AI ASSITED CODING

NAME:M.KEERTHANA **BATCH:11-CSE**

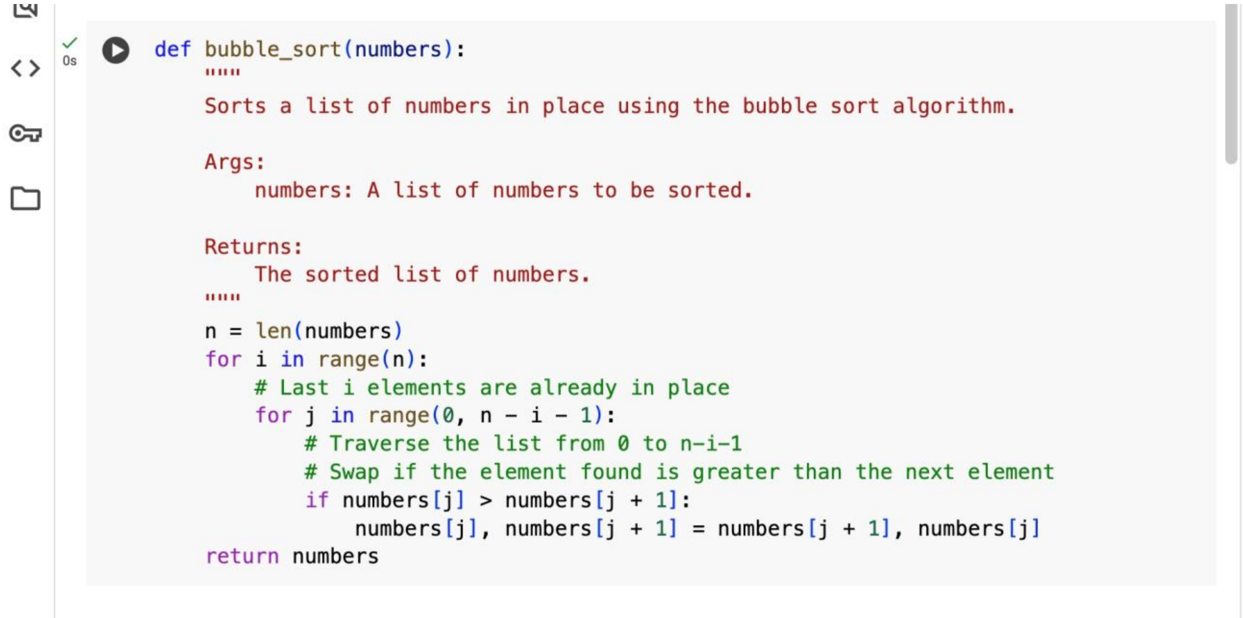
HTNO:2403A51259

TASK1:

generate a python code that performs sorting of a list using both the bubble sort algorithm and pythons built -in-sort() function .compare the two implementations

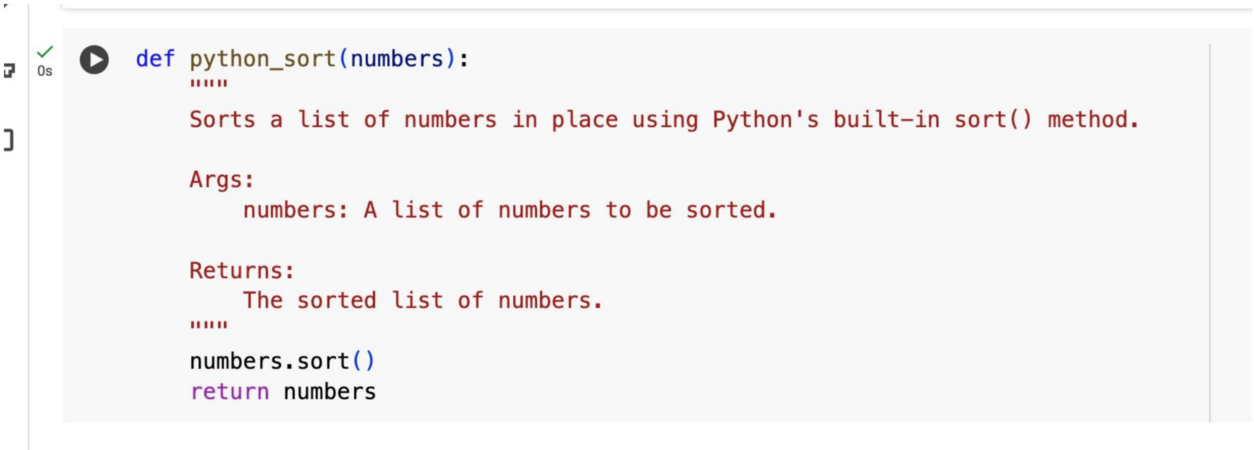
CODE:

Bubble sort:



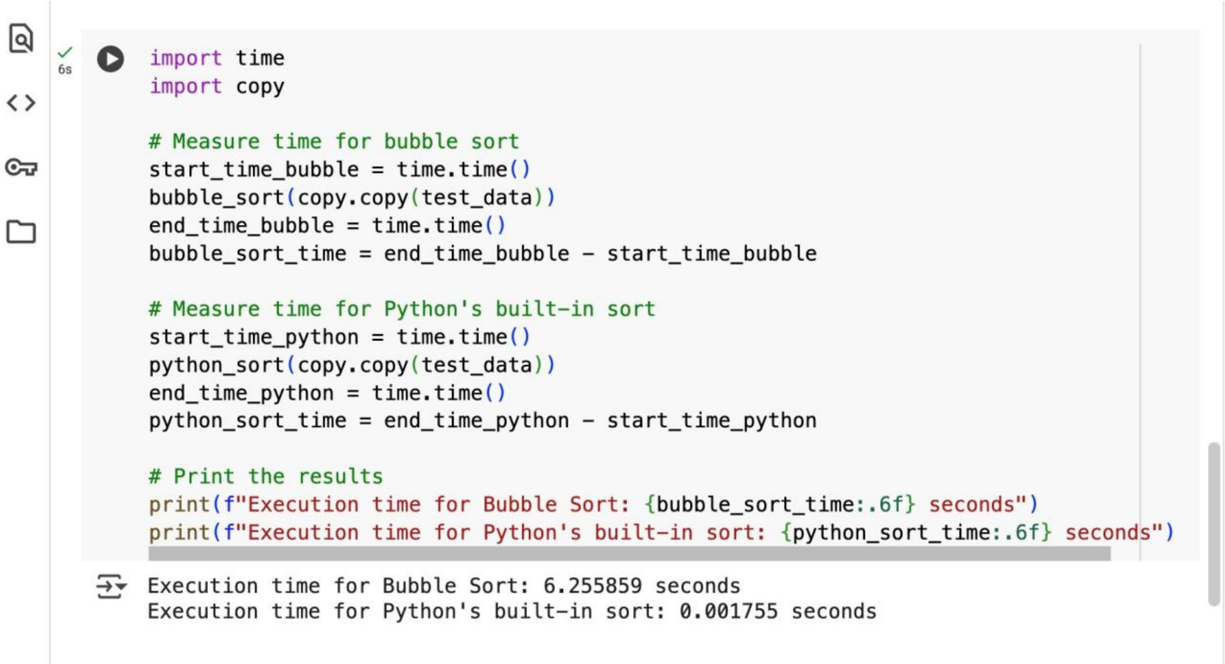
```python
def bubble_sort(numbers):
    """
    Sorts a list of numbers in place using the bubble sort algorithm.

    Args:
        numbers: A list of numbers to be sorted.

    Returns:
        The sorted list of numbers.
    """
    n = len(numbers)
    for i in range(n):
        # Last i elements are already in place
        for j in range(0, n - i - 1):
            # Traverse the list from 0 to n-i-1
            # Swap if the element found is greater than the next element
            if numbers[j] > numbers[j + 1]:
                numbers[j], numbers[j + 1] = numbers[j + 1], numbers[j]
    return numbers
```

Python’s built-in-sort:



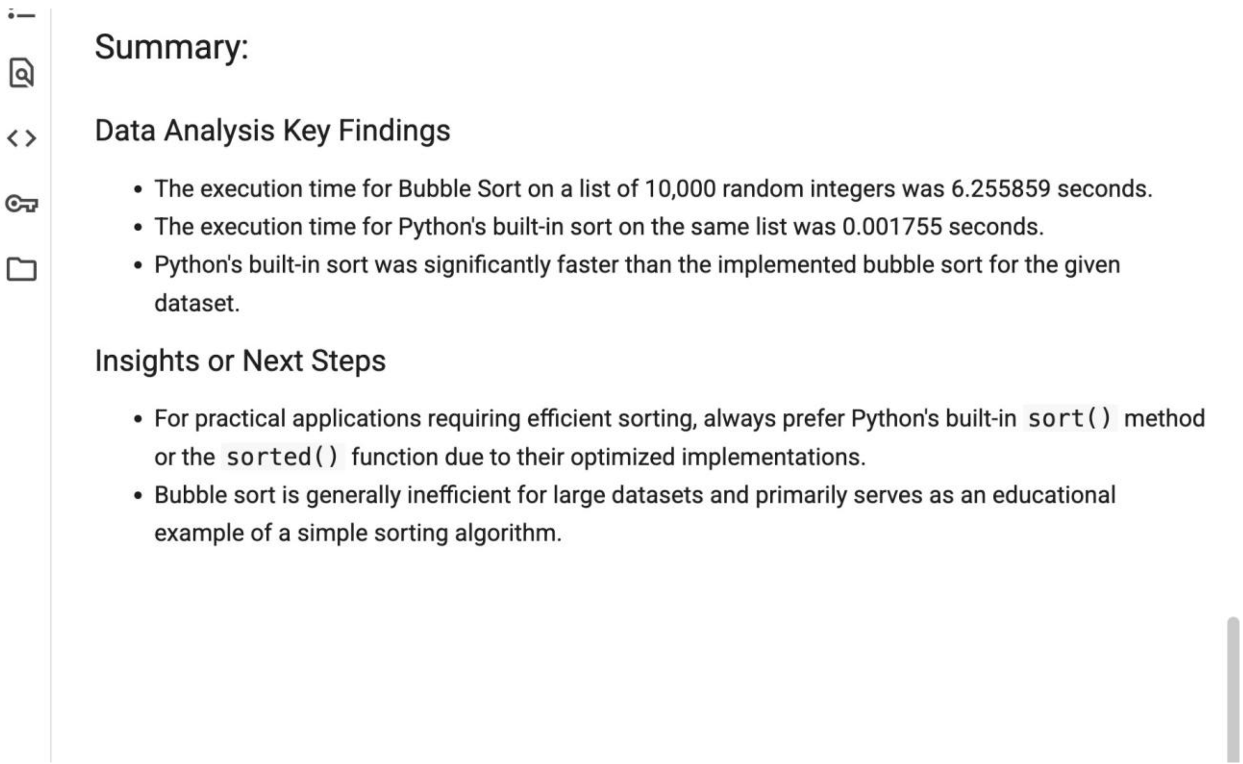
```python
def python_sort(numbers):
    """
    Sorts a list of numbers in place using Python's built-in sort() method.

    Args:
        numbers: A list of numbers to be sorted.

    Returns:
        The sorted list of numbers.
    """
    numbers.sort()
    return numbers
```

Comparison:



```python
import time
import copy

# Measure time for bubble sort
start_time_bubble = time.time()
bubble_sort(copy.copy(test_data))
end_time_bubble = time.time()
bubble_sort_time = end_time_bubble - start_time_bubble

# Measure time for Python's built-in sort
start_time_python = time.time()
python_sort(copy.copy(test_data))
end_time_python = time.time()
python_sort_time = end_time_python - start_time_python

# Print the results
print(f"Execution time for Bubble Sort: {bubble_sort_time:.6f} seconds")
print(f"Execution time for Python's built-in sort: {python_sort_time:.6f} seconds")
```

```
Execution time for Bubble Sort: 6.255859 seconds
Execution time for Python's built-in sort: 0.001755 seconds
```



Summary:

Data Analysis Key Findings

- The execution time for Bubble Sort on a list of 10,000 random integers was 6.255859 seconds.
- The execution time for Python's built-in sort on the same list was 0.001755 seconds.
- Python's built-in sort was significantly faster than the implemented bubble sort for the given dataset.

Insights or Next Steps

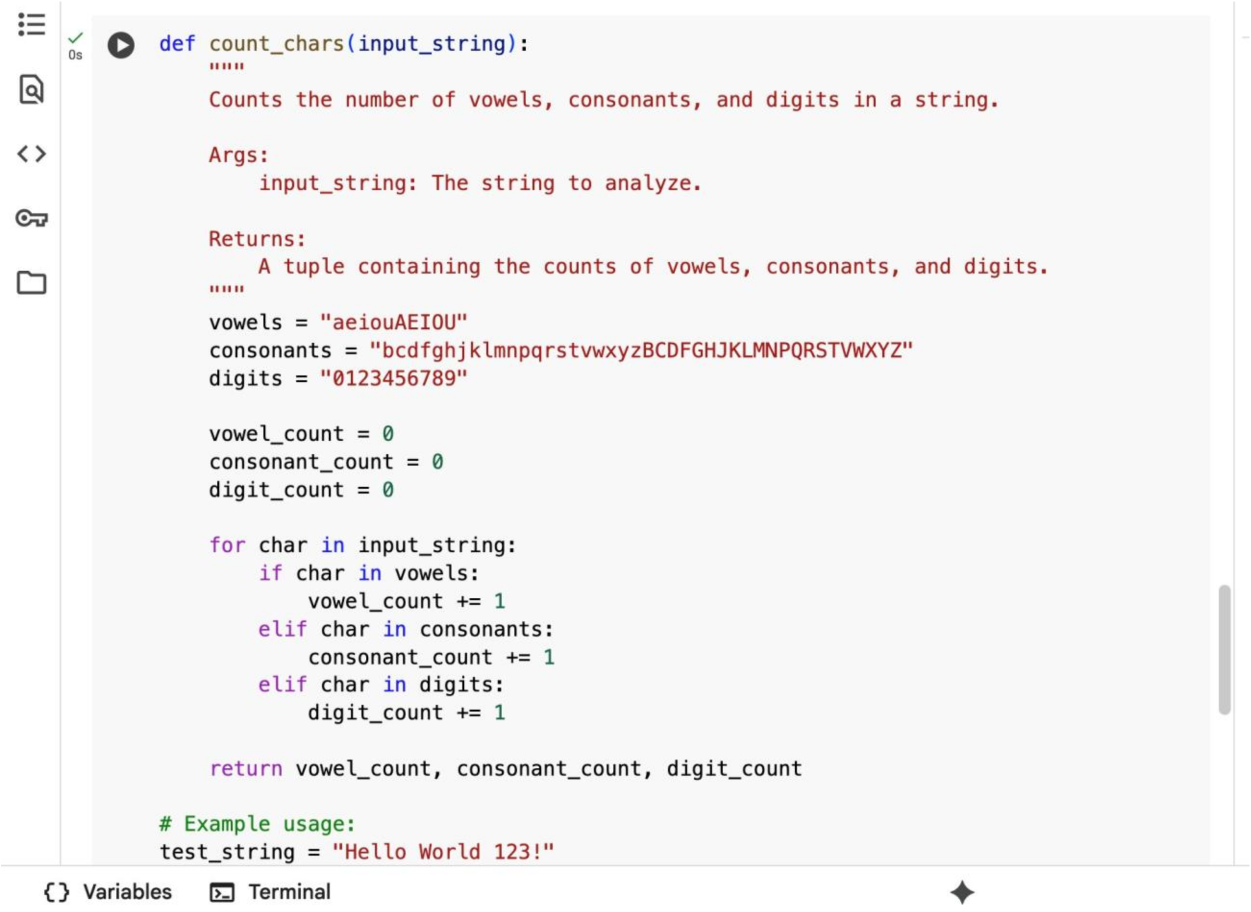
- For practical applications requiring efficient sorting, always prefer Python's built-in `sort()` method or the `sorted()` function due to their optimized implementations.
- Bubble sort is generally inefficient for large datasets and primarily serves as an educational example of a simple sorting algorithm.

TASK 2:

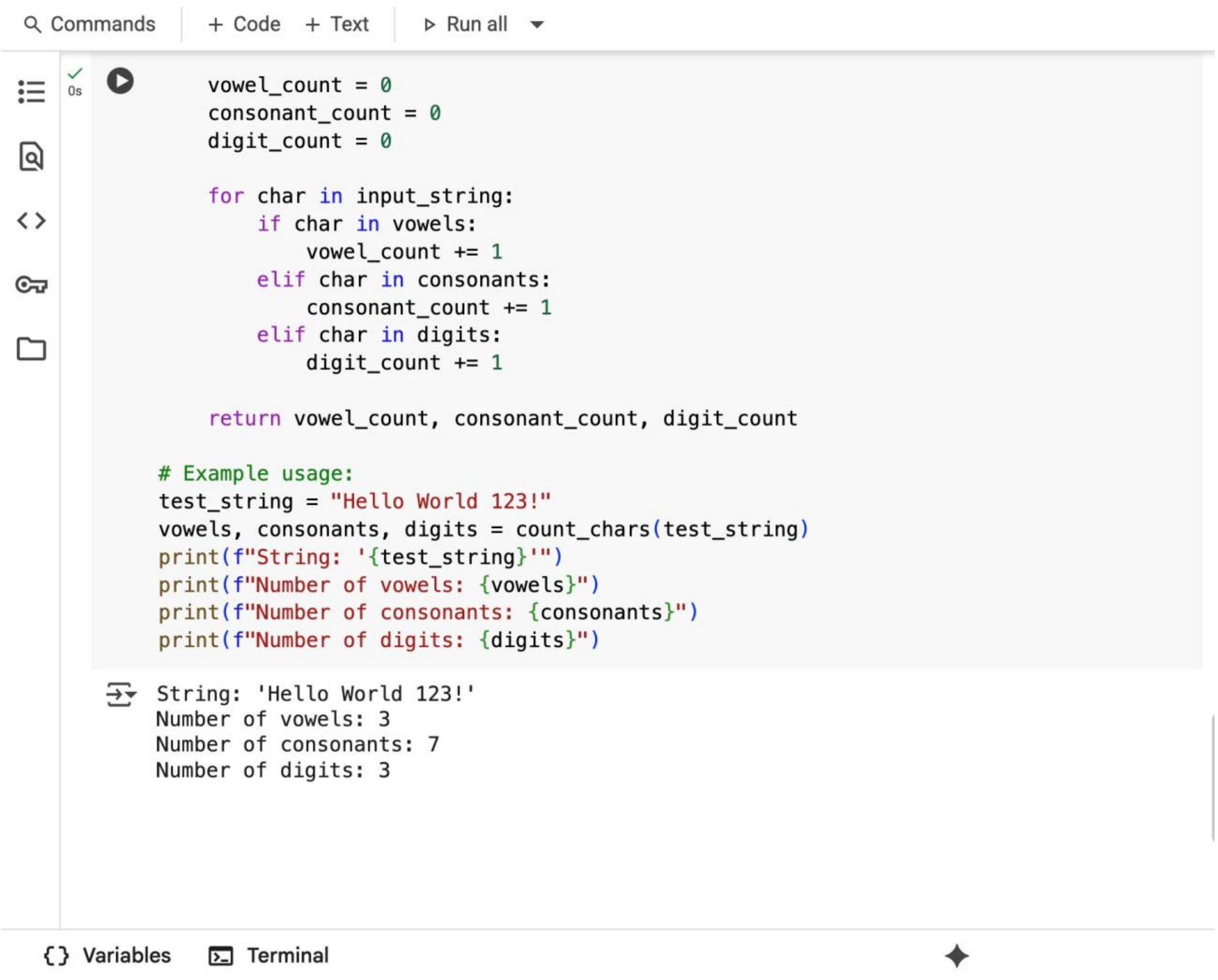
Generate a python function that takes a string and returns:

the number of vowels the number of consonants the number of digits in the string

CODE & OUTPUT:



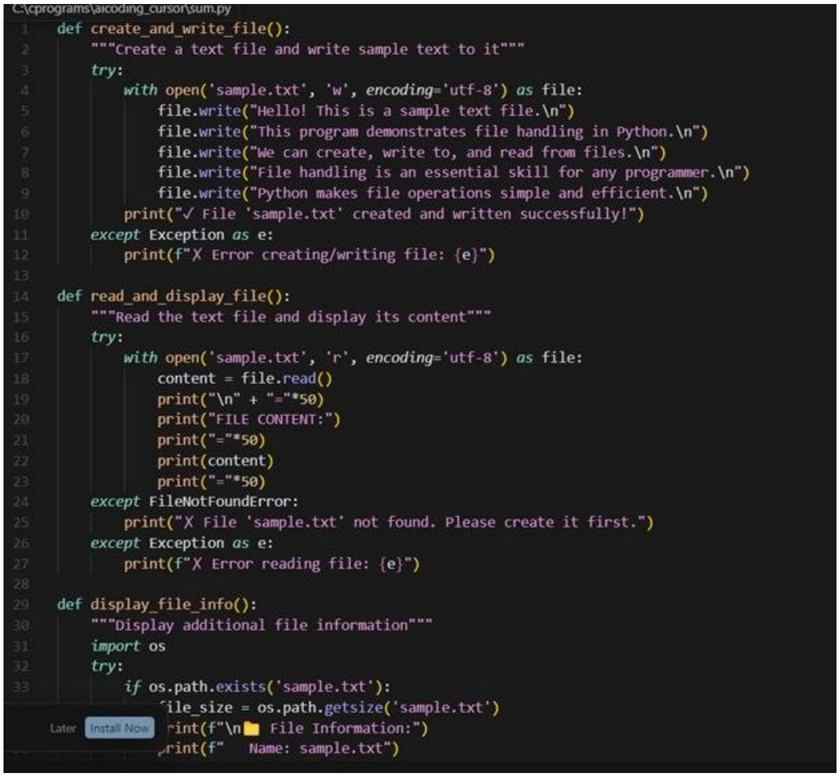
```python
def count_chars(input_string):
    """
    Counts the number of vowels, consonants, and digits in a string.

    Args:
        input_string: The string to analyze.

    Returns:
        A tuple containing the counts of vowels, consonants, and digits.
    """
    vowels = "aeiouAEIOU"
    consonants = "bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ"
    digits = "0123456789"

    vowel_count = 0
    consonant_count = 0
    digit_count = 0

    for char in input_string:
        if char in vowels:
            vowel_count += 1
        elif char in consonants:
            consonant_count += 1
        elif char in digits:
            digit_count += 1

    return vowel_count, consonant_count, digit_count

# Example usage:
test_string = "Hello World 123!"
```



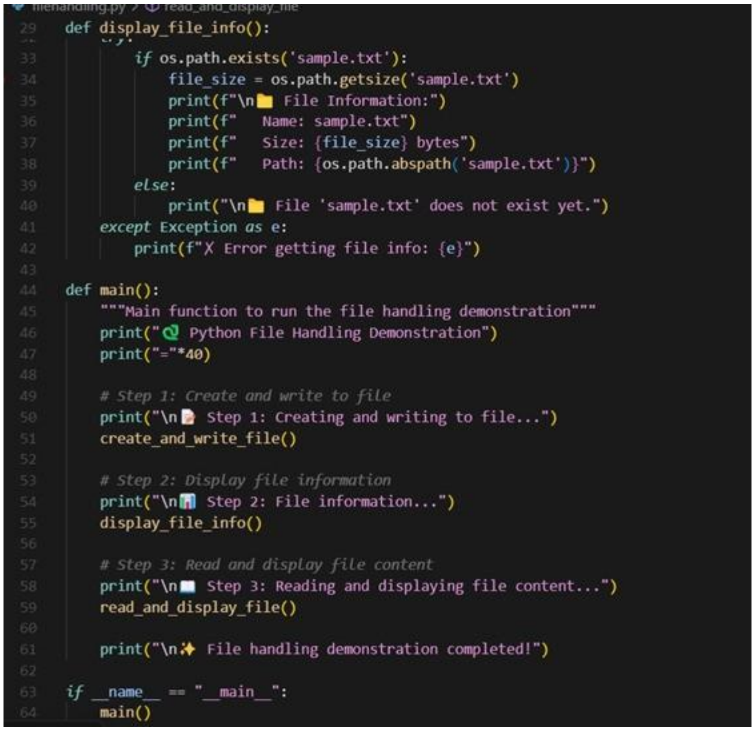
```
    vowel_count = 0
    consonant_count = 0
    digit_count = 0

    for char in input_string:
        if char in vowels:
            vowel_count += 1
        elif char in consonants:
            consonant_count += 1
        elif char in digits:
            digit_count += 1

    return vowel_count, consonant_count, digit_count

# Example usage:
test_string = "Hello World 123!"
vowels, consonants, digits = count_chars(test_string)
print(f"String: '{test_string}'")
print(f"Number of vowels: {vowels}")
print(f"Number of consonants: {consonants}")
print(f"Number of digits: {digits}")
```

```
String: 'Hello World 123!'
Number of vowels: 3
Number of consonants: 7
Number of digits: 3
```

TASK 3:

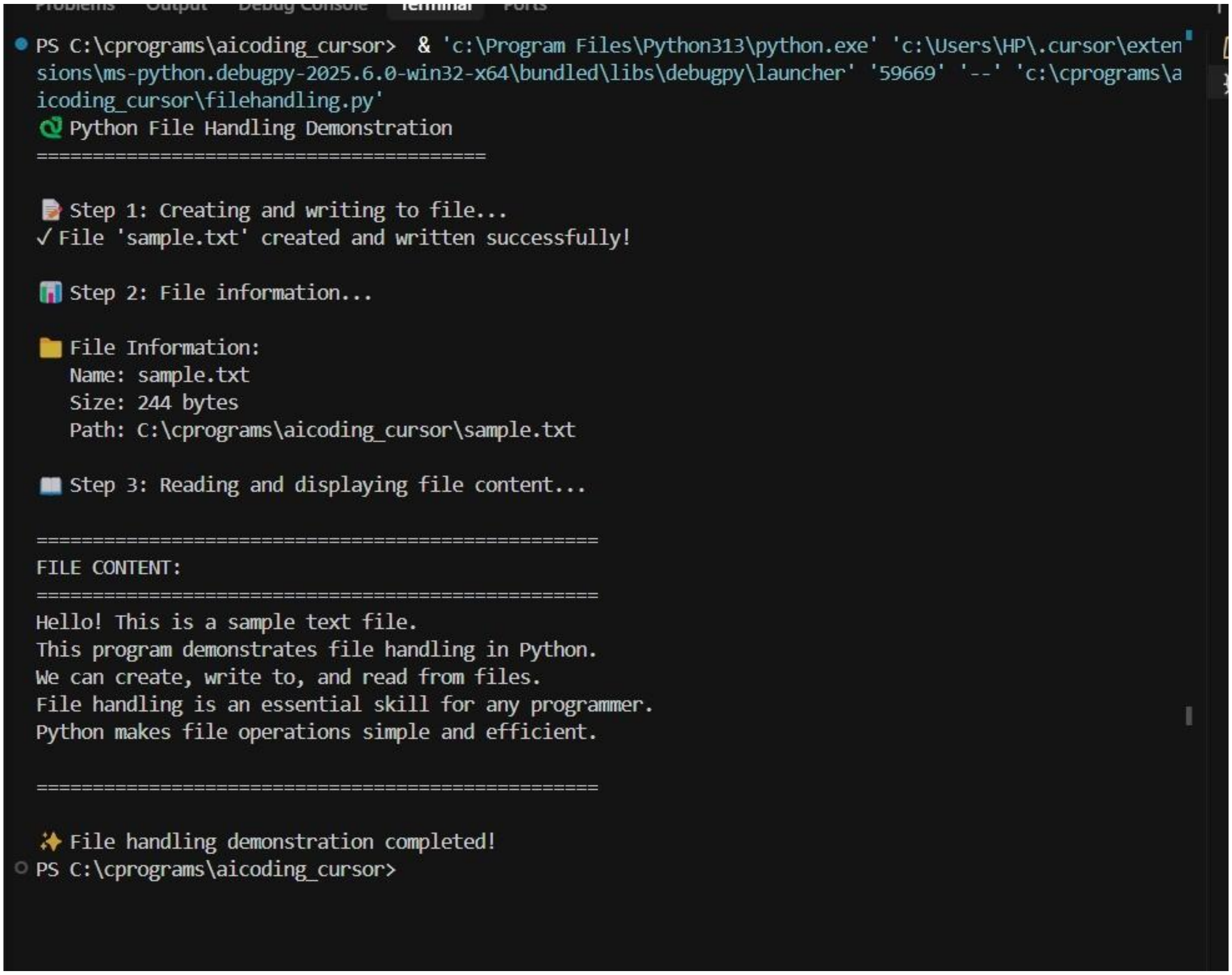


```python
def create_and_write_file():
    """Create a text file and write sample text to it"""
    try:
        with open('sample.txt', 'w', encoding='utf-8') as file:
            file.write("Hello! This is a sample text file.\n")
            file.write("This program demonstrates file handling in Python.\n")
            file.write("We can create, write to, and read from files.\n")
            file.write("File handling is an essential skill for any programmer.\n")
            file.write("Python makes file operations simple and efficient.\n")
        print("✓ File 'sample.txt' created and written successfully!")
    except Exception as e:
        print(f"X Error creating/writing file: {e}")

def read_and_display_file():
    """Read the text file and display its content"""
    try:
        with open('sample.txt', 'r', encoding='utf-8') as file:
            content = file.read()
            print("\n" + "="*50)
            print("FILE CONTENT:")
            print("="*50)
            print(content)
            print("="*50)
    except FileNotFoundError:
        print("X File 'sample.txt' not found. Please create it first.")
    except Exception as e:
        print(f"X Error reading file: {e}")

def display_file_info():
    """Display additional file information"""
    import os
    try:
        if os.path.exists('sample.txt'):
            ile_size = os.path.getsize('sample.txt')
            rint(f"\n📁 File Information:")
            rint(f"   Name: sample.txt")
```



```python
def display_file_info():
        if os.path.exists('sample.txt'):
            file_size = os.path.getsize('sample.txt')
            print(f"\n📁 File Information:")
            print(f"   Name: sample.txt")
            print(f"   Size: {file_size} bytes")
            print(f"   Path: {os.path.abspath('sample.txt')}")
        else:
            print("\n📁 File 'sample.txt' does not exist yet.")
    except Exception as e:
        print(f"X Error getting file info: {e}")

def main():
    """Main function to run the file handling demonstration"""
    print("🐍 Python File Handling Demonstration")
    print("="*40)

    # Step 1: Create and write to file
    print("\n📝 Step 1: Creating and writing to file...")
    create_and_write_file()

    # Step 2: Display file information
    print("\n📊 Step 2: File information...")
    display_file_info()

    # Step 3: Read and display file content
    print("\n📖 Step 3: Reading and displaying file content...")
    read_and_display_file()

    print("\n✨ File handling demonstration completed!")

if __name__ == "__main__":
    main()
```

output:



```
PS C:\cprograms\aicoding_cursor>  & 'c:\Program Files\Python313\python.exe' 'c:\Users\HP\.cursor\exten
sions\ms-python.debugpy-2025.6.0-win32-x64\bundled\libs\debugpy\launcher' '59669' '--' 'c:\cprograms\a
icoding_cursor\filehandling.py'
🐍 Python File Handling Demonstration
========================================

📝 Step 1: Creating and writing to file...
✓ File 'sample.txt' created and written successfully!

📊 Step 2: File information...

📁 File Information:
   Name: sample.txt
   Size: 244 bytes
   Path: C:\cprograms\aicoding_cursor\sample.txt

📖 Step 3: Reading and displaying file content...

==================================================
FILE CONTENT:
==================================================
Hello! This is a sample text file.
This program demonstrates file handling in Python.
We can create, write to, and read from files.
File handling is an essential skill for any programmer.
Python makes file operations simple and efficient.

==================================================

✨ File handling demonstration completed!
PS C:\cprograms\aicoding_cursor>
```

TASK 4:

generate a python program that implements a simple calculator using functions(add,subtarct,multiply,divide).and also explain how the code works

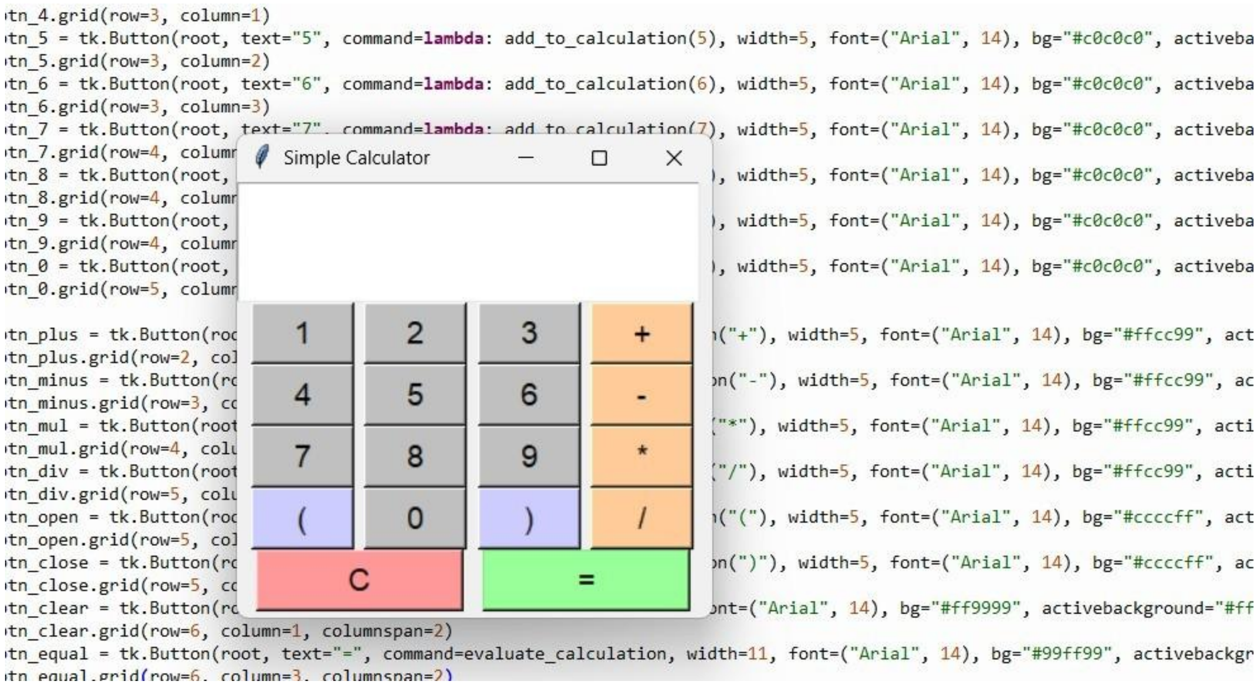
code:

```
import tkinter as tk def
add_to_calculation(symbol):
global calculation calculation
```

```
+=                    str(symbol)
text_result.delete(1.0, "end")
    text_result.insert(1.0, calculation)

def evaluate_calculation():
    global      calculation
try:
        calculation = str(eval(calculation))
text_result.delete(1.0, "end")
text_result.insert(1.0, calculation)    except:
        clear_field()
        text_result.insert(1.0, "Error")

def  clear_field():          global
calculation      calculation = ""
text_result.delete(1.0,     "end")
text_result.insert(1.0, "")

root            =             tk.Tk()
root.geometry("300x275")
root.title("Simple Calculator")
root.configure(bg="#f0f0f0") # Light grey background

calculation = ""

text_result = tk.Text(root, height=2, width=16, font=("Arial", 24),
bg="#ffffff", fg="#333333") # White background, dark text
text_result.grid(columnspan=5)

btn_1 = tk.Button(root, text="1", command=lambda:
add_to_calculation(1), width=5, font=("Arial", 14), bg="#c0c0c0",
activebackground="#a0a0a0") # Grey button
btn_1.grid(row=2,       column=1)       btn_2       =
tk.Button(root, text="2", command=lambda:
add_to_calculation(2), width=5, font=("Arial", 14), bg="#c0c0c0",
activebackground="#a0a0a0") btn_2.grid(row=2, column=2)
btn_3 = tk.Button(root, text="3", command=lambda:
add_to_calculation(3), width=5, font=("Arial", 14), bg="#c0c0c0",
activebackground="#a0a0a0") btn_3.grid(row=2,
column=3) btn_4 = tk.Button(root, text="4",
command=lambda:
add_to_calculation(4), width=5, font=("Arial", 14), bg="#c0c0c0",
activebackground="#a0a0a0") btn_4.grid(row=3,
column=1) btn_5 = tk.Button(root, text="5",
command=lambda:
add_to_calculation(5), width=5, font=("Arial", 14), bg="#c0c0c0",
activebackground="#a0a0a0") btn_5.grid(row=3,
column=2) btn_6 = tk.Button(root, text="6",
command=lambda:
add_to_calculation(6), width=5, font=("Arial", 14), bg="#c0c0c0",
activebackground="#a0a0a0") btn_6.grid(row=3,
column=3) btn_7 = tk.Button(root, text="7",
command=lambda:
add_to_calculation(7), width=5, font=("Arial", 14), bg="#c0c0c0",
activebackground="#a0c0c0") btn_7.grid(row=4,
column=1) btn_8 = tk.Button(root, text="8",
command=lambda:
add_to_calculation(8), width=5, font=("Arial", 14), bg="#c0c0c0",
activebackground="#a0c0c0") btn_8.grid(row=4, column=2)
btn_9 = tk.Button(root, text="9", command=lambda:
add_to_calculation(9), width=5, font=("Arial", 14), bg="#c0c0c0",
activebackground="#a0c0c0") btn_9.grid(row=4,
column=3) btn_0 = tk.Button(root, text="0",
command=lambda:
add_to_calculation(0), width=5, font=("Arial", 14), bg="#c0c0c0",
activebackground="#a0c0c0")
btn_0.grid(row=5, column=2)

btn_plus = tk.Button(root, text="+", command=lambda:
```

```
add_to_calculation("+"), width=5, font=("Arial", 14), bg="#ffcc99",
activebackground="#ffb366") # Orange button
btn_plus.grid(row=2, column=4) btn_minus = tk.Button(root,
text="-", command=lambda:
add_to_calculation("-"), width=5, font=("Arial", 14), bg="#ffcc99",
activebackground="#ffb366") btn_minus.grid(row=3, column=4)
btn_mul = tk.Button(root, text="*", command=lambda:
add_to_calculation("*"), width=5, font=("Arial", 14), bg="#ffcc99",
activebackground="#ffb366") btn_mul.grid(row=4, column=4)
btn_div = tk.Button(root, text="/", command=lambda:
add_to_calculation("/"), width=5, font=("Arial", 14), bg="#ffcc99",
activebackground="#ffb366") btn_div.grid(row=5, column=4)
btn_open = tk.Button(root, text="(", command=lambda:
add_to_calculation("("), width=5, font=("Arial", 14), bg="#ccccff",
activebackground="#b3b3ff") # Light blue button
btn_open.grid(row=5, column=1) btn_close = tk.Button(root,
text=")", command=lambda:
add_to_calculation(")"), width=5, font=("Arial", 14), bg="#ccccff",
activebackground="#b3b3ff") btn_close.grid(row=5, column=3)
btn_clear = tk.Button(root, text="C", command=clear_field, width=11,
font=("Arial", 14), bg="#ff9999", activebackground="#ff6666") # Red
button
btn_clear.grid(row=6, column=1, columnspan=2)
btn_equal = tk.Button(root, text="=", command=evaluate_calculation,
width=11, font=("Arial", 14), bg="#99ff99",
activebackground="#66ff66") # Green button
btn_equal.grid(row=6, column=3, columnspan=2)
```

output:



```
tn_4.grid(row=3, column=1)
tn_5 = tk.Button(root, text="5", command=lambda: add_to_calculation(5), width=5, font=("Arial", 14), bg="#c0c0c0", activeba
tn_5.grid(row=3, column=2)
tn_6 = tk.Button(root, text="6", command=lambda: add_to_calculation(6), width=5, font=("Arial", 14), bg="#c0c0c0", activeba
tn_6.grid(row=3, column=3)
tn_7 = tk.Button(root, text="7", command=lambda: add_to_calculation(7), width=5, font=("Arial", 14), bg="#c0c0c0", activeba
tn_7.grid(row=4, columr
tn_8 = tk.Button(root,                                          ), width=5, font=("Arial", 14), bg="#c0c0c0", activeba
tn_8.grid(row=4, columr
tn_9 = tk.Button(root,                                          ), width=5, font=("Arial", 14), bg="#c0c0c0", activeba
tn_9.grid(row=4, columr
tn_0 = tk.Button(root,                                          ), width=5, font=("Arial", 14), bg="#c0c0c0", activeba
tn_0.grid(row=5, columr

tn_plus = tk.Button(roc                                         ("+"), width=5, font=("Arial", 14), bg="#ffcc99", act
tn_plus.grid(row=2, col
tn_minus = tk.Button(rc                                         n("-"), width=5, font=("Arial", 14), bg="#ffcc99", ac
tn_minus.grid(row=3, cc
tn_mul = tk.Button(root                                         ("*"), width=5, font=("Arial", 14), bg="#ffcc99", acti
tn_mul.grid(row=4, colu
tn_div = tk.Button(root                                         ("/"), width=5, font=("Arial", 14), bg="#ffcc99", acti
tn_div.grid(row=5, colu
tn_open = tk.Button(roc                                         ("("), width=5, font=("Arial", 14), bg="#ccccff", act
tn_open.grid(row=5, col
tn_close = tk.Button(rc                                         n(")"), width=5, font=("Arial", 14), bg="#ccccff", ac
tn_close.grid(row=5, cc
tn_clear = tk.Button(rc                                         nt=("Arial", 14), bg="#ff9999", activebackground="#ff
tn_clear.grid(row=6, column=1, columnspan=2)
tn_equal = tk.Button(root, text="=", command=evaluate_calculation, width=11, font=("Arial", 14), bg="#99ff99", activebackgr
tn_equal.grid(row=6, column=3, columnspan=2)
```