(num1, num2):
5      # Calculate the sum of the two numbers
6      result = num1 + num2
7      # Return the result
8      return result
9
10 # Example usage of the add_numbers function
11 number1 = 5 # First number to be added
12 number2 = 10 # Second number to be added
13 # Call the function and store the result
14 sum_result = add_numbers(number1, number2)
15 # Print the result to the console
16 print(f"The sum of {number1} and {number2} is: {sum_result}")
17
```

Output:



The screenshot shows a Windows command prompt window with the following content:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Open folder in new window (ctrl + click)

PS C:\Users\RANDOMJITH> & C:/Users/RANDOMJITH/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/RANDOMJITH/OneDrive/Desktop/AI Assistant coding/vote_eligible.py"
The sum of 5 and 10 is: 15
PS C:\Users\RANDOMJITH>
```

Task-2:

Task Description #2 – Missing Error Handling

Task: Use AI to add proper error handling.

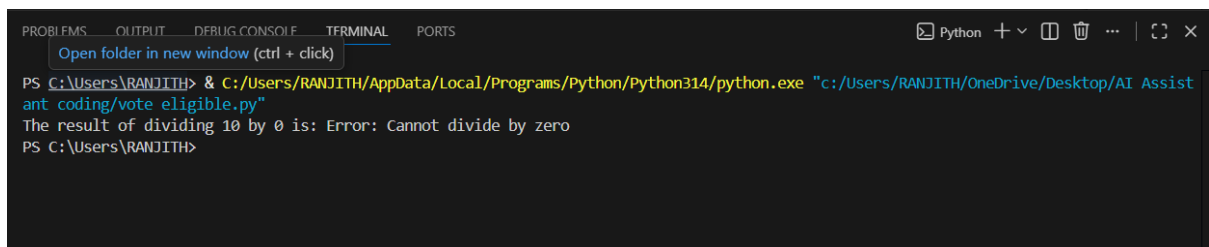
Code:

```

3
4 #task-2
5 #the below code defines a function to divide two numbers and prints the result. However, it does not handle t
6 # This function takes two numbers as input and returns their division result
7 def divide_numbers(num1, num2):
8     # Check if the second number is zero to avoid division by zero error
9     if num2 == 0:
10         return "Error: Cannot divide by zero"
11     # Calculate the division of the two numbers
12     result = num1 / num2
13     # Return the result
14     return result
15
16 # Example usage of the divide_numbers function
17 number1 = 10 # Numerator
18 number2 = 0 # Denominator (this will cause an error)
19 # Call the function and store the result
20 division_result = divide_numbers(number1, number2)
21 # Print the result to the console
22 print(f"The result of dividing {number1} by {number2} is: {division_result}")
23

```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Open folder in new window (ctrl + click)
PS C:\Users\LANJITH> & C:/Users/LANJITH/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/LANJITH/OneDrive/Desktop/AI Assist
ant coding/vote eligible.py"
The result of dividing 10 by 0 is: Error: Cannot divide by zero
PS C:\Users\LANJITH>

```

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.

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:

```

6 #task3
7 #optimize the below code that calculates the average of a list of marks and prints the corresponding grade b:
8 #the below code has a code issues and style issues and no error handling
9 # This function calculates the average of a list of marks and returns the corresponding grade
10 def calculate_grade(marks):
11     # Check if the marks list is empty to avoid division by zero error
12     if len(marks) == 0:
13         return "Error: No marks provided"
14     # Calculate the average of the marks
15     average = sum(marks) / len(marks)
16     # Determine the grade based on the average
17     if average >= 90:
18         return "Grade: A"
19     elif average >= 80:
20         return "Grade: B"
21     elif average >= 70:
22         return "Grade: C"
23     elif average >= 60:
24         return "Grade: D"
25     else:
26         return "Grade: F"
27
28 # Example usage of the calculate_grade function
29 marks_list = [85, 90, 78, 92] # List of marks
30 # Call the function and store the result
31 grade_result = calculate_grade(marks_list)
32 # Print the result to the console
33 print(f"The average grade for the marks {marks_list} is: {grade_result}")
34

```

(variable) grade_result: Literal['Error: No marks provided', 'Grade: A', 'Grade: B', 'Grade: C', 'Grade: D', 'Grade: F']

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [ ] ... | [ ] [X]
PS C:\Users\NANJITH> & C:\Users\NANJITH\AppData\Local\Programs\Python\Python314\python.exe "c:/Users/NANJITH/OneDrive/Desktop/AI Assistant coding/vote_eligible.py"
The average grade for the marks [85, 90, 78, 92] is: Grade: B
PS C:\Users\NANJITH>
```

Task Description #4: Use AI to add docstrings and inline comments to the following function.

Code:

```
C: > Users > OneDrive > Desktop > AI Assistant coding > 🗳️ vote_eligible.py > ...
1 # This function calculates the factorial of a given number n.
2 #use docstrings and inline comments
3 def factorial(n):
4     """
5     Calculate the factorial of a given number n.
6
7     Parameters:
8     n (int): The number for which to calculate the factorial. Must be a non-negative integer.
9
10    Returns:
11    int: The factorial of the given number n.
12    """
13    # Check if n is negative, if so, return an error message
14    if n < 0:
15        return "Factorial is not defined for negative numbers."
16
17    # Initialize result to 1 (the factorial of 0 is 1)
18    result = 1
19
20    # Loop from 1 to n and multiply result by each number
21    for i in range(1, n + 1):
22        result *= i
23
24    return result
25
26 # Example usage:
27 print(factorial(5)) # Output: 120
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\RANJITH> & C:/Users/RANJITH/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/RANJITH/OneDrive/Desktop/AI Assistant coding/vote_eligible.py"
120
PS C:\Users\RANJITH>
```

Task Description #5: Password Validation System (Enhanced)

Task:

1. Enhance password validation using AI assistance to include multiple security rules such as:
 - o Minimum length requirement
 - o Presence of at least one uppercase letter
 - o Presence of at least one lowercase letter

- o Presence of at least one digit
 - o Presence of at least one special character
2. Refactor the program to:
- o Use meaningful variable and function names
 - o Follow PEP 8 coding standards
 - o Include inline comments and a docstring
3. Analyze the improvements by comparing the original and AI-enhanced versions in terms of:
- o Code readability and structure
 - o Maintainability and reusability
 - o Security strength and robustness
4. Justify the AI-generated changes, explaining why each added rule and refactoring decision improves the overall quality of the program.

Code:

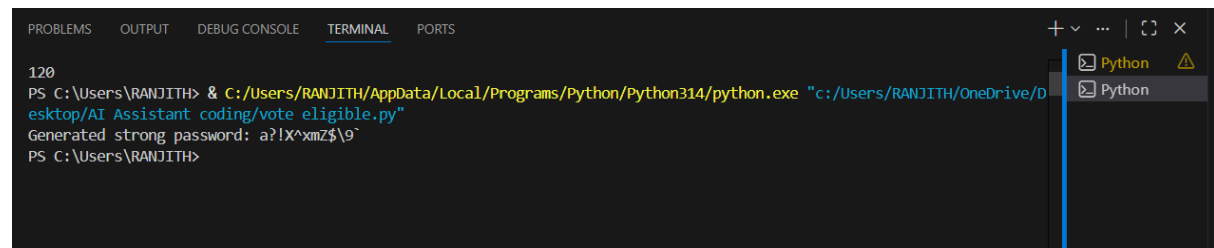
```
C:\Users\> Users > RANJITH > OneDrive > Desktop > AI Assistant coding > vote_eligible.py > ...
1  #generate a strong password by using 1 upper case,1 lower case,1 digit,1 special character and minimum length
2  #meaningful variable names and functions,pep8 coding standards,inline comments and docstrings
3  def generate_strong_password(length=12):
4      """
5      Generate a strong password with the specified length.
6
7      The password will contain at least one uppercase letter, one lowercase letter,
8      one digit, and one special character.
9
10     Parameters:
11     length (int): The desired length of the password (default is 12).
12
13     Returns:
14     str: A strong password string.
15     """
16     import random
17     import string
18
19     # Ensure the length is at least 4 to accommodate all required character types
20     if length < 4:
21         raise ValueError("Password length must be at least 4 characters.")
22
23     # Define character sets
24     uppercase_letters = string.ascii_uppercase
25     lowercase_letters = string.ascii_lowercase
26     digits = string.digits
27     special_characters = string.punctuation
28
29     # Ensure the password includes at least one character from each set
30     password = [
31         random.choice(uppercase_letters),
32         random.choice(lowercase_letters),
33         random.choice(digits),
34         random.choice(special_characters)
35     ]
```

```

86
87     # Fill the remaining length with a mix of all character sets
88     all_characters = uppercase_letters + lowercase_letters + digits + special_characters
89     password += random.choices(all_characters, k=length - 4)
90
91     # Shuffle the list to ensure randomness and convert to a string
92     random.shuffle(password)
93     return ''.join(password)
94
95 # Example usage
96 if __name__ == "__main__":
97     print("Generated strong password:", generate_strong_password(12))

```

Output:



```

120
PS C:\Users\RANDITH> & C:/Users/RANDITH/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/RANDITH/OneDrive/Desktop/AI Assistant coding/vote eligible.py"
Generated strong password: a?IX^xmZ$\\9
PS C:\Users\RANDITH>

```