

AI ASSISTED CODING

LAB_Assignment_10.5.

2303A52197

BATCH - 35

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.

The screenshot shows the Google Colab interface. On the left, there's a sidebar with icons for file operations like copy, paste, and refresh. The main area contains a code cell with the following Python code:

```
[1] def add_numbers(num1, num2):
    """
    This function takes two numbers as input
    and returns their sum.
    """
    sum_result = num1 + num2
    return sum_result

# Calling the function and printing the result
result = add_numbers(10, 20)
print("The sum is:", result)
```

The output of the cell is "The sum is: 30". To the right of the code cell, there's a "Release notes" sidebar. It shows the date "2026-01-20" and a list of changes:

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook.
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

At the bottom of the sidebar, it says "Python package upgrades" and lists "accelerate 1.11.0 -> 1.12.0".

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

The screenshot shows a Google Colab notebook titled "Untitled24.ipynb". The code cell contains the following Python code:

```
def divide_numbers(num1, num2):
    """
    This function divides num1 by num2
    and handles possible errors.
    """
    try:
        result = num1 / num2
        return result
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."
    except TypeError:
        return "Error: Please enter valid numeric values."

# Calling the function
output = divide_numbers(10, 0)
print(output)
```

The output of the code cell is:

```
... Error: Division by zero is not allowed.
```

The Colab interface includes a sidebar with file navigation, a toolbar with various icons, and a status bar at the bottom right showing RAM and Disk usage.

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.

```
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_total_and_average(marks_list):
    """
    This function calculates the total and average
    of a list of student marks.
    """
    total_marks = sum(marks_list)
    average_marks = total_marks / len(marks_list)
    return total_marks, average_marks

def calculate_grade(average_marks):
    """
    This function returns the grade
    based on the average marks.
    """
    if average_marks >= 90:
        return "A"
    elif average_marks >= 75:
        return "B"
    elif average_marks >= 60:
        return "C"
    else:
        return "F"

def validate_marks(marks_list):
    """
    This function checks whether the marks list
    contains valid numeric values between 0 and 100.
    """

```

Release notes

Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4
- dataproc-spark-connect 0.8.3 -> 1.0.1
- diffusers 0.35.2 -> 0.36.0
- flax 0.10.7 -> 0.11.2
- google 2.0.3 -> 3.0.0
- google-adk 1.17.0 -> 1.21.0
- google-auth 2.38.0 -> 2.43.0
- google-genai 1.49.0 -> 1.55.0
- gradio 5.49.1 -> 5.50.0
- holidays 0.84 -> 0.88
- humanize 4.14.0 -> 4.15.0
- langchain 0.3.27 -> 1.2.4
- langsmt 0.4.42 -> 0.6.4

```

def validate_marks(marks_list):
    """
    This function checks whether the marks list
    contains valid numeric values between 0 and 100.
    """
    if not marks_list:
        return False

    for mark in marks_list:
        if not isinstance(mark, (int, float)) or mark < 0 or mark > 100:
            return False

    return True

# Main Program
student_marks = [78, 85, 90, 66, 88]

if validate_marks(student_marks):
    total, average = calculate_total_and_average(student_marks)
    grade = calculate_grade(average)

    print("Total Marks:", total)
    print("Average Marks:", average)
    print("Grade:", grade)
else:
    print("Error: Invalid marks data provided.")

```

Release notes

Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4
- dataproc-spark-connect 0.8.3 -> 1.0.1
- diffusers 0.35.2 -> 0.36.0
- flax 0.10.7 -> 0.11.2
- google 2.0.3 -> 3.0.0
- google-adk 1.17.0 -> 1.21.0
- google-auth 2.38.0 -> 2.43.0
- google-genai 1.49.0 -> 1.55.0
- gradio 5.49.1 -> 5.50.0
- holidays 0.84 -> 0.88
- humanize 4.14.0 -> 4.15.0
- langchain 0.3.27 -> 1.2.4
- langsmt 0.4.42 -> 0.6.4

```

# Main Program
student_marks = [78, 85, 90, 66, 88]

if validate_marks(student_marks):
    total, average = calculate_total_and_average(student_marks)
    grade = calculate_grade(average)

    print("Total Marks:", total)
    print("Average Marks:", average)
    print("Grade:", grade)
else:
    print("Error: Invalid marks data provided.")

...
Total Marks: 407
Average Marks: 81.4
Grade: B

```

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):
    """
    This function calculates the factorial of a given number.

    Parameters:
    n (int): A positive integer

    Returns:
    int: Factorial of the given number
    """
    result = 1 # Initialize result to 1

    # Loop from 1 to n (inclusive)
    for i in range(1, n + 1):
        result *= i # Multiply result by i

    return result # Return the final factorial value

# Function call and output
print(factorial(5))

```

Release notes

Please follow our [blog](#) to see more information about new features, tips tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook.
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available via use in Antigravity, Cursor, and Windsurf via the Open VSX Registry.
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.1.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4
- dataproc-spark-connect 0.8.3 -> 1.0.1
- diffusers 0.35.2 -> 0.36.0
- flax 0.10.7 -> 0.11.2
- google 2.0.3 -> 3.0.0
- google-adx 1.17.0 -> 1.21.0
- google-auth 2.38.0 -> 2.43.0
- google-genie 1.49.0 -> 1.55.0
- gradio 5.49.1 -> 5.50.0
- hailfish 0.94 -> 0.98

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:

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.

The screenshot shows a Google Colab notebook titled "Untitled24.ipynb". The code defines a function `is_strong_password` that checks if a password is strong based on multiple security rules. The rules include a minimum length of 8 characters, at least one uppercase letter, one lowercase letter, one digit, and one special character. The code uses a for loop to iterate through each character in the password and checks its type using `isupper()`, `islower()`, `isdigit()`, and `isalnum()` methods. The function returns `False` if any rule is not met, otherwise it returns `True`. The sidebar on the right displays the "Release notes" for January 2026, which mention the launch of Data Explorer and Gemini 3.

```
def is_strong_password(password):
    """
    This function checks whether a password is strong
    based on multiple security rules.

    Rules:
    1. Minimum length of 8 characters
    2. At least one uppercase letter
    3. At least one lowercase letter
    4. At least one digit
    5. At least one special character
    """
    if len(password) < 8:
        return False

    has_uppercase = False
    has_lowercase = False
    has_digit = False
    has_special_character = False

    for character in password:
        if character.isupper():
            has_uppercase = True
        elif character.islower():
            has_lowercase = True
        elif character.isdigit():
            has_digit = True
        elif not character.isalnum():
            has_special_character = True

    return (
        has_uppercase
        and has_lowercase
        and has_digit
        and has_special_character
    )
```

The screenshot shows the same Google Colab notebook after AI refactoring. The code has been simplified and organized. The main logic for checking password rules is moved into a separate function `for character in password:`. The original function `is_strong_password` now only contains a call to this helper function and handles the user input. The AI has also added a main program section where it prompts the user for a password and prints "Strong password" if the condition is met, or "Weak password" otherwise. The sidebar on the right displays the "Release notes" for January 2026, which mention the launch of Data Explorer and Gemini 3.

```
# Main Program
user_password = input("Enter your password: ")

if is_strong_password(user_password):
    print("Strong password")
else:
    print("Weak password")
```