ASSIGNMENT- 10.5

Name:A.Abhilash Goud

HT.No: 2303A51359

Batch: 20

Task Description #1 – Variable Naming Issues

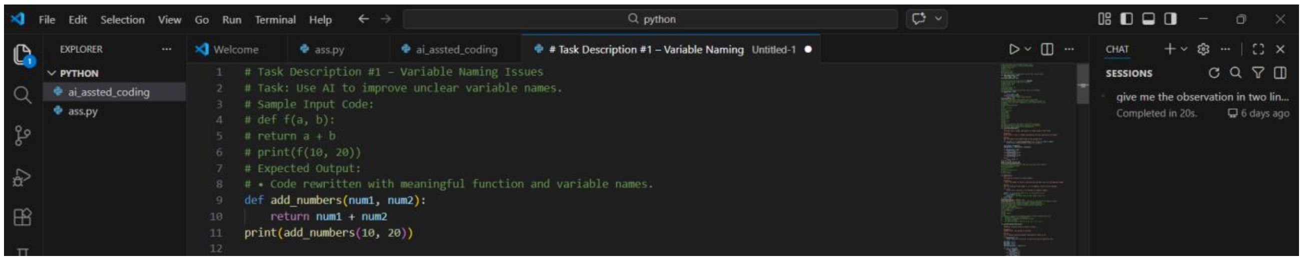
Sample input :

def f(a,b):

return a+b

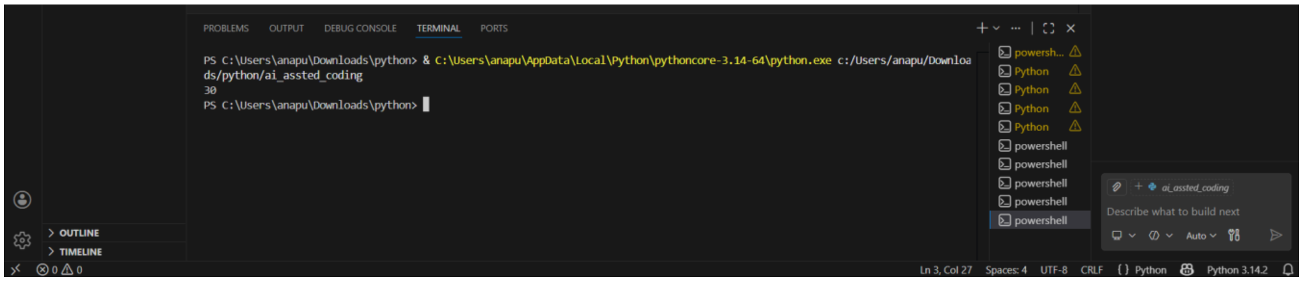
Print(f(10,20)

Code & probmt



```python
# Task Description #1 – Variable Naming Issues
# Task: Use AI to improve unclear variable names.
# Sample Input Code:
# def f(a, b):
# return a + b
# print(f(10, 20))
# Expected Output:
# • Code rewritten with meaningful function and variable names.
def add_numbers(num1, num2):
    return num1 + num2
print(add_numbers(10, 20))
```

Result:



```
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
30
PS C:\Users\anapu\Downloads\python> 
```

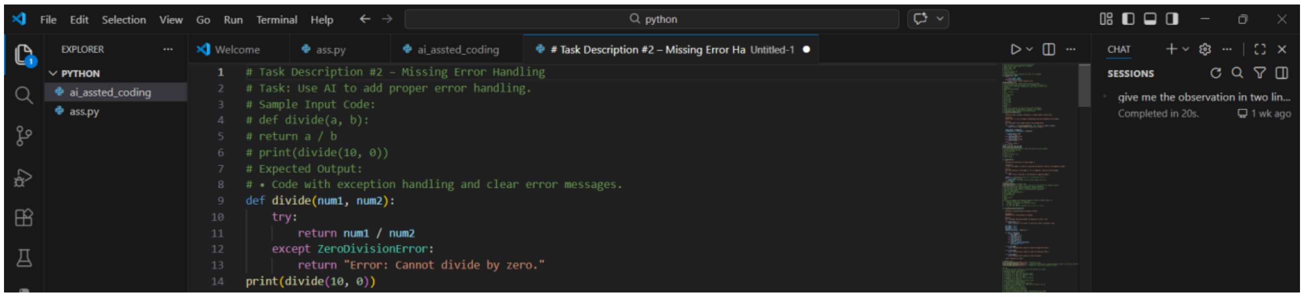
Observation:

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

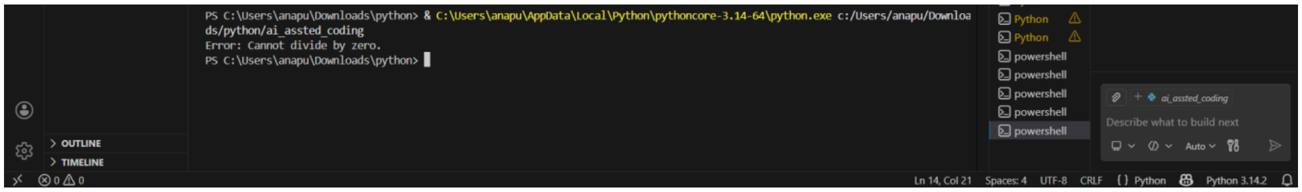
Sample Input Code:
# def divide(a, b):
# return a / b
# print(divide(10, 0))

Prompt &Code:



```python
# 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.
def divide(num1, num2):
    try:
        return num1 / num2
    except ZeroDivisionError:
        return "Error: Cannot divide by zero."
print(divide(10, 0))
```

Result:



```
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")

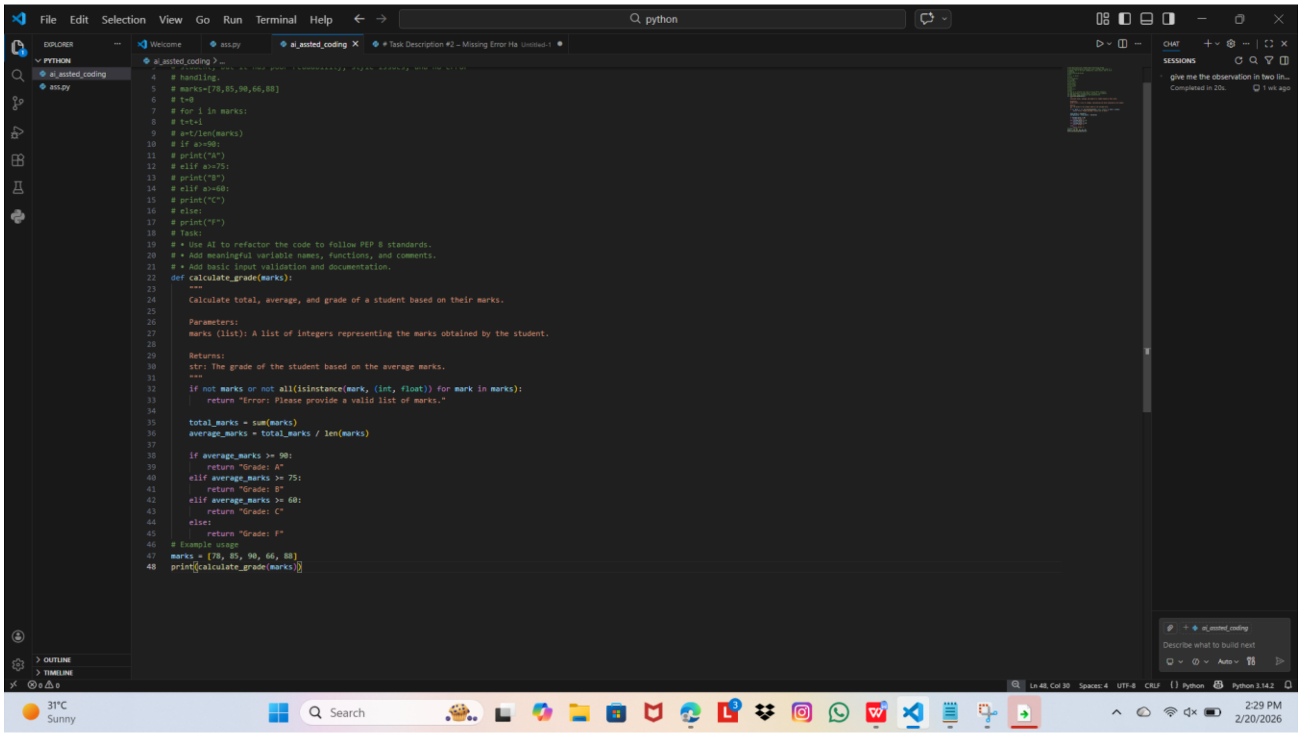
elif a>=60:

print("C")

else:

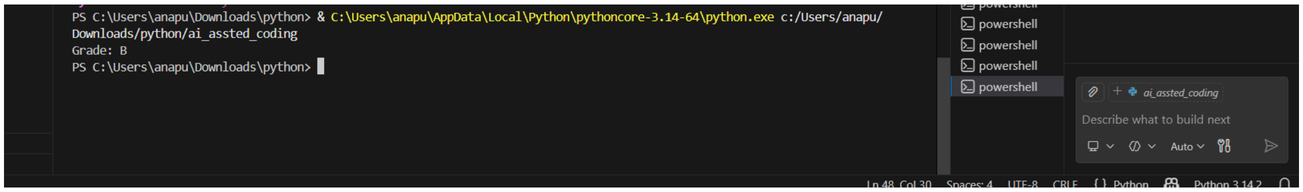
print("F")

Prompt & Code:



```python
# 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.
def calculate_grade(marks):
    """
    Calculate total, average, and grade of a student based on their marks.

    Parameters:
    marks (list): A list of integers representing the marks obtained by the student.

    Returns:
    str: The grade of the student based on the average marks.
    """
    if not marks or not all(isinstance(mark, (int, float)) for mark in marks):
        return "Error: Please provide a valid list of marks."

    total_marks = sum(marks)
    average_marks = total_marks / len(marks)

    if average_marks >= 90:
        return "Grade: A"
    elif average_marks >= 75:
        return "Grade: B"
    elif average_marks >= 60:
        return "Grade: C"
    else:
        return "Grade: F"
# Example usage
marks = [78, 85, 90, 66, 88]
print(calculate_grade(marks))
```

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):

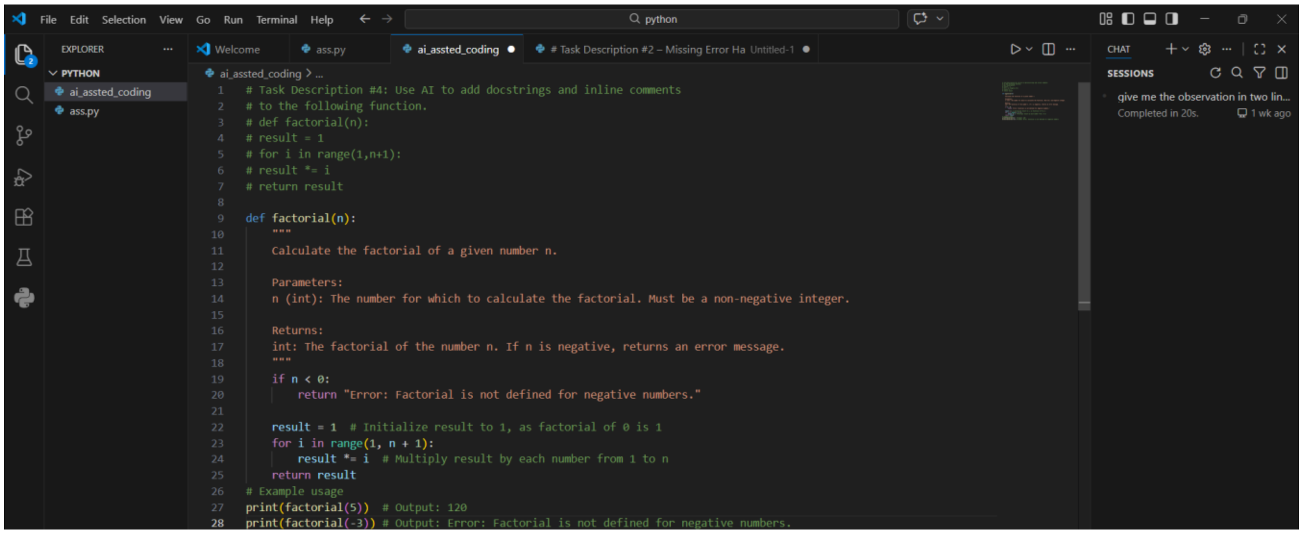
result = 1

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

result *= i

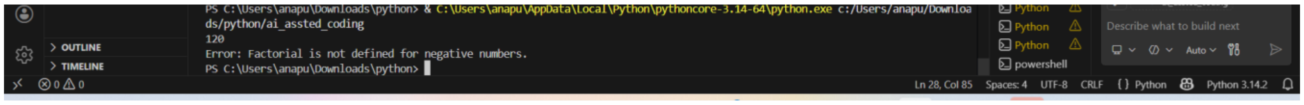
return result

Prompt &Code:



```python
# 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

def factorial(n):
    """
    Calculate the factorial of a given number n.

    Parameters:
    n (int): The number for which to calculate the factorial. Must be a non-negative integer.

    Returns:
    int: The factorial of the number n. If n is negative, returns an error message.
    """
    if n < 0:
        return "Error: Factorial is not defined for negative numbers."

    result = 1  # Initialize result to 1, as factorial of 0 is 1
    for i in range(1, n + 1):
        result *= i  # Multiply result by each number from 1 to n
    return result
# Example usage
print(factorial(5))  # Output: 120
print(factorial(-3)) # Output: Error: Factorial is not defined for negative numbers.
```

Result:



```
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
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: ")

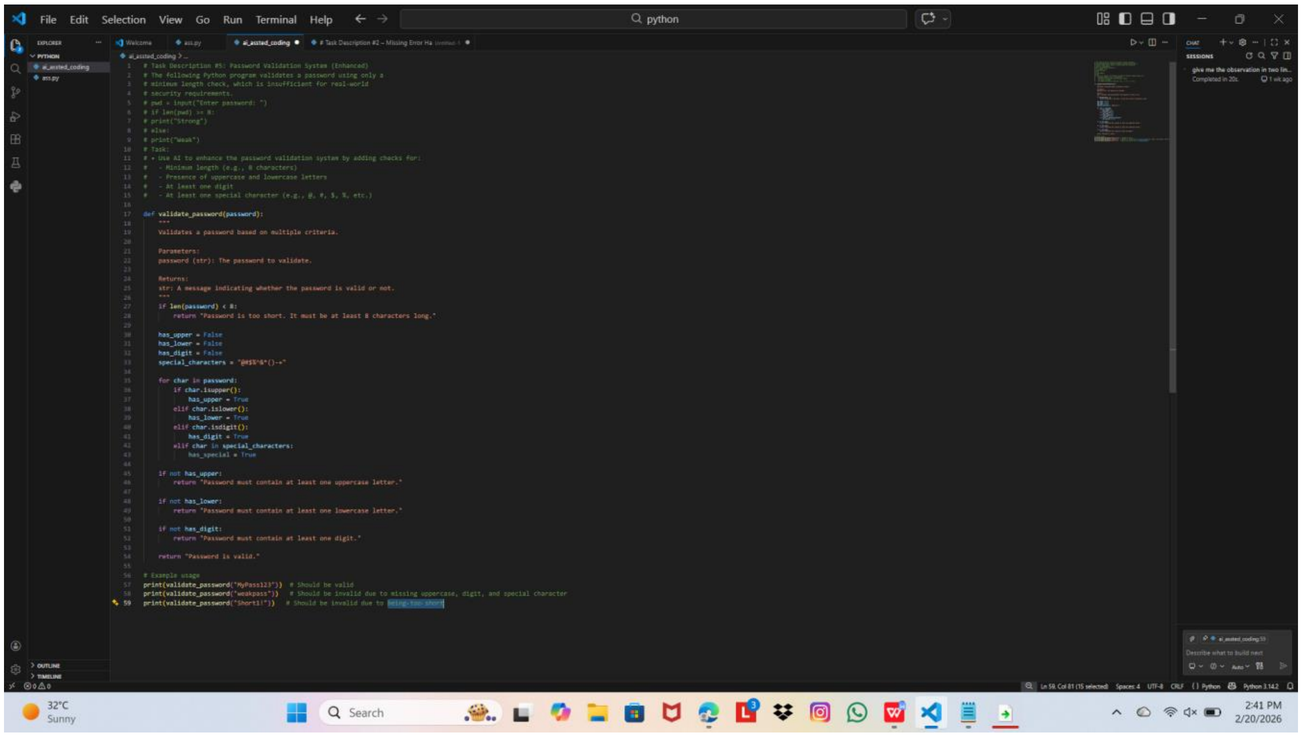
if len(pwd) >= 8:

print("Strong")

else:

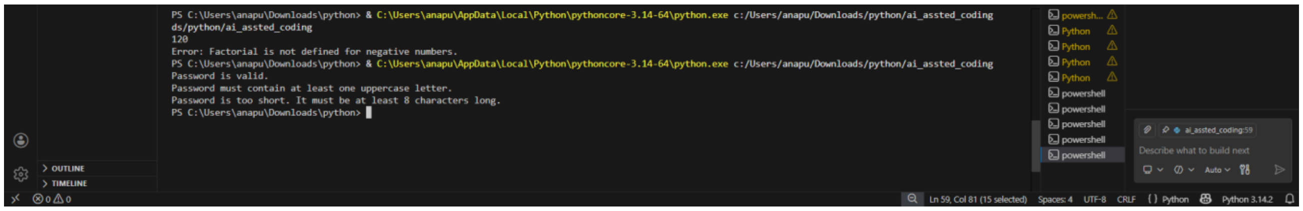
print("Weak")

Prombt & Code:



```python
# 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")
# Task:
# • Use AI to enhance the password validation system by adding checks for:
#   - Minimum length (e.g., 8 characters)
#   - Presence of uppercase and lowercase letters
#   - At least one digit
#   - At least one special character (e.g., @, #, $, %, etc.)

def validate_password(password):
    """
    Validates a password based on multiple criteria.

    Parameters:
    password (str): The password to validate.

    Returns:
    str: A message indicating whether the password is valid or not.
    """
    if len(password) < 8:
        return "Password is too short. It must be at least 8 characters long."

    has_upper = False
    has_lower = False
    has_digit = False
    special_characters = "@#$%^&*()-+"

    for char in password:
        if char.isupper():
            has_upper = True
        elif char.islower():
            has_lower = True
        elif char.isdigit():
            has_digit = True
        elif char in special_characters:
            has_special = True

    if not has_upper:
        return "Password must contain at least one uppercase letter."

    if not has_lower:
        return "Password must contain at least one lowercase letter."

    if not has_digit:
        return "Password must contain at least one digit."

    return "Password is valid."

# Example usage
print(validate_password("MyPass123"))  # Should be valid
print(validate_password("weakpass"))   # Should be invalid due to missing uppercase, digit, and special character
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.