

# ASSIGNMENT- 10.5

Name:A.Abhilash Goud

HT.No: 2303A51359

Batch: 20

## Task Description #1 – Variable Naming Issues

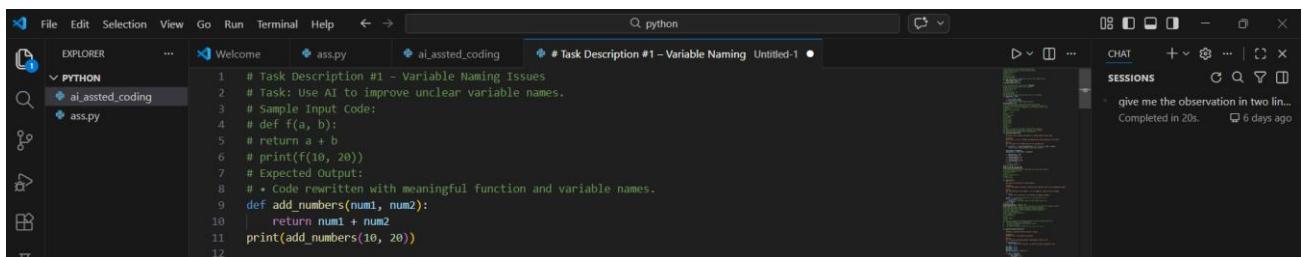
Sample input :

```
def f(a,b):
```

```
    return a+b
```

```
Print(f(10,20))
```

## Code & probmt

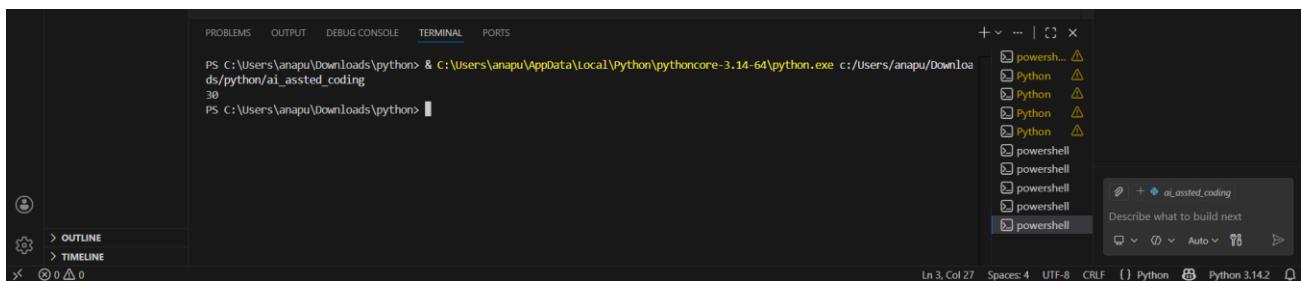


The screenshot shows the Visual Studio Code interface. The code editor has a Python file named 'ass.py' open, containing the following code:

```
1  # Task Description #1 - Variable Naming Issues
2  # Task: Use AI to improve unclear variable names.
3  # Sample Input code:
4  def f(a, b):
5      # return a + b
6  # print(f(10, 20))
7  # Expected Output:
8  # ✘ Code rewritten with meaningful function and variable names.
9  def add_numbers(num1, num2):
10     return num1 + num2
11  print(add_numbers(10, 20))
```

The terminal tab shows the command 'python ass.py' being run, resulting in the output '30'.

## Result:



The screenshot shows the VS Code terminal tab with the following output:

```
PS C:\Users\anapu\Downloads\python> & C:\Users\anapu\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/anapu/Downloads/python/ai_assed_coding
30
PS C:\Users\anapu\Downloads\python>
```

## Observation:

The original code used unclear and non-descriptive names, making it hard to understand the function's purpose. Using meaningful names improves clarity, readability, and overall code quality.

## # Task Description #2 – Missing Error Handling

### Sample Input Code:

```
# def divide(a, b):
```

```
# return a / b
```

```
# print(divide(10, 0))
```

## Prompt & Code:

The screenshot shows the Visual Studio Code interface. In the top bar, there are tabs for 'Welcome', 'ass.py', 'ai\_assted\_coding', and '# Task Description #2 – Missing Error Handling'. The 'ai\_assted\_coding' tab is active, displaying the following Python code:

```
1 # Task Description #2 - Missing Error Handling
2 # Task: Use AI to add proper error handling.
3 # Sample Input code:
4 # def divide(a, b):
5 #     return a / b
6 # print(divide(10, 0))
7 # Expected output:
8 # * Code with exception handling and clear error messages.
9 def divide(num1, num2):
10    try:
11        return num1 / num2
12    except ZeroDivisionError:
13        return "Error: Cannot divide by zero."
14 print(divide(10, 0))
```

In the bottom right corner of the code editor, there is a 'SESSIONS' panel showing a single session named 'ai\_assted\_coding'.

## Result:

The screenshot shows a terminal window with the following output:

```
PS C:\Users\anapu\Downloads\python> & C:\Users\anapu\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/anapu/Downloads/python/ai_assted_coding
Error: Cannot divide by zero.
PS C:\Users\anapu\Downloads\python>
```

The terminal window also includes a sidebar with file navigation and a status bar at the bottom.

## Observation:

The original code did not handle division by zero, which could cause the program to crash. Adding exception handling makes the program safer and provides a clear, user-friendly error message instead of stopping abruptly.

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

Sample input code:

```
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")
```

## Prompt & Code:

The screenshot shows the Visual Studio Code interface with the following details:

- File Structure:** The Explorer sidebar shows a folder named "ai\_assted\_coding" containing two files: "ass.py" and "ai\_assted\_coding.py".
- Code Editor:** The main editor window displays Python code for calculating student marks and determining grades. The code includes comments explaining its purpose and follows PEP 8 standards.
- Terminal:** The terminal at the bottom shows the command "PS C:\Users\anapu\Downloads\python> & C:\Users\anapu\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/anapu/Downloads/python/ai\_assted\_coding" followed by the output "Grade: B".
- Status Bar:** The status bar at the bottom right indicates the date as "2/20/2026" and the time as "2:29 PM".

## Result:

The terminal output shows the execution of the Python script and its result:

```
PS C:\Users\anapu\Downloads\python> & C:\Users\anapu\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/anapu/Downloads/python/ai_assted_coding
Grade: B
PS C:\Users\anapu\Downloads\python>
```

## Observation:

The refactored code improves readability by using meaningful variable names, proper formatting, functions, and clear documentation following PEP 8 standards.

It also adds basic input validation, making the program more reliable and user-friendly by preventing errors from invalid data.

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

Sample input:

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

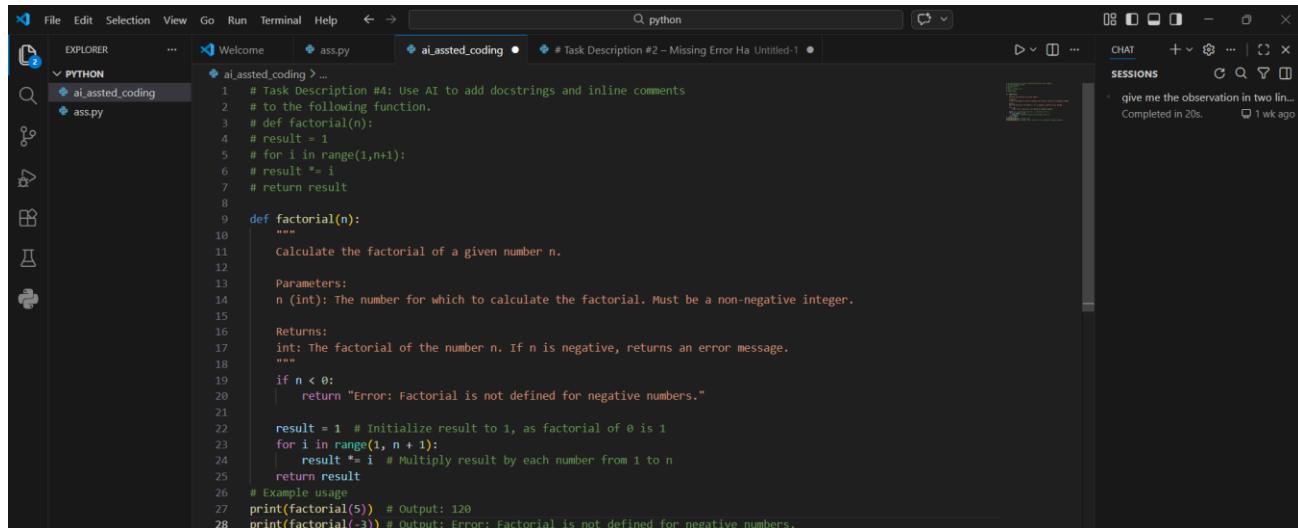
```
    result = 1
```

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

```
        result *= i
```

```
    return result
```

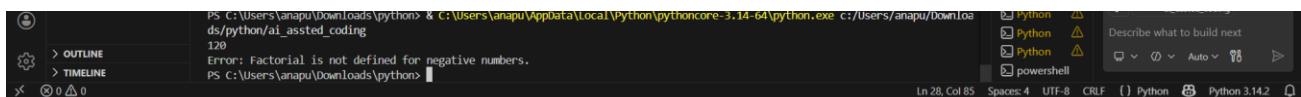
## Prompt & Code:



The screenshot shows a code editor interface with several tabs open. The active tab contains the following Python code:

```
1  # Task Description #4: Use AI to add docstrings and inline comments
2  # To the following function.
3  # def factorial(n):
4  #     result = 1
5  #     for i in range(1,n+1):
6  #         result *= i
7  #     return result
8
9  def factorial(n):
10     """
11     Calculate the factorial of a given number n.
12
13     Parameters:
14     n (int): The number for which to calculate the factorial. Must be a non-negative integer.
15
16     Returns:
17     int: The factorial of the number n. If n is negative, returns an error message.
18     """
19     if n < 0:
20         return "Error: Factorial is not defined for negative numbers."
21
22     result = 1 # Initialize result to 1, as factorial of 0 is 1
23     for i in range(1, n + 1):
24         result *= i # Multiply result by each number from 1 to n
25     return result
26
27 # Example usage
28 print(factorial(5)) # Output: 120
29 print(factorial(-3)) # Output: Error: Factorial is not defined for negative numbers.
```

## Result:



The terminal window shows the command being run and the resulting output:

```
PS C:\Users\anapu\Downloads\python> & C:\Users\anapu\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/anapu/Downloads/python/ai_assisted_coding
120
Error: Factorial is not defined for negative numbers.
PS C:\Users\anapu\Downloads\python>
```

## Observation

The updated function includes a clear docstring and inline comments, which improve understanding of the code's purpose and logic.

It also adds basic validation for negative numbers, making the function more informative and reliable.

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

Sample input:

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

if len(pwd) >= 8:
    print("Strong")

else:
    print("Weak")
```

Prompt & Code:

The screenshot shows a code editor window with a Python script named `ai_assisted_coding.py`. The code implements a password validation function that checks for length, uppercase letters, lowercase letters, digits, and special characters. It includes a docstring and several test cases at the bottom. The code editor has a status bar showing the file path, line count (Ln 59), column count (Col 81), and other details.

```
1 # Task Description #5: Password Validation System (Enhanced)
2 # The following Python program validates a password using only a
3 # single if statement. This is insufficient for real-world
4 # security requirements.
5 # Enter your password:
6 # Enter password: "Strong"
7 # print("Strong")
8 # print("Weak")
9 #
10 # Task:
11 # Add to enhance the password validation system by adding checks for:
12 # - Minimum length (e.g., 8 characters)
13 # - Presence of uppercase and lowercase letters
14 # - At least one digit
15 # - At least one special character (e.g., @, #, $, %, etc.)
16
17 def validate_password(password):
18     """
19         Validates a password based on multiple criteria.
20
21     Parameters:
22         password (str): The password to validate.
23
24     Returns:
25         str: A message indicating whether the password is valid or not.
26
27     """
28     if len(password) < 8:
29         return "Password is too short. It must be at least 8 characters long."
30
31     has_upper = False
32     has_lower = False
33     has_digit = False
34     special_characters = "#@%$%^!&*"
35
36     for char in password:
37         if char.isupper():
38             has_upper = True
39         elif char.islower():
40             has_lower = True
41         elif char.isdigit():
42             has_digit = True
43         elif char in special_characters:
44             has_special = True
45
46     if not has_upper:
47         return "Password must contain at least one uppercase letter."
48
49     if not has_lower:
50         return "Password must contain at least one lowercase letter."
51
52     if not has_digit:
53         return "Password must contain at least one digit."
54
55     return "Password is valid."
56
57 # Example usage
58 print(validate_password("MyPass123")) # Should be valid
59 print(validate_password("MyPass123!")) # Should be invalid due to missing uppercase, digit, and special character
60 print(validate_password("short1!")) # Should be invalid due to being too short!
```

Result:

The screenshot shows a terminal window with the command `python ai_assisted_coding.py` run. The output shows the validation of three different passwords: a valid one ('MyPass123'), an invalid one ('MyPass123!'), and another invalid one ('short1!'). The terminal also shows the environment variables and the Python version.

```
PS C:\Users\anapu\Downloads\python> & C:\Users\anapu\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/anapu/Downloads/python/ai_assisted_coding.py
Error: factorial is not defined for negative numbers.
PS C:\Users\anapu\Downloads\python> & C:\Users\anapu\AppData\Local\Python\pythoncore-3.14-64\python.exe c:/Users/anapu/Downloads/python/ai_assisted_coding.py
Password is valid.
Password must contain at least one uppercase letter.
Password is too short. It must be at least 8 characters long.
PS C:\Users\anapu\Downloads\python>
```

**Observation:**

The enhanced password validation improves security by checking for length, uppercase and lowercase letters, digits, and special characters instead of just minimum length.

However, there is a small issue: the variable `has_special` is used without being initialized and the final check for a special character is missing, which could cause an error.