AI ASSISTED CODING

LAB_Assignment_10.5.

2303A52197

BATCH - 35

Task Description #1 – Variable Naming Issues

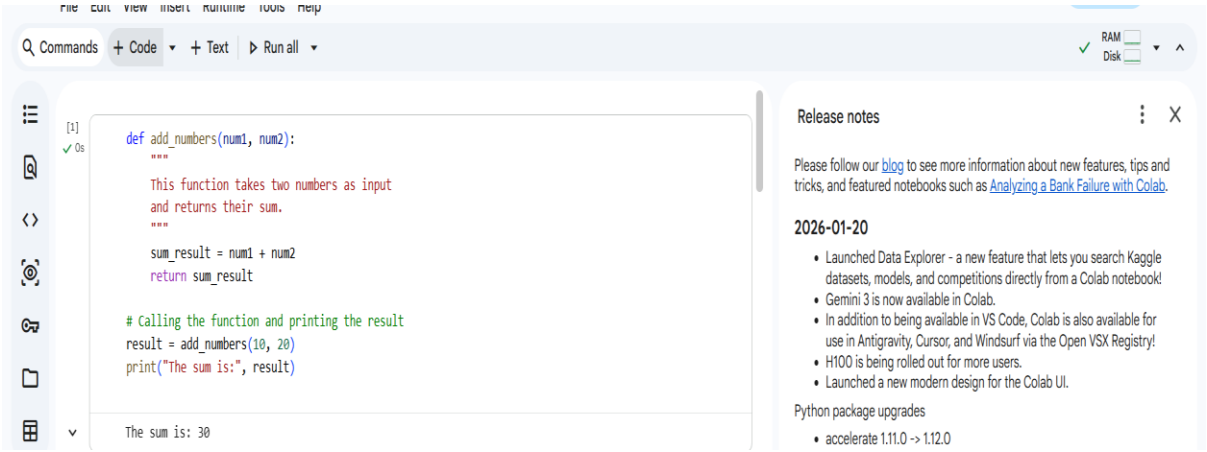
Task: Use AI to improve unclear variable names.

Sample Input Code:

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

Expected Output:

• Code rewritten with meaningful function and variable names.



Task Description #2 – Missing Error Handling

Task: Use AI to add proper error handling.

Sample Input Code:

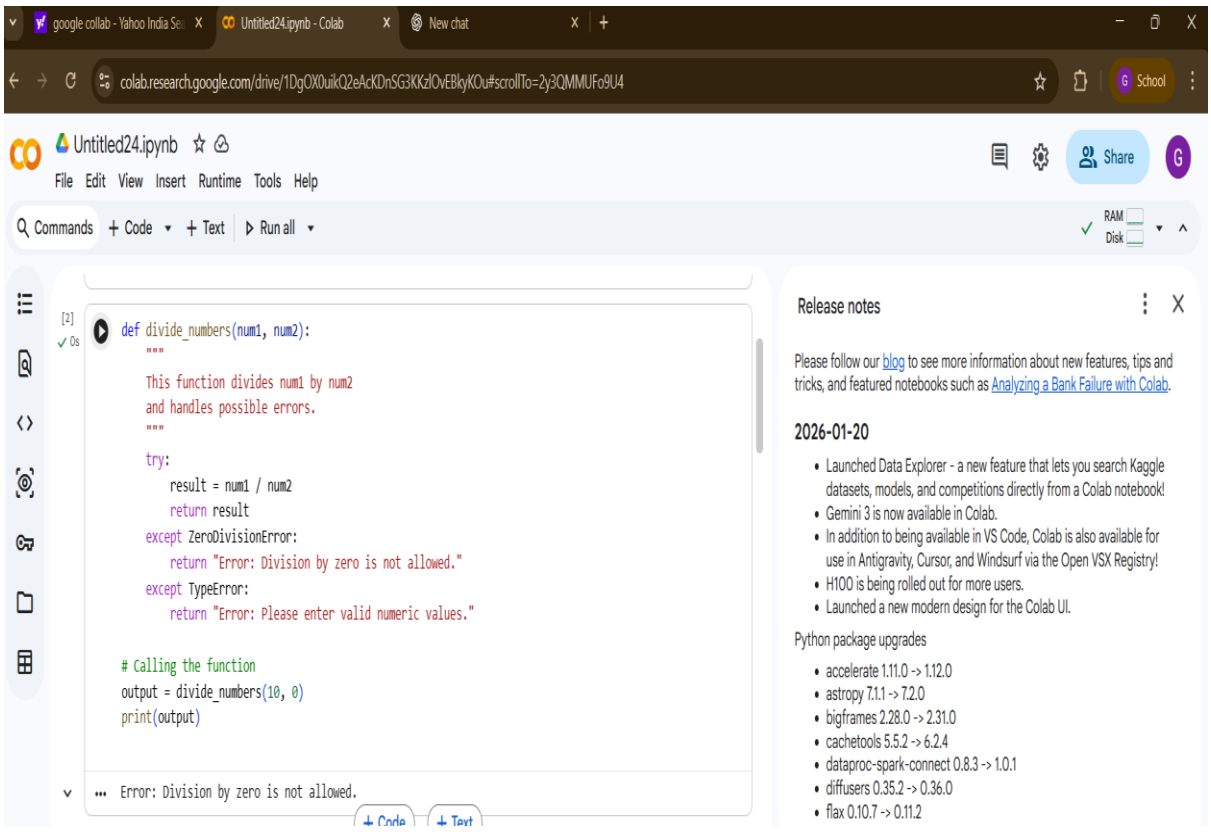
def divide(a, b):

return a / b

print(divide(10, 0))

Expected Output:

• Code with exception handling and clear error messages



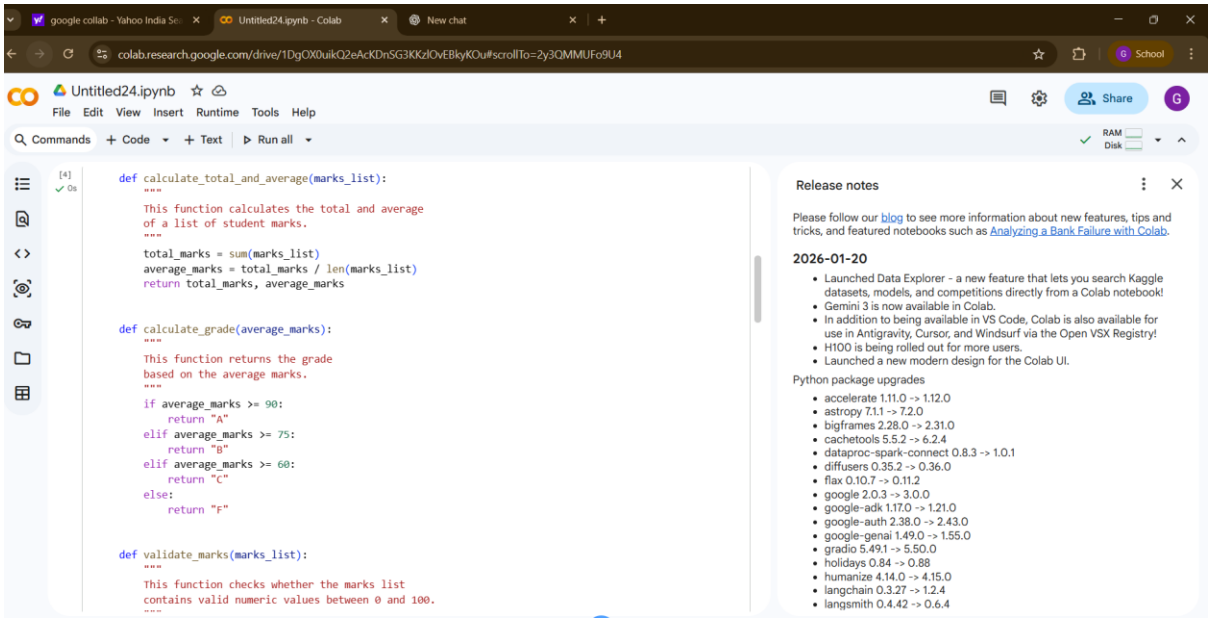
Task Description #3: Student Marks Processing System

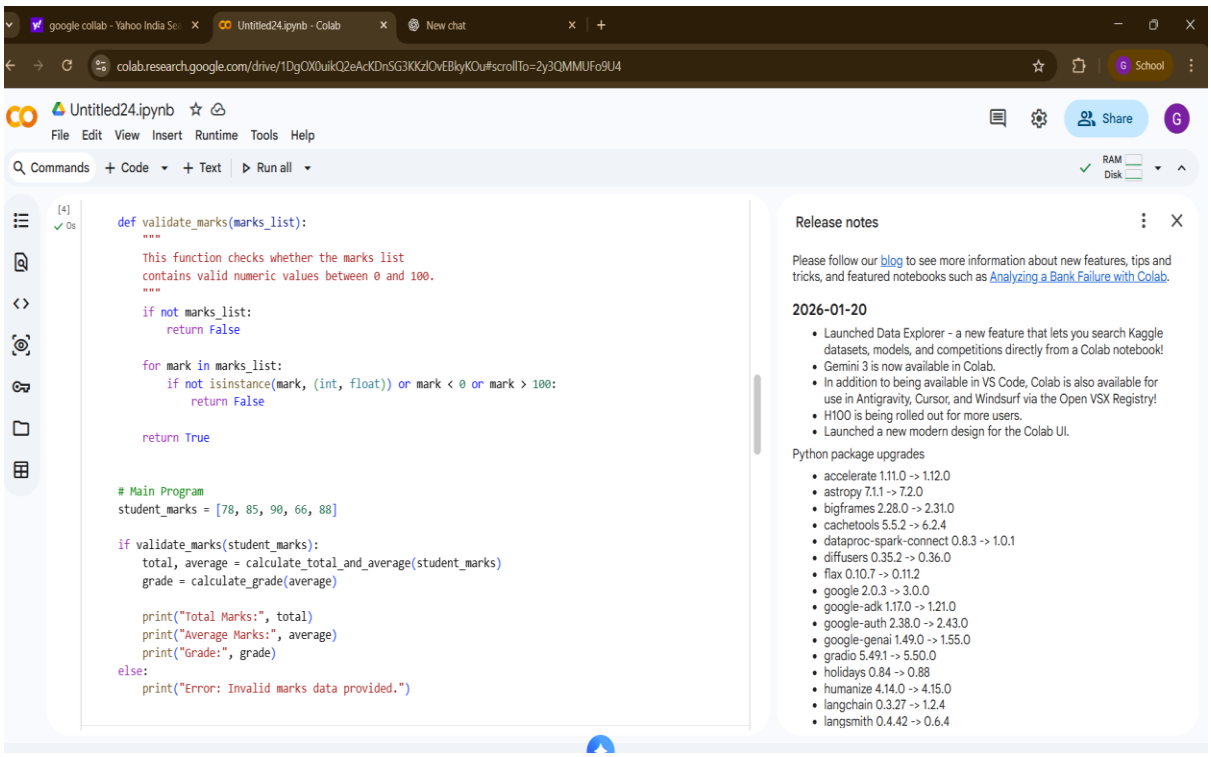
The following program calculates total, average, and grade of a student, but it has poor readability, style issues, and no error handling.

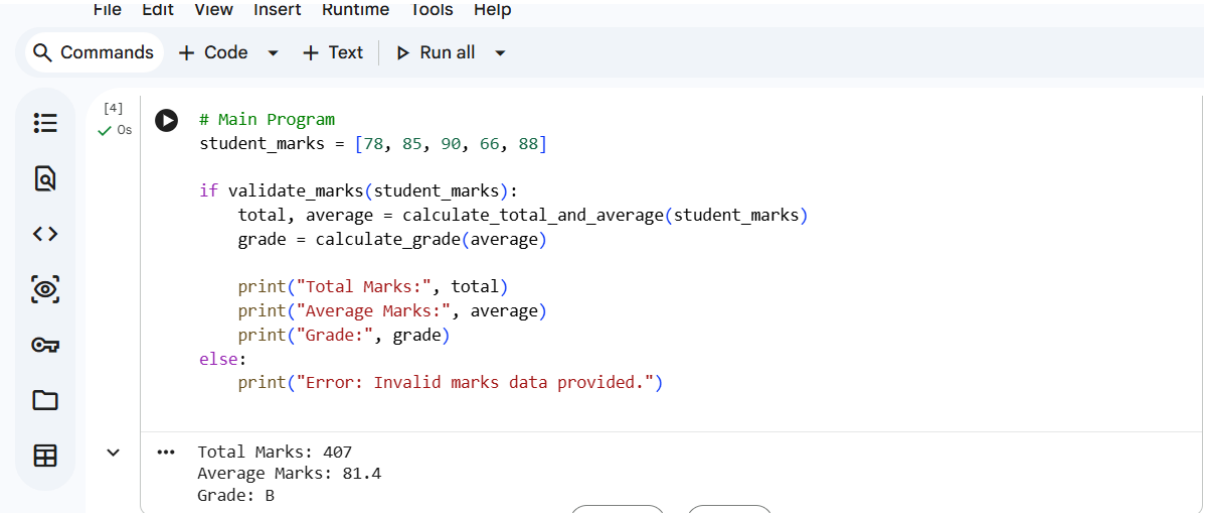
```
marks=[78,85,90,66,88]
t=0
for i in marks:
t=t+i
a=t/len(marks)
if a>=90:
print("A")
elif a>=75:
print("B")
elif a>=60:
print("C")
else:
print("F")
```

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.


google collab - Yahoo India Se
Untitled24.ipynb - Colab
New chat
colab.research.google.com/drive/1DgOX0uikQ2eAcKDnSG3KKzIOvEBkyKOu#scrollTo=2y3QMMUFo9U4
School
Untitled24.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[4]
0s
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
langsmith 0.4.42 -> 0.6.4


google collab - Yahoo India Se
Untitled24.ipynb - Colab
New chat
colab.research.google.com/drive/1DgOX0uikQ2eAcKDnSG3KKzIOvEBkyKOu#scrollTo=2y3QMMUFo9U4
School
Untitled24.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[4]
0s
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
langsmith 0.4.42 -> 0.6.4



Task Description #4: Use AI to add docstrings and inline comments to the following function.

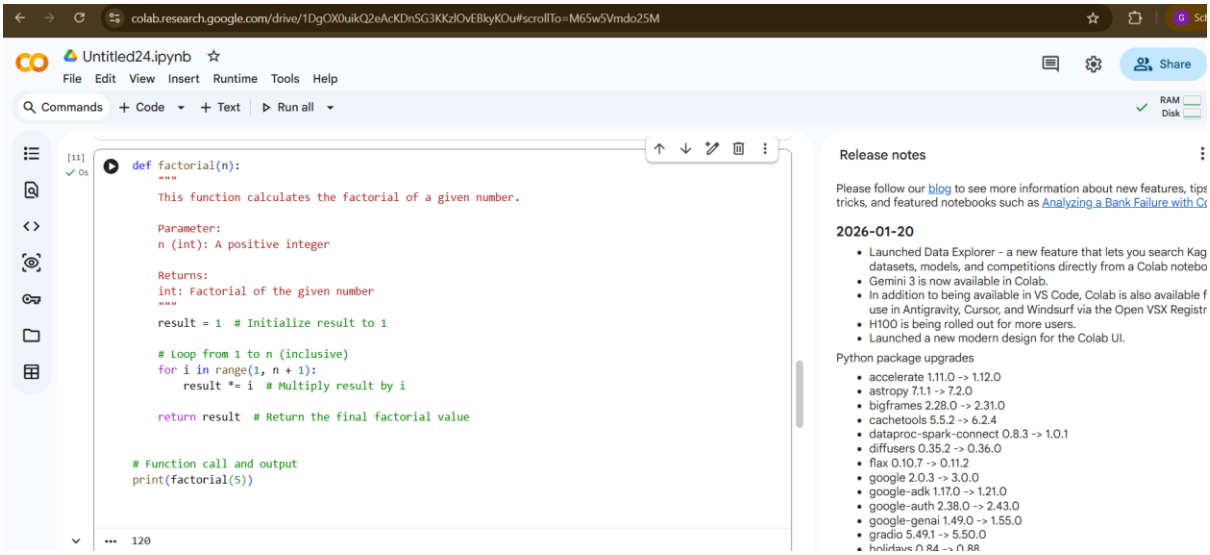
def factorial(n):

result = 1

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

result *= i

return result



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.