

AI Assisted Coding Assignment 10.5

Name: V.Harivamsh

Hall ticket no: 2303A51266

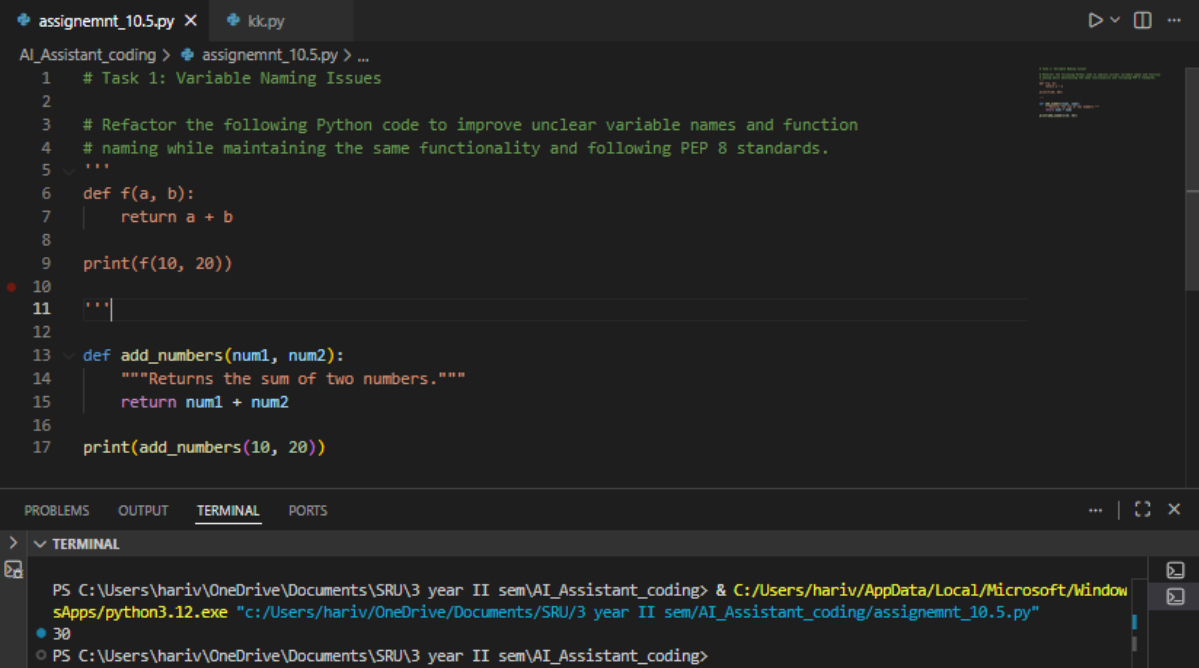
Batch no: 19

Task 1: Variable Naming Issues

Prompt:

Refactor the following Python code to improve unclear variable names and function naming while maintaining the same functionality and following PEP 8 standards. `def f(a, b): return a + b` `print(f(10, 20))`

Code & Output:



The screenshot shows a code editor with two tabs: 'assignemnt_10.5.py' and 'kk.py'. The active tab 'assignemnt_10.5.py' contains the following Python code:

```
1 # Task 1: Variable Naming Issues
2
3 # Refactor the following Python code to improve unclear variable names and function
4 # naming while maintaining the same functionality and following PEP 8 standards.
5 '''
6 def f(a, b):
7     return a + b
8
9 print(f(10, 20))
10
11 '''
12
13 def add_numbers(num1, num2):
14     """Returns the sum of two numbers."""
15     return num1 + num2
16
17 print(add_numbers(10, 20))
```

The bottom panel of the editor shows the 'TERMINAL' output:

```
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignemnt_10.5.py"
30
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

Explanation:

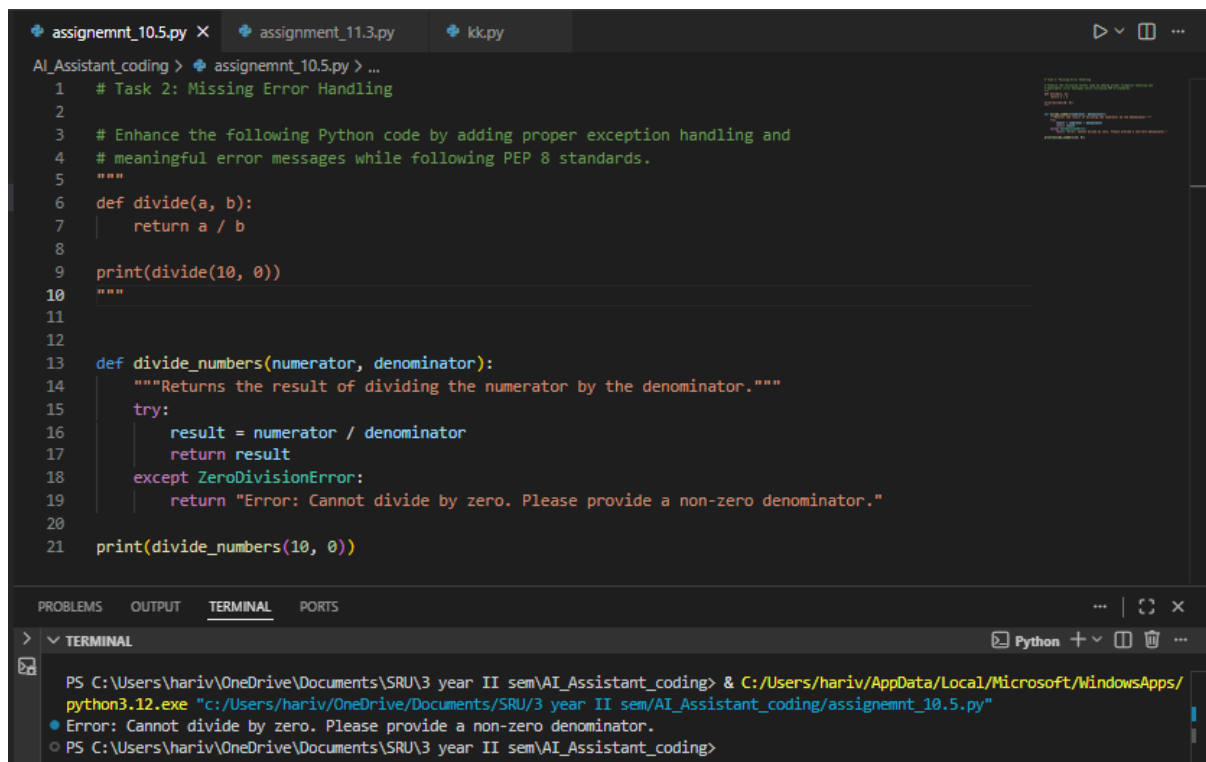
The original code uses short and unclear names like **f**, **a**, and **b**, which makes it hard to understand the purpose of the function. In the improved version, these are replaced with clear names like `add_numbers`, `first_number`, and `second_number`. This makes the code easier to read and understand at first glance. It also follows proper naming conventions, making the code cleaner and easier to maintain without changing how it works.

Task 2: Missing Error Handling

Prompt:

Enhance the following Python code by adding proper exception handling and meaningful error messages while following PEP 8 standards. `def divide(a, b): return a / b print(divide(10, 0))`

Code & Output:



```
assignment_10.5.py x assignment_11.3.py kk.py
AI_Assistant_coding > assignment_10.5.py > ...
1 # Task 2: Missing Error Handling
2
3 # Enhance the following Python code by adding proper exception handling and
4 # meaningful error messages while following PEP 8 standards.
5 """
6 def divide(a, b):
7     return a / b
8
9 print(divide(10, 0))
10 """
11
12
13 def divide_numbers(numerator, denominator):
14     """Returns the result of dividing the numerator by the denominator."""
15     try:
16         result = numerator / denominator
17         return result
18     except ZeroDivisionError:
19         return "Error: Cannot divide by zero. Please provide a non-zero denominator."
20
21 print(divide_numbers(10, 0))

PROBLEMS OUTPUT TERMINAL PORTS
> v TERMINAL Python + v [ ] [ ] [ ]
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignment_10.5.py"
Error: Cannot divide by zero. Please provide a non-zero denominator.
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

Explanation:

The original code does not check for division by zero, which can cause the program to crash at runtime. The improved version adds a try-except block to handle this situation safely. It also uses clear variable names, making the code easier to understand. This makes the

program more reliable, avoids unexpected crashes, and provides a better user experience. Proper error handling is important for writing real-world, production-quality software.

Task 3: Student Marks Processing System

Prompt:

Refactor the given student marks program to improve readability, follow PEP 8 standards, include meaningful variable names, modular functions, documentation, and basic validation.

```
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")
```

Code & Output:

Prompt:

Enhance the factorial function by adding a proper docstring and meaningful inline comments.

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

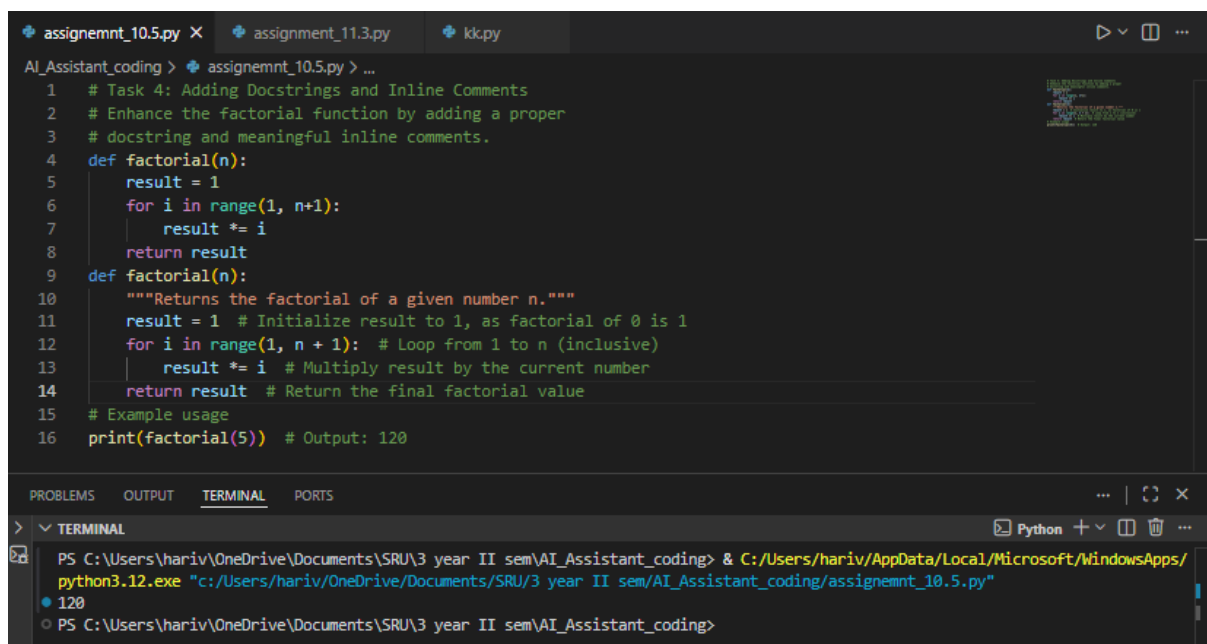
```
    result = 1
```

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

```
        result *= i
```

```
    return result
```

Code & Output:



```
AI_Assistant_coding > assignemnt_10.5.py > ...
1  # Task 4: Adding Docstrings and Inline Comments
2  # Enhance the factorial function by adding a proper
3  # docstring and meaningful inline comments.
4  def factorial(n):
5      result = 1
6      for i in range(1, n+1):
7          result *= i
8      return result
9  def factorial(n):
10     """Returns the factorial of a given number n."""
11     result = 1 # Initialize result to 1, as factorial of 0 is 1
12     for i in range(1, n + 1): # Loop from 1 to n (inclusive)
13         result *= i # Multiply result by the current number
14     return result # Return the final factorial value
15 # Example usage
16 print(factorial(5)) # Output: 120
```

```
PROBLEMS OUTPUT TERMINAL PORTS
> TERMINAL
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignemnt_10.5.py"
• 120
○ PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

Explanation:

The AI-enhanced function includes a clear docstring that explains the purpose of the function and its parameters. It also adds simple inline comments to describe each step in the logic. The variable names are improved to make their meaning clear. All these changes make the code easier to read and better documented.

Task 5: Password Validation System (Enhanced)

Prompt:

Enhance and refactor the password validation system by adding multiple security checks and improving structure according to PEP 8 standards.

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

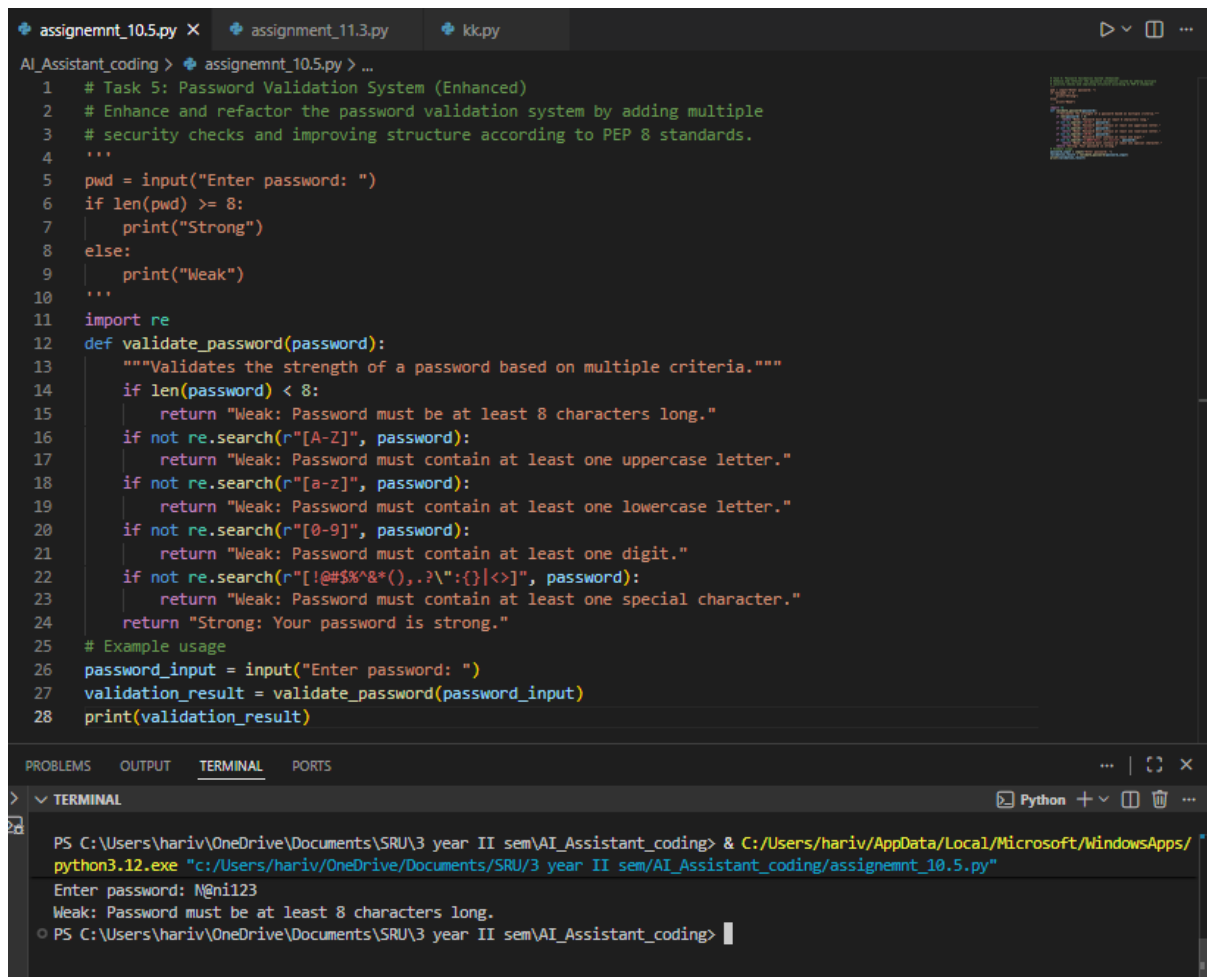
```
if len(pwd) >= 8:
```

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

```
else:
```

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

Code & Output:



```
assignemnt_10.5.py X assignment_11.3.py kk.py
AI_Assistant_coding > assignemnt_10.5.py > ...
1  # Task 5: Password Validation System (Enhanced)
2  # Enhance and refactor the password validation system by adding multiple
3  # security checks and improving structure according to PEP 8 standards.
4  '''
5  pwd = input("Enter password: ")
6  if len(pwd) >= 8:
7      print("Strong")
8  else:
9      print("Weak")
10 '''
11 import re
12 def validate_password(password):
13     """Validates the strength of a password based on multiple criteria."""
14     if len(password) < 8:
15         return "Weak: Password must be at least 8 characters long."
16     if not re.search(r"[A-Z]", password):
17         return "Weak: Password must contain at least one uppercase letter."
18     if not re.search(r"[a-z]", password):
19         return "Weak: Password must contain at least one lowercase letter."
20     if not re.search(r"[0-9]", password):
21         return "Weak: Password must contain at least one digit."
22     if not re.search(r"[!@#$%^&*(),.?\":{}|<>]", password):
23         return "Weak: Password must contain at least one special character."
24     return "Strong: Your password is strong."
25 # Example usage
26 password_input = input("Enter password: ")
27 validation_result = validate_password(password_input)
28 print(validation_result)

PROBLEMS OUTPUT TERMINAL PORTS
> v TERMINAL Python + v [ ] [ ] ...
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding> & C:/Users/hariv/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/hariv/OneDrive/Documents/SRU/3 year II sem/AI_Assistant_coding/assignemnt_10.5.py"
Enter password: N@n1123
Weak: Password must be at least 8 characters long.
PS C:\Users\hariv\OneDrive\Documents\SRU\3 year II sem\AI_Assistant_coding>
```

Explanation:

The improved program adds several password security rules, such as checking for uppercase letters, lowercase letters, numbers, and special characters. The logic is organized into a separate function with proper documentation. Compared to the original version, this one is more secure and easier to understand. It is also better structured, making it easier to maintain and update in the future.