

ASSIGNMENT- 10.5

Name: A. Abhilash Goud

HT.No: 2303A51359

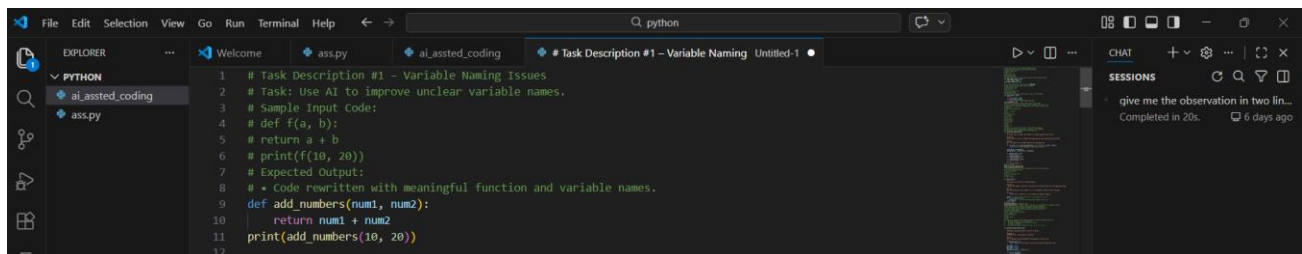
Batch: 20

Task Description #1 – Variable Naming Issues

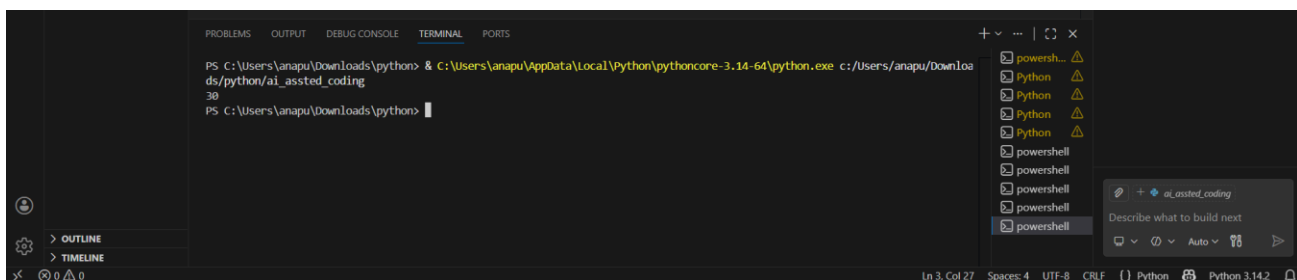
Sample input :

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

Code & probmt



Result:



Observation:

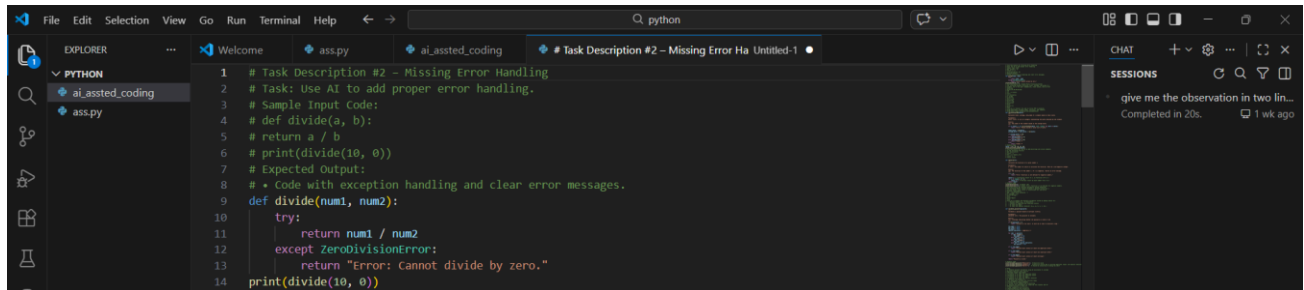
The original code used unclear and non-descriptive names, making it hard to understand the function's purpose. Using meaningful names improves clarity, readability, and overall code quality.

Task Description #2 – Missing Error Handling

Sample Input Code:

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

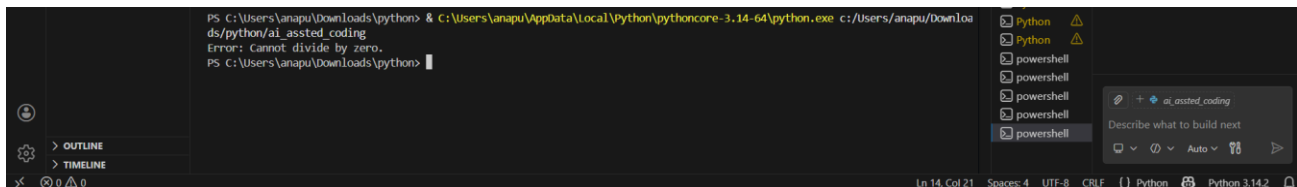
Prompt & Code:



The screenshot shows the Visual Studio Code editor with a Python file named `ass.py`. The code implements a `divide` function with exception handling for `ZeroDivisionError`. A chat window on the right shows a prompt asking for an observation in two lines, which has been completed in 20 seconds.

```
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))
```

Result:



The screenshot shows a terminal window where the Python script is executed. The output is an error message: "Error: Cannot divide by zero." The terminal also shows the command used to run the script and the current directory.

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

Observation:

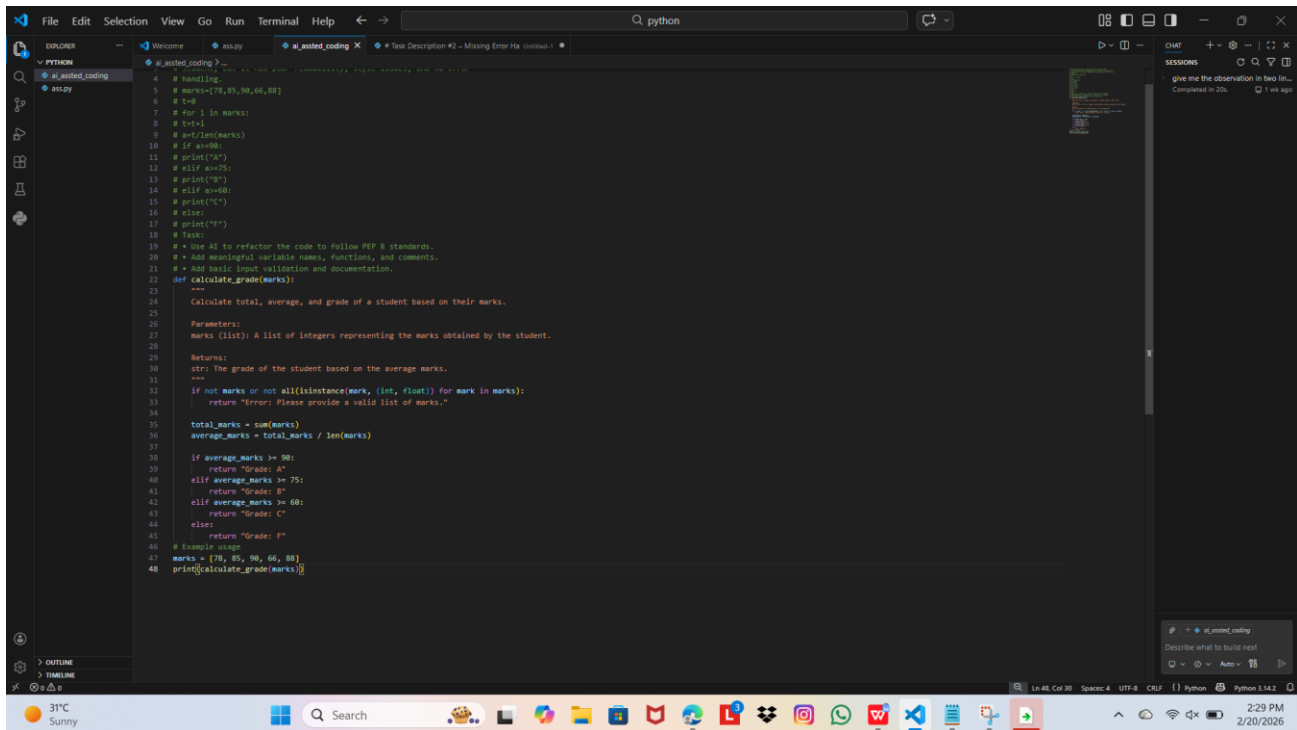
The original code did not handle division by zero, which could cause the program to crash. Adding exception handling makes the program safer and provides a clear, user-friendly error message instead of stopping abruptly.

Task Description #3: Student Marks Processing System

Sample input code:

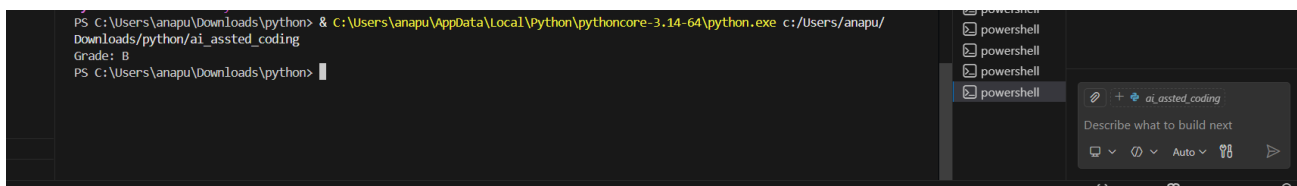
```
marks=[78,85,90,66,88]
t=0
for i in marks:
    t=t+i
a=t/len(marks)
if a>=90:
    print("A")
elif a>=75:
    print("B")
elif a>=60:
    print("C")
else:
    print("F")
```

Prompt & Code:



```
1 # Prompt: Refactor the following code to follow PEP 8 standards.
2 # Use meaningful variable names, proper formatting, and clear documentation.
3 # Add basic input validation to ensure the program is user-friendly.
4 # Example: marks = [78, 85, 90, 66, 88]
5 # Expected output: Grade: B
6
7 # Original code:
8 # marks = [78, 85, 90, 66, 88]
9 # total = 0
10 # for i in marks:
11 #     total += i
12 # average = total / len(marks)
13 # if average >= 90:
14 #     print("A")
15 # elif average >= 75:
16 #     print("B")
17 # elif average >= 60:
18 #     print("C")
19 # else:
20 #     print("F")
21
22 # Refactored code:
23
24 """
25 Calculate total, average, and grade of a student based on their marks.
26
27 Parameters:
28 marks (list): A list of integers representing the marks obtained by the student.
29
30 Returns:
31 str: The grade of the student based on the average marks.
32 """
33 if not marks or not all(isinstance(mark, (int, float)) for mark in marks):
34     return "Error: Please provide a valid list of marks."
35
36 total_marks = sum(marks)
37 average_marks = total_marks / len(marks)
38
39 if average_marks >= 90:
40     return "Grade: A"
41 elif average_marks >= 75:
42     return "Grade: B"
43 elif average_marks >= 60:
44     return "Grade: C"
45 else:
46     return "Grade: F"
47
48 # Example usage:
49 marks = [78, 85, 90, 66, 88]
50 print(calculate_grade(marks))
```

Result:



```
PS C:\Users\anapu\Downloads\python> python ai_assisted_coding.py
Grade: B
PS C:\Users\anapu\Downloads\python>
```

Observation:

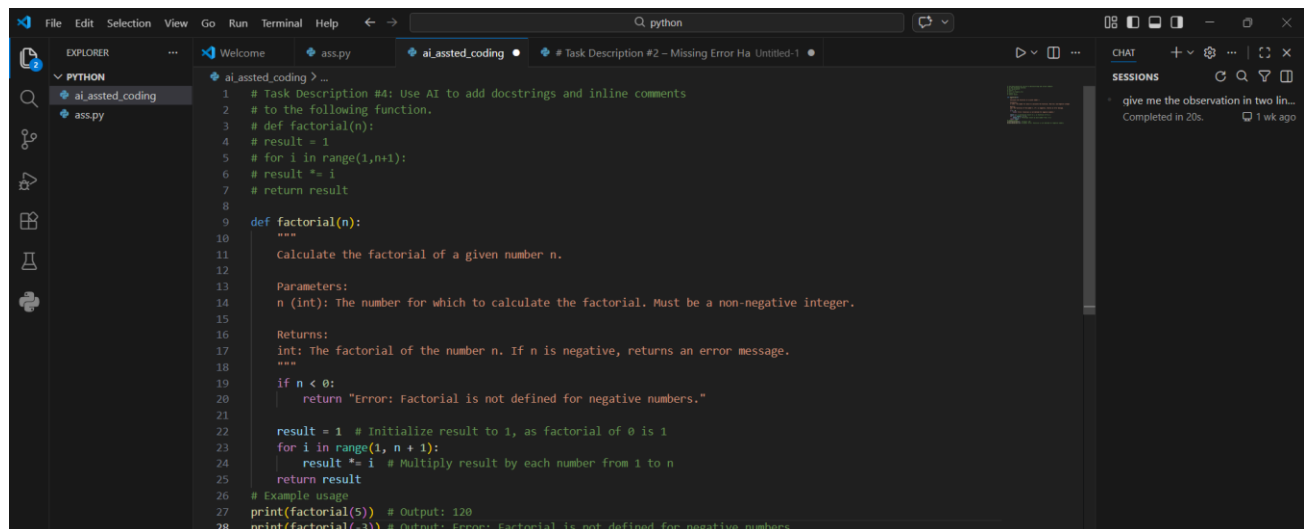
The refactored code improves readability by using meaningful variable names, proper formatting, functions, and clear documentation following PEP 8 standards. It also adds basic input validation, making the program more reliable and user-friendly by preventing errors from invalid data.

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
```

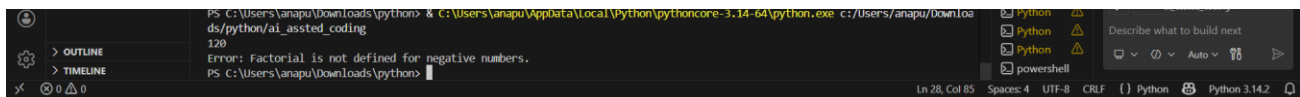
Prompt & Code:



The screenshot shows the VS Code editor with the file explorer on the left showing 'ai_assted_coding' and 'ass.py'. The main editor window displays the updated Python code for the factorial function. The code includes a docstring with a description, parameters, and returns, as well as inline comments for each step of the function's logic. The chat panel on the right shows a session with the prompt 'give me the observation in two lin...' and a response 'Completed in 20s. 1 wk ago'.

```
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 screenshot shows the terminal output of the Python script. The first line shows the command 'PS C:\Users\anapu\Downloads\python> & C:\Users\anapu\AppData\Local\Python\pythoncore-3.14-64\python.exe c:\Users\anapu\downloa ds/python/ai_assted_coding' and the output '120'. The second line shows the command 'PS C:\Users\anapu\Downloads\python>' and the output '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\downloa ds/python/ai_assted_coding  
120  
PS C:\Users\anapu\Downloads\python> 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: ")
```

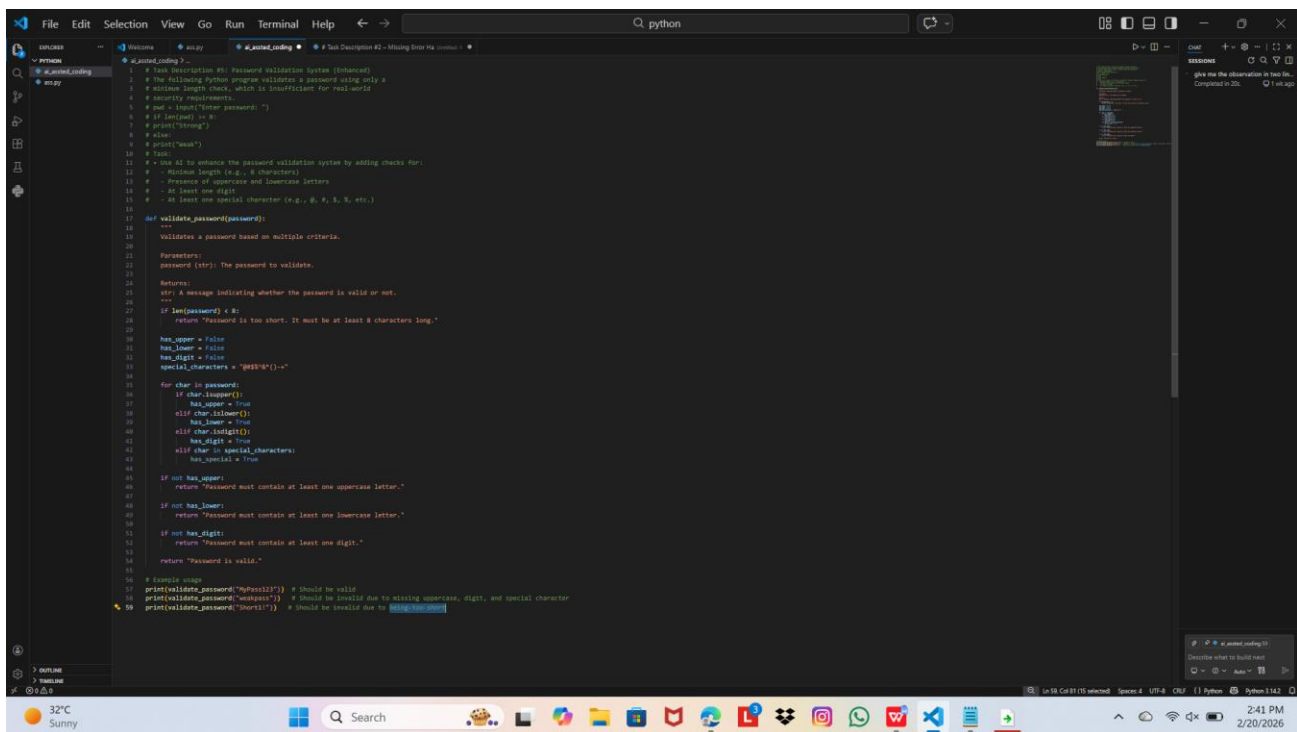
```
if len(pwd) >= 8:
```

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

```
else:
```

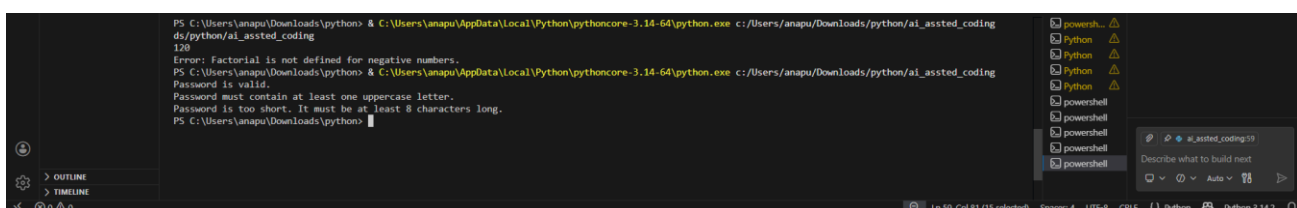
```
    print("Weak")
```

Prompt & Code:



```
1 # Task Description #5: Password Validation System (Enhanced)
2 # The following Python program validates a password using only a
3 # minimum length check, which is insufficient for real-world
4 # security requirements.
5 # pwd = input("Enter password: ")
6 # if len(pwd) >= 8:
7 #     print("Strong")
8 # else:
9 #     print("Weak")
10 # Task
11 # Use AI 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., !, @, #, $, %, etc.)
16
17 def validate_password(password):
18     """
19     Validate a password based on multiple criteria.
20
21     Parameters:
22     password (str): The password to validate.
23
24     Returns:
25     str: A message indicating whether the password is valid or not.
26     """
27     if len(password) < 8:
28         return "Password is too short. It must be at least 8 characters long."
29
30     has_upper = False
31     has_lower = False
32     has_digit = False
33     special_characters = "!@#$%^&*()-_+="
34
35     for char in password:
36         if char.isupper():
37             has_upper = True
38         elif char.islower():
39             has_lower = True
40         elif char.isdigit():
41             has_digit = True
42         elif char in special_characters:
43             has_special = True
44
45     if not has_upper:
46         return "Password must contain at least one uppercase letter."
47     if not has_lower:
48         return "Password must contain at least one lowercase letter."
49     if not has_digit:
50         return "Password must contain at least one digit."
51     if not has_special:
52         return "Password must contain at least one special character."
53     return "Password is valid."
54
55 # Example usage
56 print(validate_password("MyP@ssw0rd!")) # Should be valid
57 print(validate_password("weakpass")) # Should be invalid due to missing uppercase, digit, and special character
58 print(validate_password("short1")) # Should be invalid due to being too short
```

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
ds/python/ai_assted_coding
120
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_assted_coding
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.