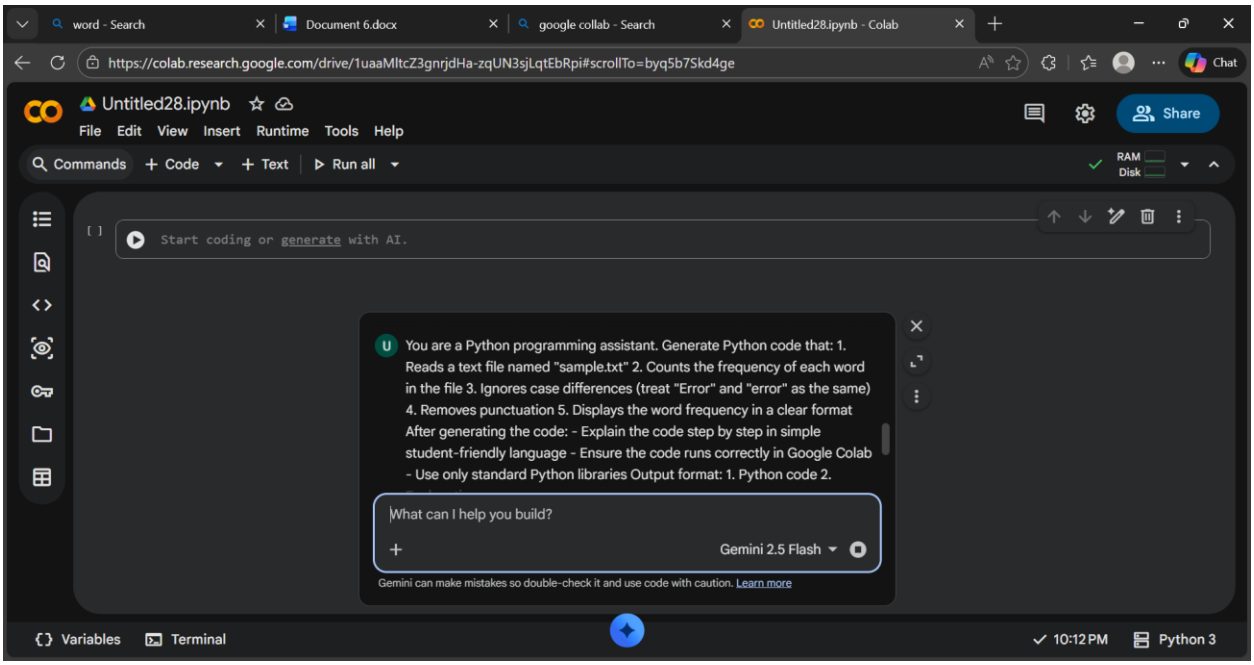
Assignment 2.3 AI ASSISTED CODING

HTNO:2303A51305

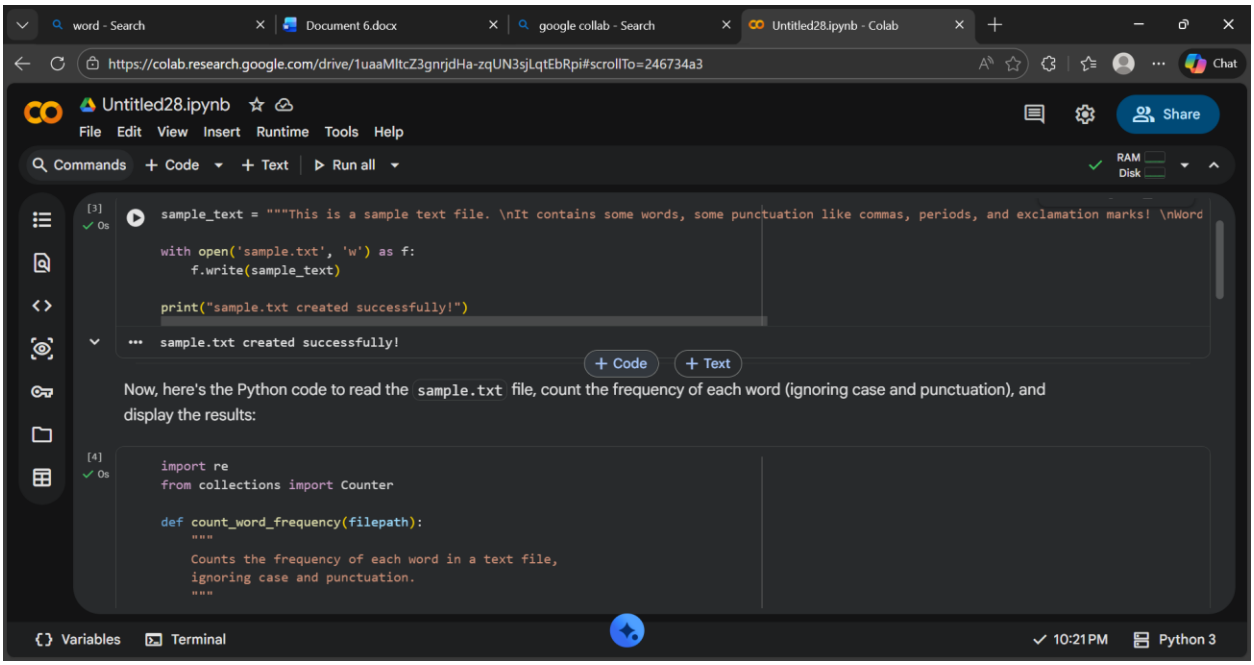
BTNO:05

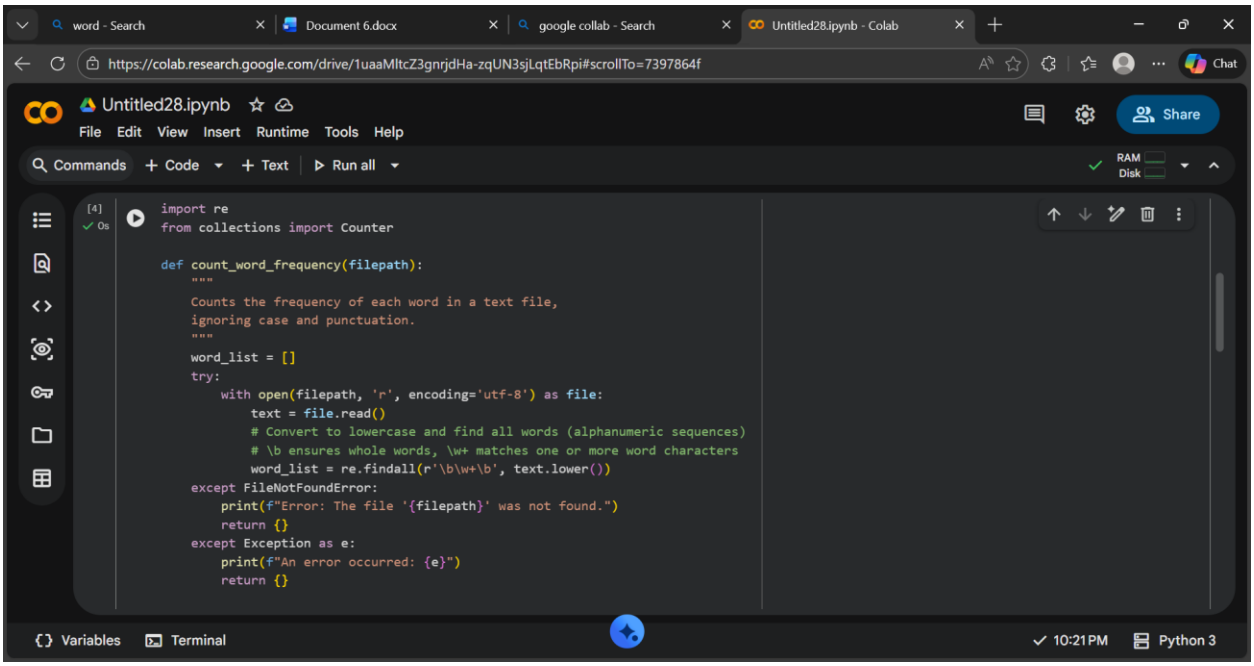
TASK 1:

Prompt:

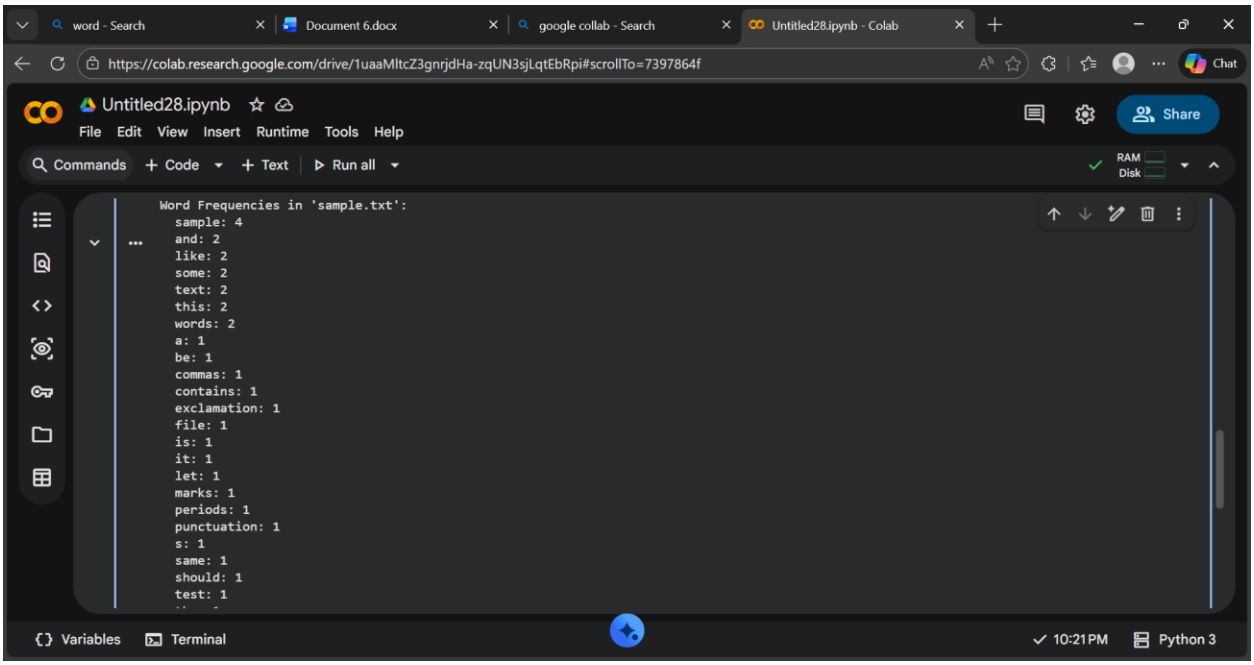


Code:


word - Search
Document 6.docx
google collab - Search
Untitled28.ipynb - Colab
https://colab.research.google.com/drive/1uaaMltcZ3gnrjdHa-zqUN3sjLqtEbRpi#scrollTo=246734a3
Chat
Untitled28.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[3] 0s
sample_text = """This is a sample text file. \nIt contains some words, some punctuation like commas, periods, and exclamation marks! \nWord
with open('sample.txt', 'w') as f:
f.write(sample_text)
print("sample.txt created successfully!")
sample.txt created successfully!
+ Code + Text
Now, here's the Python code to read the sample.txt file, count the frequency of each word (ignoring case and punctuation), and display the results:
[4] 0s
import re
from collections import Counter
def count_word_frequency(filepath):
"""
Counts the frequency of each word in a text file,
ignoring case and punctuation.
"""
Variables Terminal
10:21 PM
Python 3


word - Search
Document 6.docx
google collab - Search
Untitled28.ipynb - Colab
https://colab.research.google.com/drive/1uaaMltcZ3gnrjdHa-zqUN3sjLqtEbRpi#scrollTo=7397864f
Chat
Untitled28.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[4] 0s
import re
from collections import Counter
def count_word_frequency(filepath):
"""
Counts the frequency of each word in a text file,
ignoring case and punctuation.
"""
word_list = []
try:
with open(filepath, 'r', encoding='utf-8') as file:
text = file.read()
# Convert to lowercase and find all words (alphanumeric sequences)
# \b ensures whole words, \w+ matches one or more word characters
word_list = re.findall(r'\b\w+\b', text.lower())
except FileNotFoundError:
print(f"Error: The file '{filepath}' was not found.")
return {}
except Exception as e:
print(f"An error occurred: {e}")
return {}
Variables Terminal
10:21 PM
Python 3

Output:



Explanation:

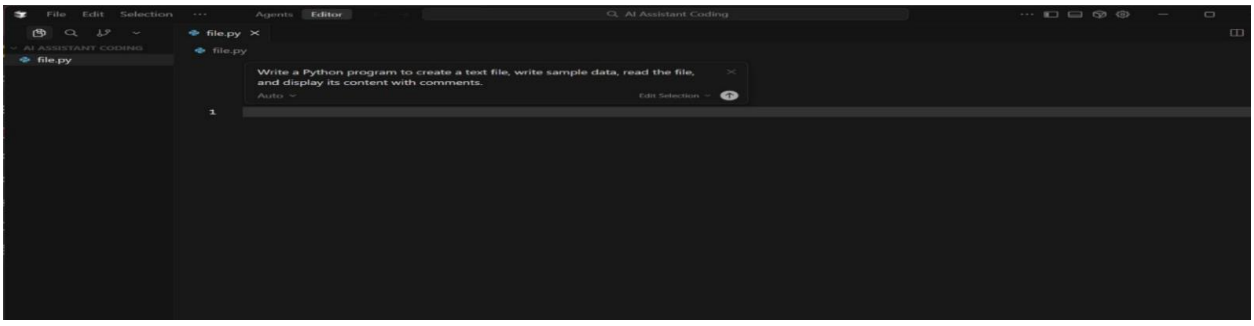
The program reads text from a file and converts all words to lowercase to avoid case differences.

It removes punctuation and splits the text into words, then counts how many times each word appears.

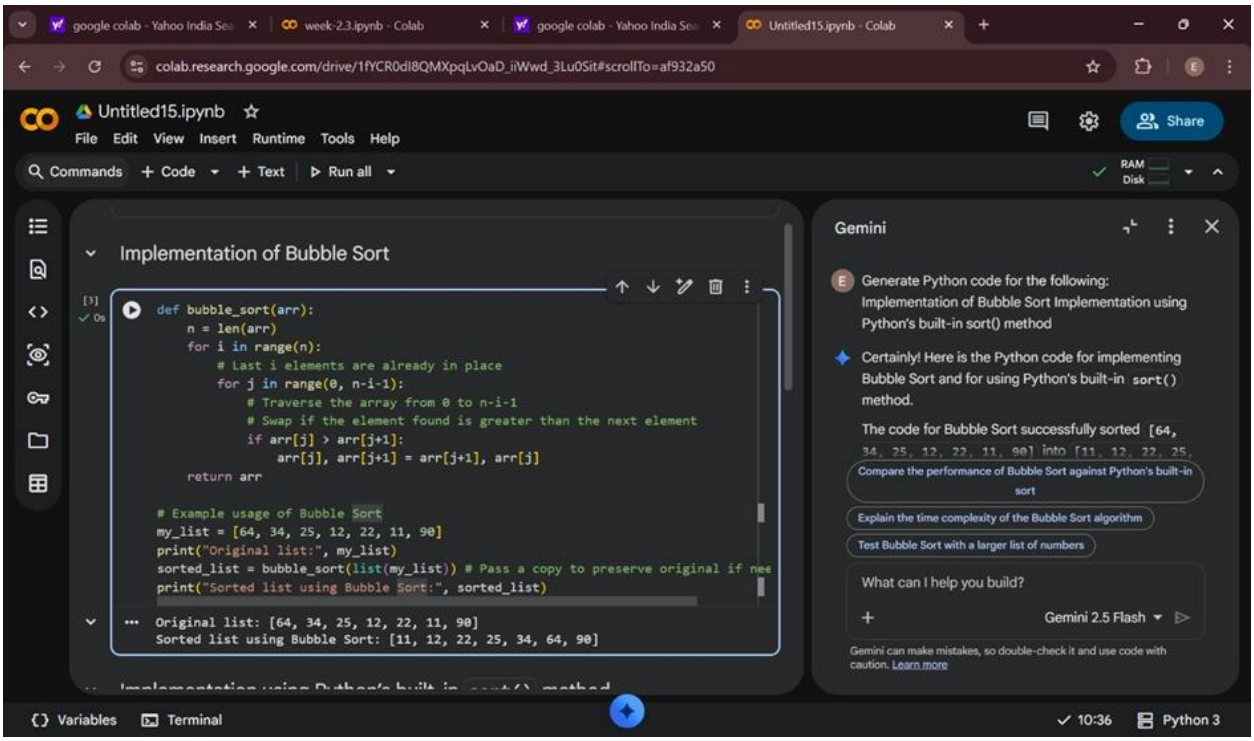
Finally, it prints each word along with its frequency in the file.

Task 2:

Prompt:



Code:



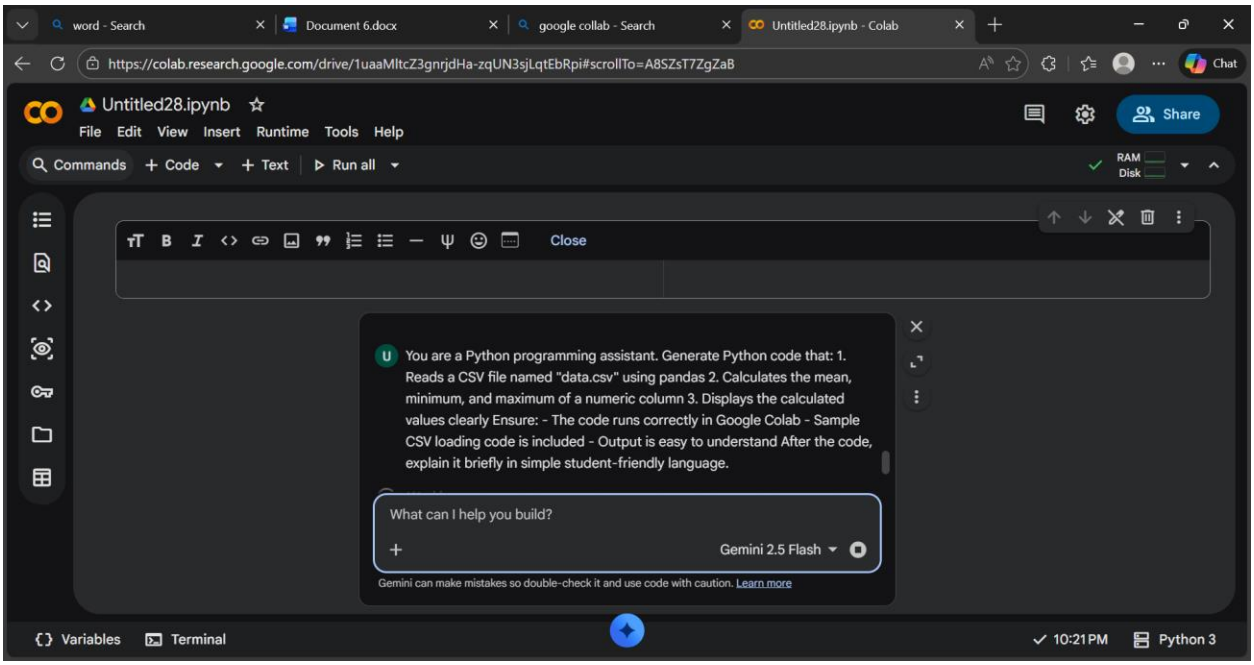
Output:

Explanation: The program creates a text file and writes sample text into it using file handling methods.

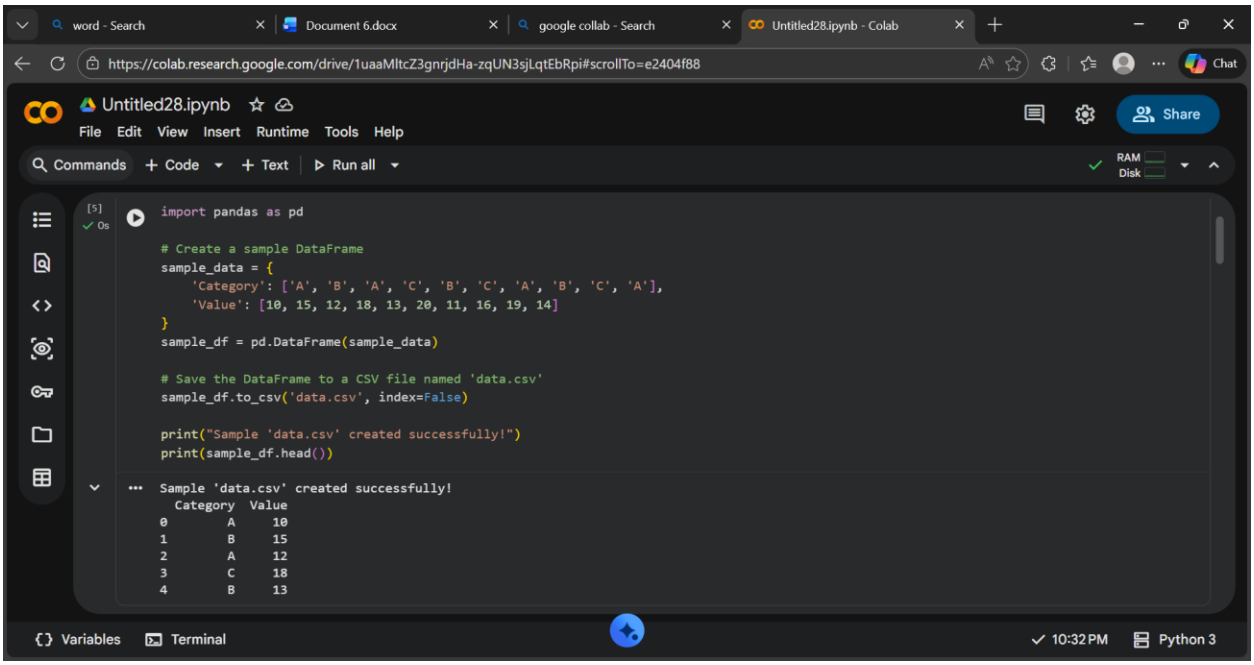
It then reads the file content and displays it on the screen.

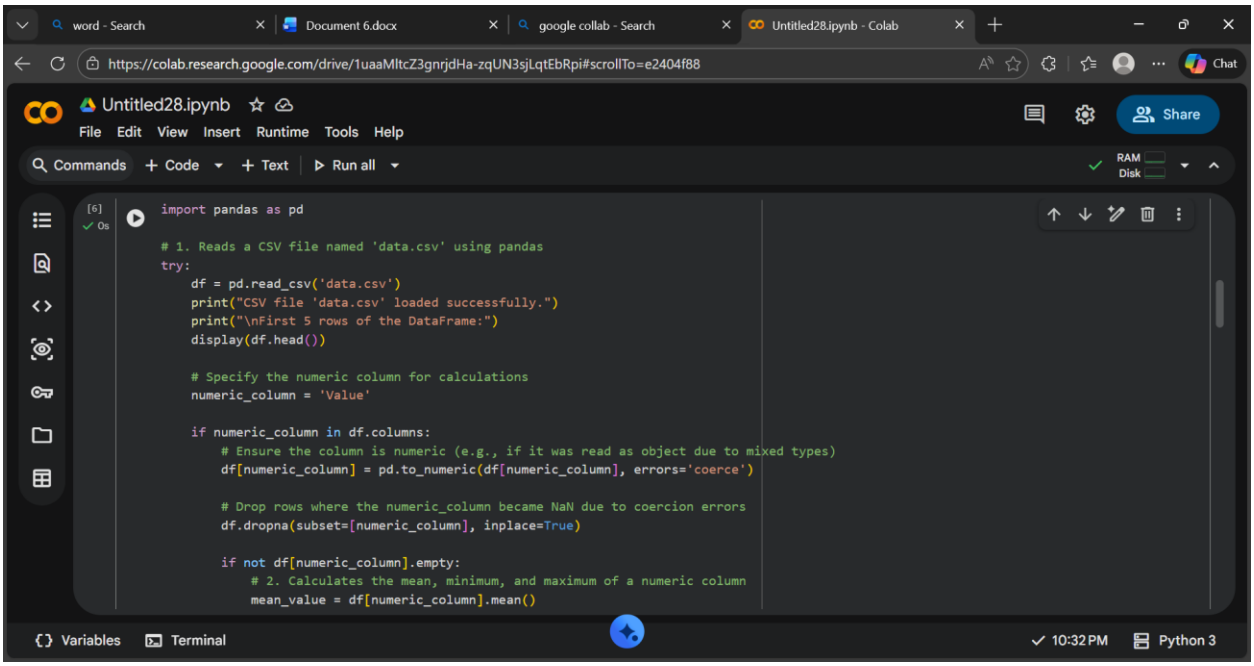
Task 3: CSV Data Analysis

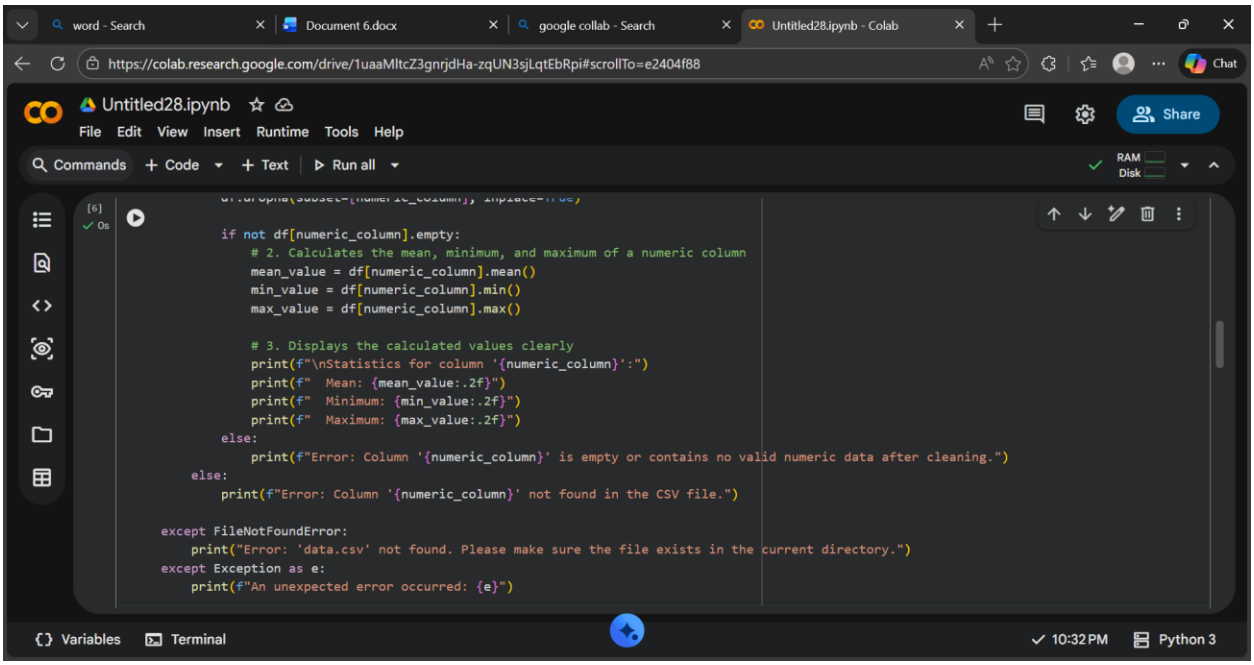
Prompt:


word - Search
Document 6.docx
google collab - Search
Untitled28.ipynb - Colab
https://colab.research.google.com/drive/1uaaMltcZ3gnrjdHa-zqUN3sjLqtEbRpi#scrollTo=A8SZsT7ZgZaB
Chat
Untitled28.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
Close
You are a Python programming assistant. Generate Python code that: 1. Reads a CSV file named "data.csv" using pandas 2. Calculates the mean, minimum, and maximum of a numeric column 3. Displays the calculated values clearly Ensure: - The code runs correctly in Google Colab - Sample CSV loading code is included - Output is easy to understand After the code, explain it briefly in simple student-friendly language.
What can I help you build?
Gemini 2.5 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
10:21 PM
Python 3

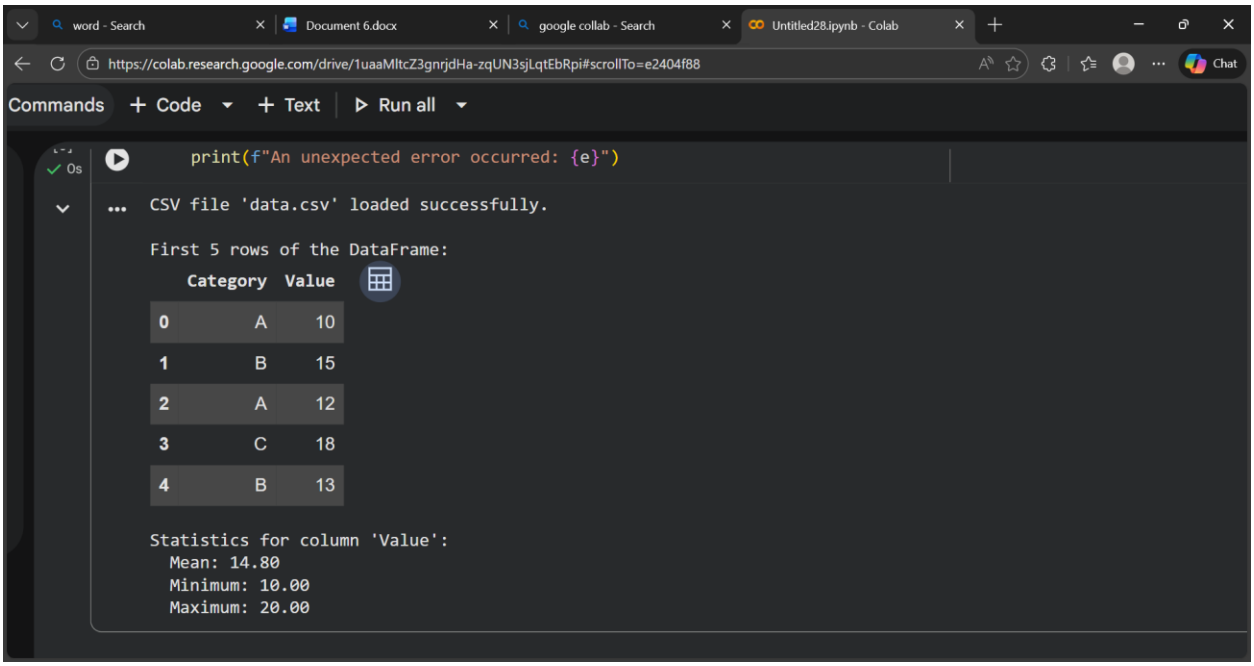
Code:


word - Search
Document 6.docx
google collab - Search
Untitled28.ipynb - Colab
https://colab.research.google.com/drive/1uaaMltcZ3gnrjdHa-zqUN3sjLqtEbRpi#scrollTo=e2404f88
Chat
Untitled28.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[5]
0s
import pandas as pd
# Create a sample DataFrame
sample_data = {
'Category': ['A', 'B', 'A', 'C', 'B', 'C', 'A', 'B', 'C', 'A'],
'Value': [10, 15, 12, 18, 13, 20, 11, 16, 19, 14]
}
sample_df = pd.DataFrame(sample_data)
# Save the DataFrame to a CSV file named 'data.csv'
sample_df.to_csv('data.csv', index=False)
print("Sample 'data.csv' created successfully!")
print(sample_df.head())
Sample 'data.csv' created successfully!
Category Value
0 A 10
1 B 15
2 A 12
3 C 18
4 B 13
Variables
Terminal
10:32 PM
Python 3


word - Search
Document 6.docx
google collab - Search
Untitled28.ipynb - Colab
https://colab.research.google.com/drive/1uaaMltcZ3gnrjdHa-zqUN3sjLqtEbRpi#scrollTo=e2404f88
Chat
Untitled28.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[6]
0s
import pandas as pd
# 1. Reads a CSV file named 'data.csv' using pandas
try:
df = pd.read_csv('data.csv')
print("CSV file 'data.csv' loaded successfully.")
print("\nFirst 5 rows of the DataFrame:")
display(df.head())
# Specify the numeric column for calculations
numeric_column = 'Value'
if numeric_column in df.columns:
# Ensure the column is numeric (e.g., if it was read as object due to mixed types)
df[numeric_column] = pd.to_numeric(df[numeric_column], errors='coerce')
# Drop rows where the numeric_column became NaN due to coercion errors
df.dropna(subset=[numeric_column], inplace=True)
if not df[numeric_column].empty:
# 2. Calculates the mean, minimum, and maximum of a numeric column
mean_value = df[numeric_column].mean()
Variables
Terminal
10:32 PM
Python 3



Output:

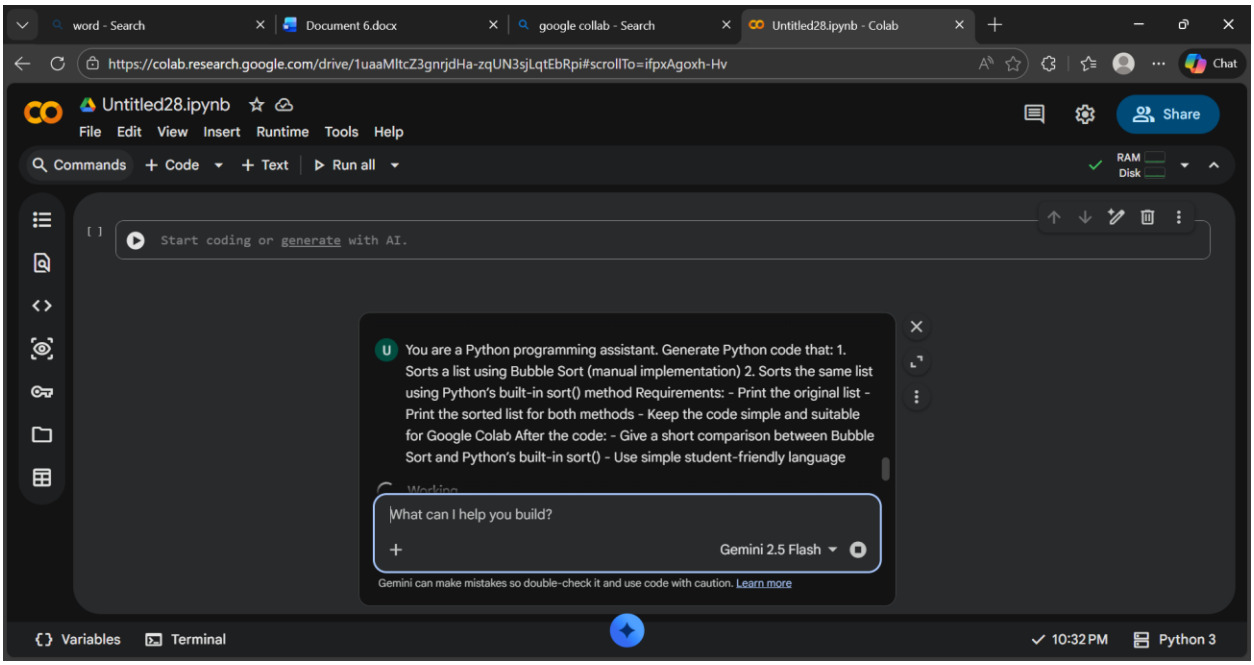


Explanation: Manual sorting uses loops and comparisons to arrange elements step by step.

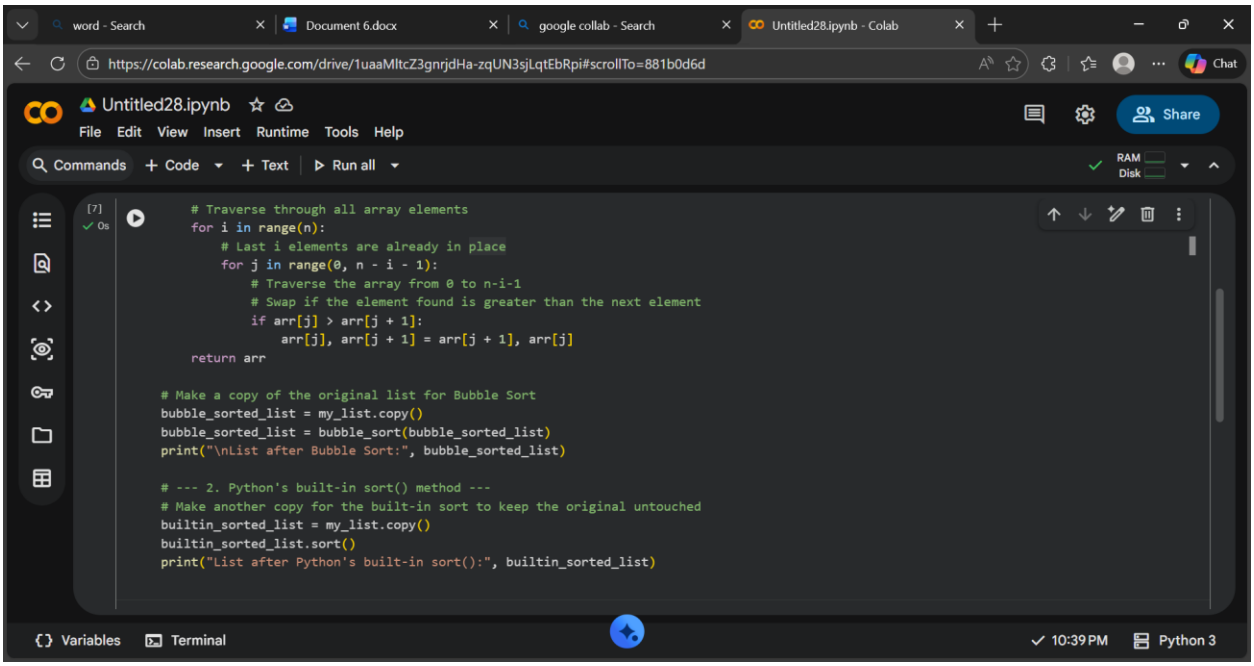
Built-in sorting is faster and simpler as Python handles the sorting internally.

Task 4: Sorting Lists – Manual vs Built-in

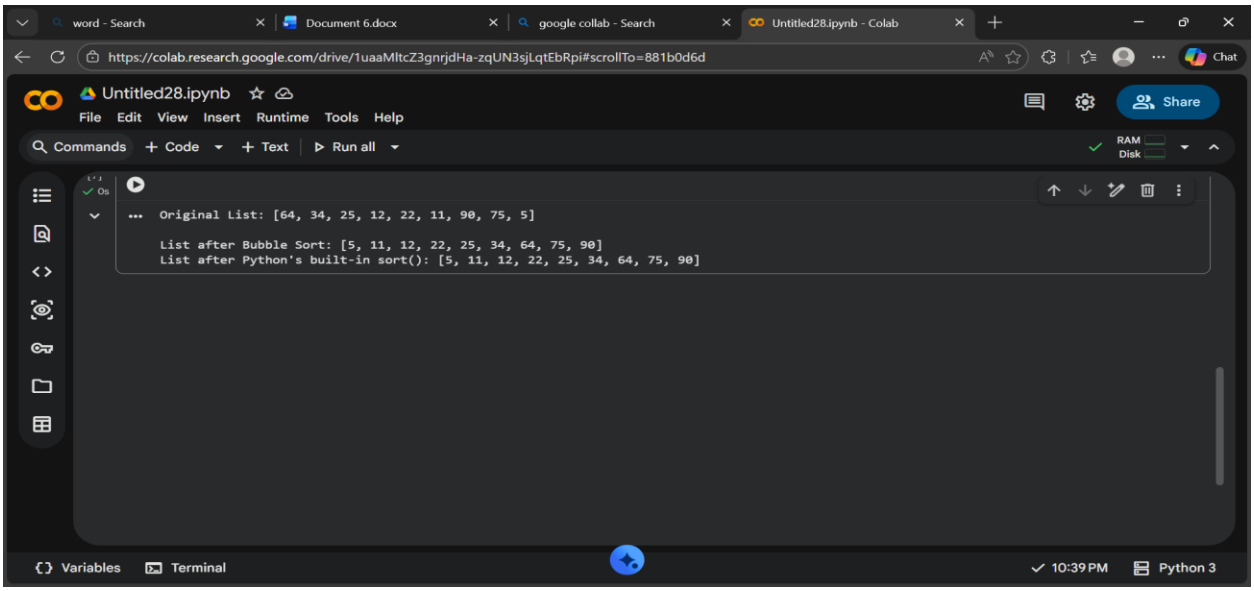
Prompt:


word - Search
Document 6.docx
google collab - Search
Untitled28.ipynb - Colab
https://colab.research.google.com/drive/1uaaMltcZ3gnrjdHa-zqUN3sjLqtEbRpi#scrollTo=ifpxAgoxh-Hv
Chat
Untitled28.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[ ]
Start coding or generate with AI.
You are a Python programming assistant. Generate Python code that: 1. Sorts a list using Bubble Sort (manual implementation) 2. Sorts the same list using Python's built-in sort() method Requirements: - Print the original list - Print the sorted list for both methods - Keep the code simple and suitable for Google Colab After the code: - Give a short comparison between Bubble Sort and Python's built-in sort() - Use simple student-friendly language
Working...
What can I help you build?
Gemini 2.5 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
10:32 PM
Python 3

Code:


word - Search
Document 6.docx
google collab - Search
Untitled28.ipynb - Colab
https://colab.research.google.com/drive/1uaaMltcZ3gnrjdHa-zqUN3sjLqtEbRpi#scrollTo=881b0d6d
Chat
Untitled28.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[7]
0s
# Traverse through all array elements
for i in range(n):
# Last i elements are already in place
for j in range(0, n - i - 1):
# Traverse the array from 0 to n-i-1
# Swap if the element found is greater than the next element
if arr[j] > arr[j + 1]:
arr[j], arr[j + 1] = arr[j + 1], arr[j]
return arr
# Make a copy of the original list for Bubble Sort
bubble_sorted_list = my_list.copy()
bubble_sorted_list = bubble_sort(bubble_sorted_list)
print("\nList after Bubble Sort:", bubble_sorted_list)
# --- 2. Python's built-in sort() method ---
# Make another copy for the built-in sort to keep the original untouched
builtin_sorted_list = my_list.copy()
builtin_sorted_list.sort()
print("List after Python's built-in sort():", builtin_sorted_list)
Variables
Terminal
10:39 PM
Python 3

Output:



Expalantion: Bubble sort repeatedly compares and swaps elements and is slow for large lists.

Python’s built-in sort() is faster and more efficient because it uses optimized algorithms.