

NAME : P SAI VENKAT.

ROLL NO : 2403A510G0

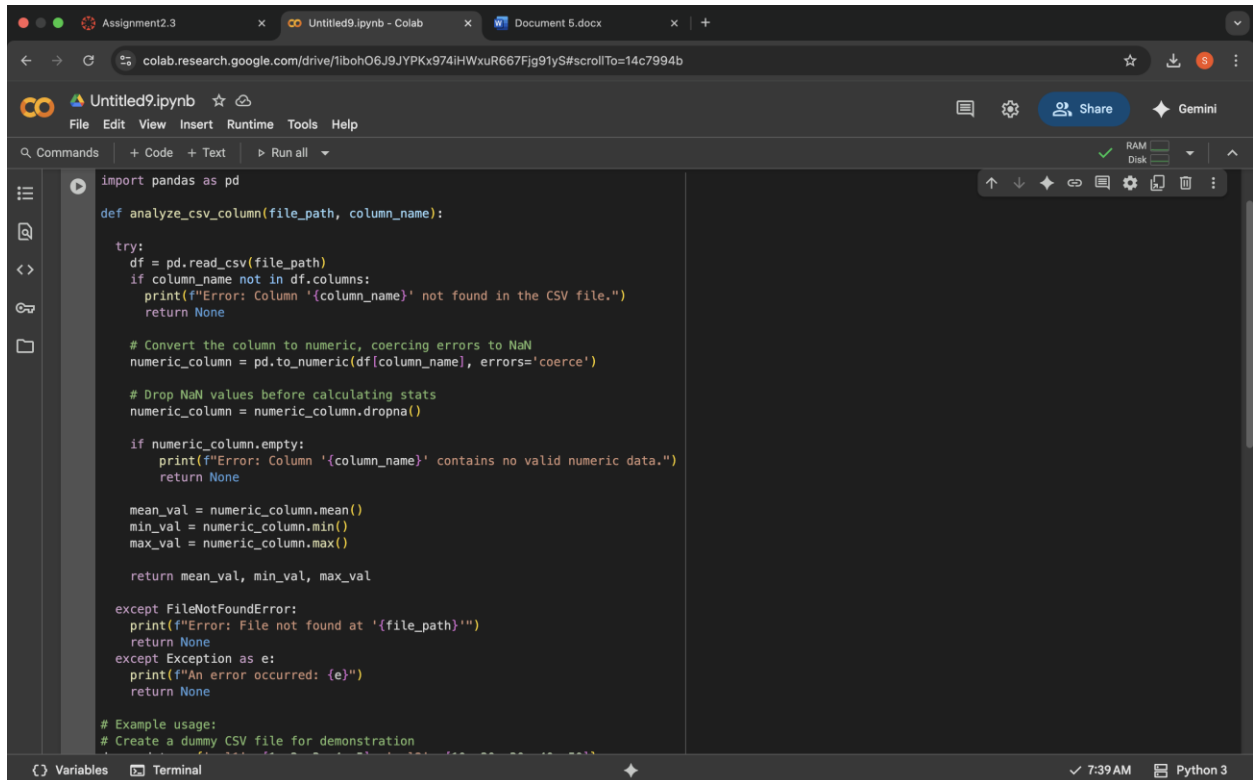
## ASSIGNMENT : 2.3

### Task Description#1

- Use Google Gemini in Colab to write a function that reads a CSV file and calculates mean, min, max.

### Expected Output#1

- Functional code with output and screenshot



The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: `colab.research.google.com/drive/1ibohO6J9JYPKx974iHWxuR667Fjg91yS#scrollTo=14c7994b`. The notebook is titled "Untitled9.ipynb" and has tabs for "Assignment2.3", "Untitled9.ipynb - Colab", and "Document 5.docx". The code editor contains the following Python code:

```
import pandas as pd

def analyze_csv_column(file_path, column_name):

    try:
        df = pd.read_csv(file_path)
        if column_name not in df.columns:
            print(f"Error: Column '{column_name}' not found in the CSV file.")
            return None

        # Convert the column to numeric, coercing errors to NaN
        numeric_column = pd.to_numeric(df[column_name], errors='coerce')

        # Drop NaN values before calculating stats
        numeric_column = numeric_column.dropna()

        if numeric_column.empty:
            print(f"Error: Column '{column_name}' contains no valid numeric data.")
            return None

        mean_val = numeric_column.mean()
        min_val = numeric_column.min()
        max_val = numeric_column.max()

        return mean_val, min_val, max_val

    except FileNotFoundError:
        print(f"Error: File not found at '{file_path}'")
        return None
    except Exception as e:
        print(f"An error occurred: {e}")
        return None

# Example usage:
# Create a dummy CSV file for demonstration
```

The bottom of the interface shows a "Variables" tab and a "Terminal" tab. The status bar at the bottom indicates the time is 7:39 AM and the environment is Python 3.

```
return None

mean_val = numeric_column.mean()
min_val = numeric_column.min()
max_val = numeric_column.max()

return mean_val, min_val, max_val

except FileNotFoundError:
    print(f"Error: File not found at '{file_path}'")
    return None
except Exception as e:
    print(f"An error occurred: {e}")
    return None

# Example usage:
# Create a dummy CSV file for demonstration
dummy_data = {'col1': [1, 2, 3, 4, 5], 'col2': [10, 20, 30, 40, 50]}
dummy_df = pd.DataFrame(dummy_data)
dummy_csv_path = 'dummy.csv'
dummy_df.to_csv(dummy_csv_path, index=False)

# Analyze a column
results = analyze_csv_column(dummy_csv_path, 'col2')

if results:
    mean, min_val, max_val = results
    print(f"Analysis of 'col2' in '{dummy_csv_path}':")
    print(f"Mean: {mean}")
    print(f"Min: {min_val}")
    print(f"Max: {max_val}")

Analysis of 'col2' in 'dummy.csv':
Mean: 30.0
Min: 10
Max: 50
```

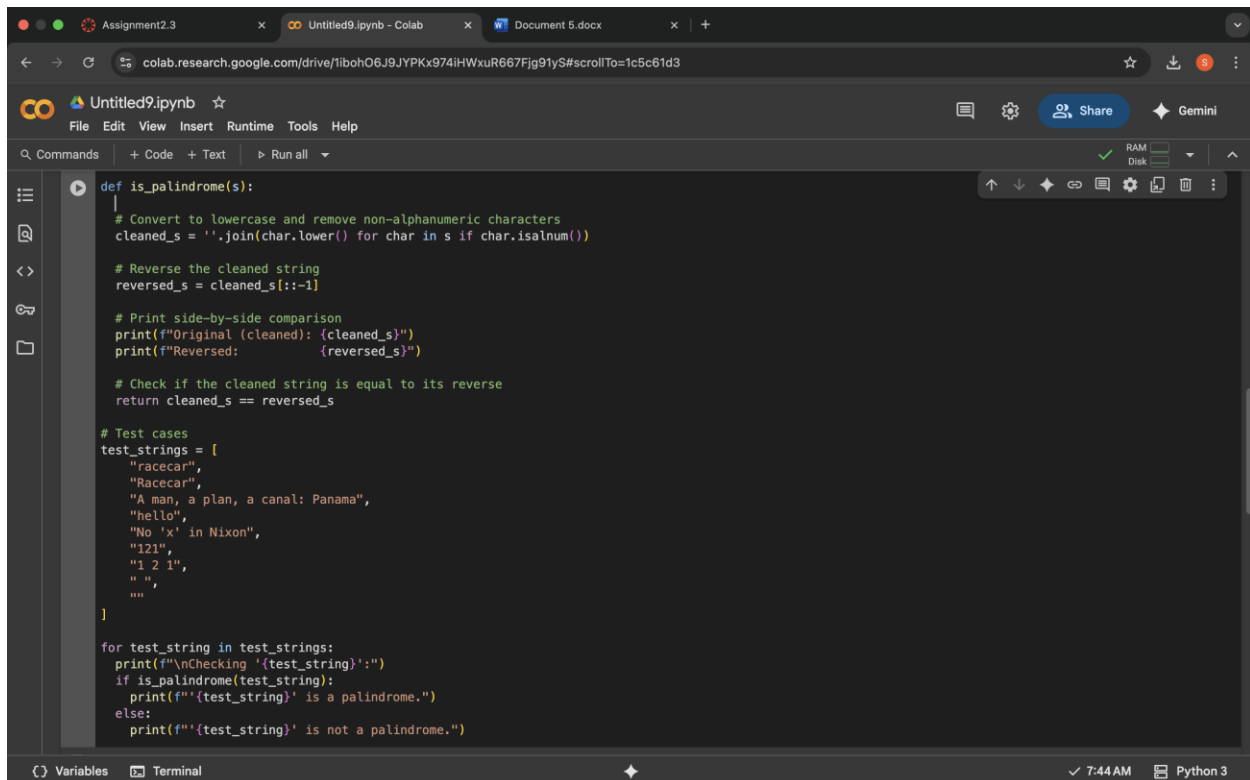
## Task Description#2

- Compare Gemini and Copilot outputs for a palindrome check function.

## Expected Output#2

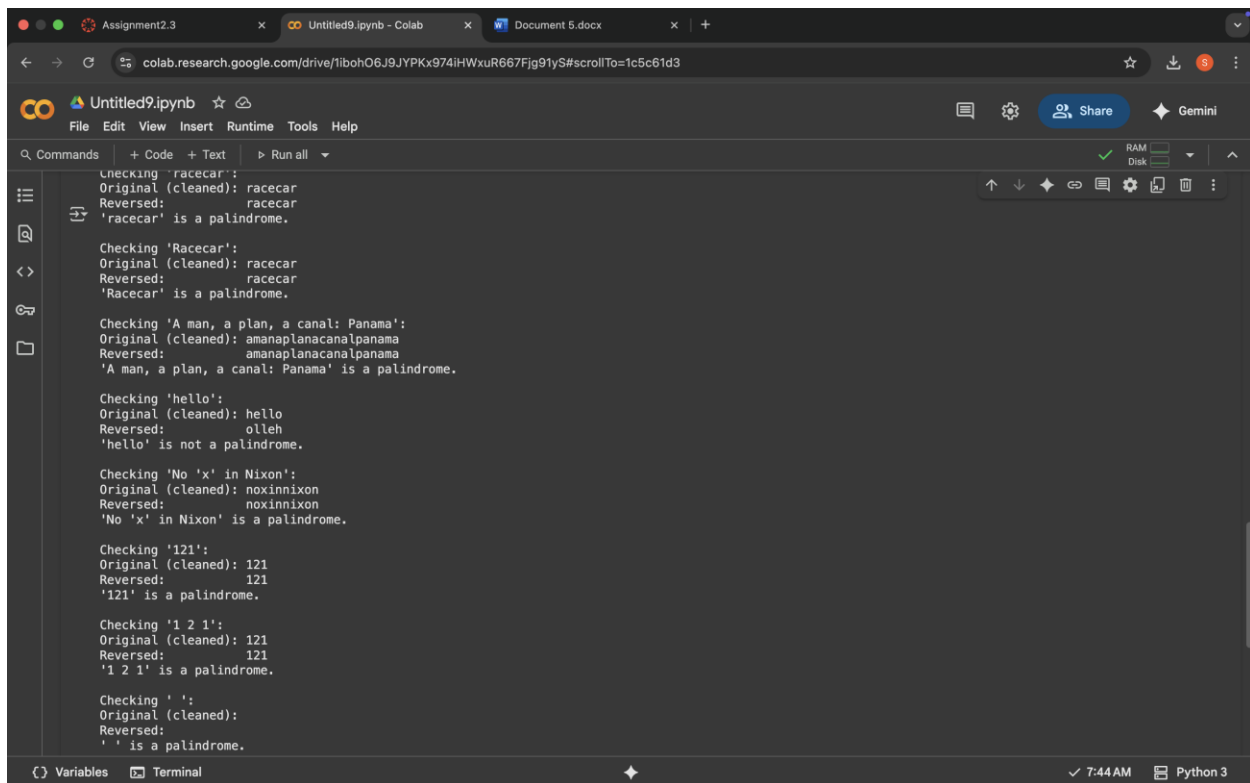
- Side-by-side comparison and observations

GEMINI COLAB :



The screenshot shows a Google Colab notebook titled 'Untitled9.ipynb'. The code defines a function `is_palindrome(s)` that takes a string `s` and returns a boolean. The function first cleans the string by converting it to lowercase and removing non-alphanumeric characters. It then reverses the cleaned string and compares it to the original cleaned string. The code includes several test cases in a list `test_strings`, including 'racecar', 'Racecar', 'A man, a plan, a canal: Panama', 'hello', 'No 'x' in Nixon', '121', and '1 2 1'. The code iterates over these test cases and prints the results of the `is_palindrome` function for each.

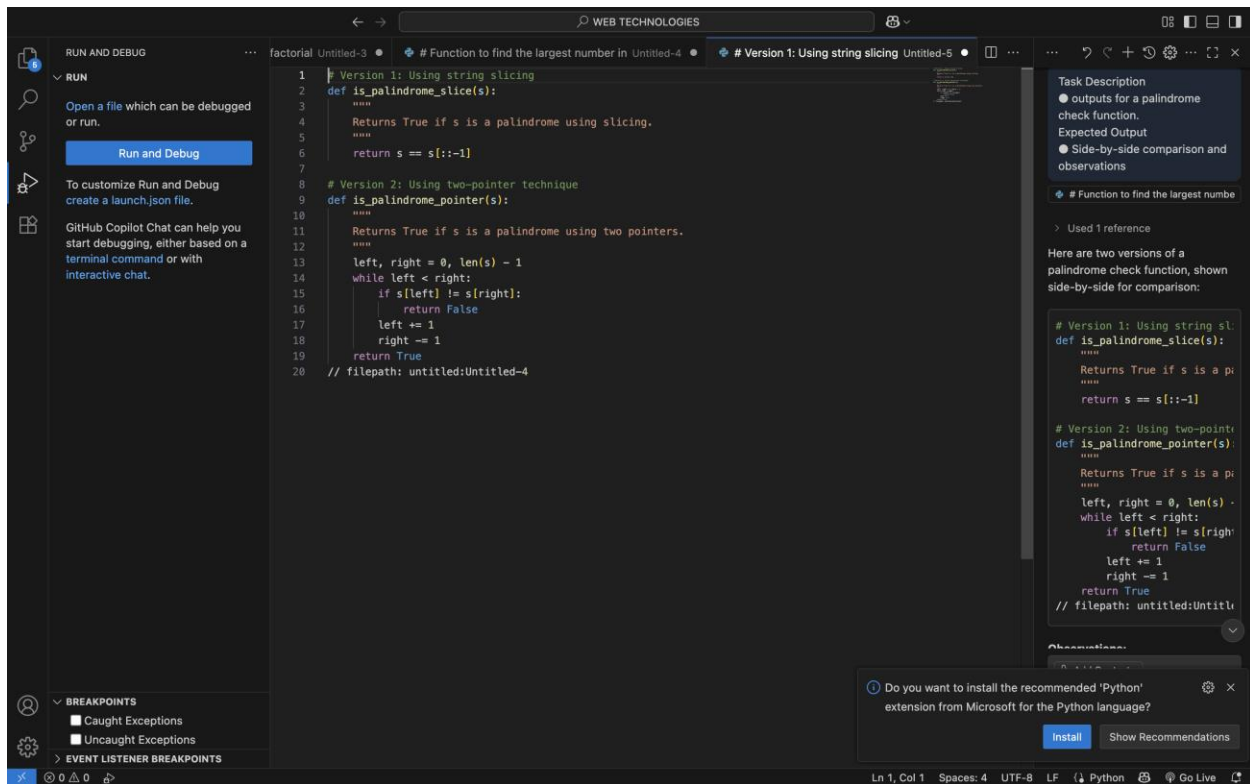
```
def is_palindrome(s):  
    # Convert to lowercase and remove non-alphanumeric characters  
    cleaned_s = ''.join(char.lower() for char in s if char.isalnum())  
  
    # Reverse the cleaned string  
    reversed_s = cleaned_s[::-1]  
  
    # Print side-by-side comparison  
    print(f"Original (cleaned): {cleaned_s}")  
    print(f"Reversed: {reversed_s}")  
  
    # Check if the cleaned string is equal to its reverse  
    return cleaned_s == reversed_s  
  
# Test cases  
test_strings = [  
    "racecar",  
    "Racecar",  
    "A man, a plan, a canal: Panama",  
    "hello",  
    "No 'x' in Nixon",  
    "121",  
    "1 2 1",  
    "",  
    ""  
]  
  
for test_string in test_strings:  
    print(f"\nChecking '{test_string}':")  
    if is_palindrome(test_string):  
        print(f"'{test_string}' is a palindrome.")  
    else:  
        print(f"'{test_string}' is not a palindrome.")
```



The screenshot shows the same Google Colab notebook, but now displaying the output of the code. The output shows the results of the `is_palindrome` function for each test case. The output is as follows:

```
Checking 'racecar':  
Original (cleaned): racecar  
Reversed: racecar  
'racecar' is a palindrome.  
  
Checking 'Racecar':  
Original (cleaned): racecar  
Reversed: racecar  
'Racecar' is a palindrome.  
  
Checking 'A man, a plan, a canal: Panama':  
Original (cleaned): amanaplanacanalpanama  
Reversed: amanaplanacanalpanama  
'A man, a plan, a canal: Panama' is a palindrome.  
  
Checking 'hello':  
Original (cleaned): hello  
Reversed: olleh  
'hello' is not a palindrome.  
  
Checking 'No 'x' in Nixon':  
Original (cleaned): noxinnixon  
Reversed: noxinnixon  
'No 'x' in Nixon' is a palindrome.  
  
Checking '121':  
Original (cleaned): 121  
Reversed: 121  
'121' is a palindrome.  
  
Checking '1 2 1':  
Original (cleaned): 121  
Reversed: 121  
'1 2 1' is a palindrome.  
  
Checking '':  
Original (cleaned):  
Reversed:  
' ' is a palindrome.
```

COPILOT:



OBSERVATIONS :

## Gemini

### 1. Detailed explanation

- Gemini usually first explains what a palindrome number is.
- Then, it gives step-by-step reasoning before writing the code.

### 2. Multiple approaches

- Often shows **string conversion method** (`str(num) == str(num)[::-1]`).
- Sometimes also explains **mathematical method** (reversing digits without converting to string).

### 3. Human-friendly

- Code comes with comments, examples, and test cases.
- Output is usually more "tutorial-style"

## Copilot

### 1. Direct coding focus

- Copilot quickly suggests code with little/no explanation.
- If you type "check if number is palindrome in python", it auto-completes code directly.

### 2. Preference for shortest solution

- Mostly gives the **string conversion method** (one-liner).

- b. Less likely to provide multiple methods unless you explicitly prompt.

### 3. Less descriptive

- a. Minimal comments/documentation.
- b. Very task-oriented → just generates working code fast

#### Task Description#3

- Ask Gemini to explain a Python function (to calculate area of various shapes) line by line..

#### Expected Output#3

- Detailed explanation with code snippet

```
import math

def calculate_area(shape, parameters):
    if shape == 'circle':
        if 'radius' in parameters:
            radius = parameters['radius']
            if radius >= 0:
                return math.pi * radius**2
            else:
                print("Error: Radius cannot be negative.")
                return None
        else:
            print("Error: Missing 'radius' parameter for circle.")
            return None
    elif shape == 'rectangle':
        if 'length' in parameters and 'width' in parameters:
            length = parameters['length']
            width = parameters['width']
            if length >= 0 and width >= 0:
                return length * width
            else:
                print("Error: Length and width cannot be negative.")
                return None
        else:
            print("Error: Missing 'length' or 'width' parameter for rectangle.")
            return None
    elif shape == 'triangle':
        if 'base' in parameters and 'height' in parameters:
            base = parameters['base']
            height = parameters['height']
            if base >= 0 and height >= 0:
                return 0.5 * base * height
            else:
                print("Error: Base and height cannot be negative.")
                return None
        else:
            print("Error: Missing 'base' or 'height' parameter for triangle.")
            return None
    else:
        print(f"Error: Unsupported shape '{shape}'.")
        return None
```

# This line returns None.  
else:  
# This line starts the final conditional block: if the shape is not supported, it prints an error message and returns None.  
print(f"Error: Unsupported shape '{shape}'.")  
# This line prints an error message indicating that the shape is not supported.  
return None  
# This line returns None to indicate that the area cannot be calculated.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Assignment2.3 x 2.3.jynb - Colab x Document 5.docx x Gemini vs Copilot palindrome x +

colab.research.google.com/drive/1ibohO6J9JYPKx974IHxwR667Fjg91yS#scrollTo=5cca6979

2.3.jynb ☆ Saving... File Edit View Insert Runtime Tools Help

Commands + Code + Text ▶ Run all

```
print("Error: Missing 'length' or 'width' parameter for rectangle.")
return None
elif shape == 'triangle':
    if 'base' in parameters and 'height' in parameters:
        base = parameters['base']
        height = parameters['height']
        if base >= 0 and height >= 0:
            return 0.5 * base * height
        else:
            print("Error: Base and height cannot be negative.")
            return None
    else:
        print("Error: Missing 'base' or 'height' parameter for triangle.")
        return None
else:
    print(f"Error: Unsupported shape '{shape}'.")
    return None

# Example usage:
circle_area = calculate_area('circle', {'radius': 5})
print(f"Area of circle with radius 5: {circle_area}")

rectangle_area = calculate_area('rectangle', {'length': 4, 'width': 6})
print(f"Area of rectangle with length 4 and width 6: {rectangle_area}")

triangle_area = calculate_area('triangle', {'base': 3, 'height': 8})
print(f"Area of triangle with base 3 and height 8: {triangle_area}")

invalid_shape_area = calculate_area('square', {'side': 5})
print(f"Area of square with side 5: {invalid_shape_area}")

missing_param_area = calculate_area('circle', {'radius': 5})
print(f"Area of circle with missing parameter: {missing_param_area}")

negative_param_area = calculate_area('rectangle', {'length': -4, 'width': 6})
print(f"Area of rectangle with negative length: {negative_param_area}")
```

# This line returns None.  
else:  
# This line starts the final conditional block: if t  
print(f"Error: Unsupported shape '{shape}'.")  
# This line prints an error message indicating that  
return None  
# This line returns None to indicate that the area c

What can I help you build?  
Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal 7:53 AM Python 3

Assignment2.3 x 2.3.jynb - Colab x Document 5.docx x Gemini vs Copilot palindrome x +

colab.research.google.com/drive/1ibohO6J9JYPKx974IHxwR667Fjg91yS#scrollTo=5cca6979

2.3.jynb ☆ Share Gemini

Commands + Code + Text ▶ Run all

```
print(f"Area of square with side 5: {invalid_shape_area}")

missing_param_area = calculate_area('circle', {})
print(f"Area of circle with missing parameter: {missing_param_area}")

negative_param_area = calculate_area('rectangle', {'length': -4, 'width': 6})
print(f"Area of rectangle with negative length: {negative_param_area}")
```

Area of circle with radius 5: 78.53981633974483  
Area of rectangle with length 4 and width 6: 24  
Area of triangle with base 3 and height 8: 12.0  
Error: Unsupported shape 'square'.  
Area of square with side 5: None  
Error: Missing 'radius' parameter for circle.  
Area of circle with missing parameter: None  
Error: Length and width cannot be negative.  
Area of rectangle with negative length: None

# This line returns None.  
else:  
# This line starts the final conditional block: if t  
print(f"Error: Unsupported shape '{shape}'.")  
# This line prints an error message indicating that  
return None  
# This line returns None to indicate that the area c

What can I help you build?  
Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal 7:53 AM Python 3

EXPLANATION LINE BY LINE :

import math

**This line imports the 'math' module, which provides mathematical functions like pi.**

```
def calculate_area(shape, parameters):
```

**This line defines a function named 'calculate\_area' that takes two arguments: 'shape' (a string) and 'parameters' (a dictionary).**

```
    """Calculates the area of various shapes.
```

**This is a docstring, explaining what the function does.**

```
    Args: shape: A string indicating the type of shape ('circle', 'rectangle', 'triangle'). parameters: A dictionary containing the parameters specific to the shape (e.g., 'radius' for circle, 'length' and 'width' for rectangle, 'base' and 'height' for triangle). # These lines describe the arguments the function expects.
```

```
    Returns: The calculated area of the shape, or None if the shape is invalid or parameters are missing.
```

**This line describes what the function returns.**

```
    """ if shape == 'circle': # This line starts a conditional block: if the 'shape' is 'circle', the following code is executed. if 'radius' in parameters: # This line checks if the 'parameters' dictionary contains the key 'radius'. radius = parameters['radius'] # This line assigns the value associated with the key 'radius' from the 'parameters' dictionary to the variable 'radius'. if radius >= 0: # This line checks if the 'radius' is a non-negative number. return math.pi * radius**2 # If the radius is valid, this line calculates the area of a circle using the formula pi * radius^2 and returns the result. else: # This line starts a block for when the radius is negative. print("Error: Radius cannot be negative.") # This line prints an error message indicating that the radius cannot
```

```

be negative. return None # This line returns None to indicate that the area could not be calculated due to a
negative radius. else: # This line starts a block for when the 'radius' parameter is missing for a circle.
print("Error: Missing 'radius' parameter for circle.") # This line prints an error message indicating the missing
parameter. return None # This line returns None to indicate that the area could not be calculated due to a
missing parameter. elif shape == 'rectangle': # This line starts another conditional block: if the 'shape' is
'rectangle', the following code is executed. if 'length' in parameters and 'width' in parameters: # This line
checks if the 'parameters' dictionary contains both 'length' and 'width' keys. length = parameters['length'] #
This line assigns the value for 'length' from the 'parameters' dictionary to the 'length' variable. width =
parameters['width'] # This line assigns the value for 'width' from the 'parameters' dictionary to the 'width'
variable. if length >= 0 and width >= 0: # This line checks if both 'length' and 'width' are non-negative
numbers. return length * width # If both are valid, this line calculates the area of a rectangle and returns the
result. else: # This line starts a block for when length or width are negative. print("Error: Length and width
cannot be negative.") # This line prints an error message. return None # This line returns None. else: # This
line starts a block for when 'length' or 'width' parameters are missing for a rectangle. print("Error: Missing
'length' or 'width' parameter for rectangle.") # This line prints an error message. return None # This line
returns None. elif shape == 'triangle': # This line starts another conditional block: if the 'shape' is 'triangle', the
following code is executed. if 'base' in parameters and 'height' in parameters: # This line checks if the
'parameters' dictionary contains both 'base' and 'height' keys. base = parameters['base'] # This line assigns
the value for 'base' from the 'parameters' dictionary to the 'base' variable. height = parameters['height'] # This
line assigns the value for 'height' from the 'parameters' dictionary to the 'height' variable. if base >= 0 and
height >= 0: # This line checks if both 'base' and 'height' are non-negative numbers. return 0.5 * base * height
# If both are valid, this line calculates the area of a triangle and returns the result. else: # This line starts a
block for when base or height are negative. print("Error: Base and height cannot be negative.") # This line
prints an error message. return None # This line returns None. else: # This line starts a block for when 'base'
or 'height' parameters are missing for a triangle. print("Error: Missing 'base' or 'height' parameter for triangle.")
# This line prints an error message. return None # This line returns None. else: # This line starts the final
conditional block: if the 'shape' is not any of the above. print(f"Error: Unsupported shape '{shape}'.") # This
line prints an error message indicating that the shape is not supported. return None # This line returns None
to indicate that the area could not be calculated for an unsupported shape.

```

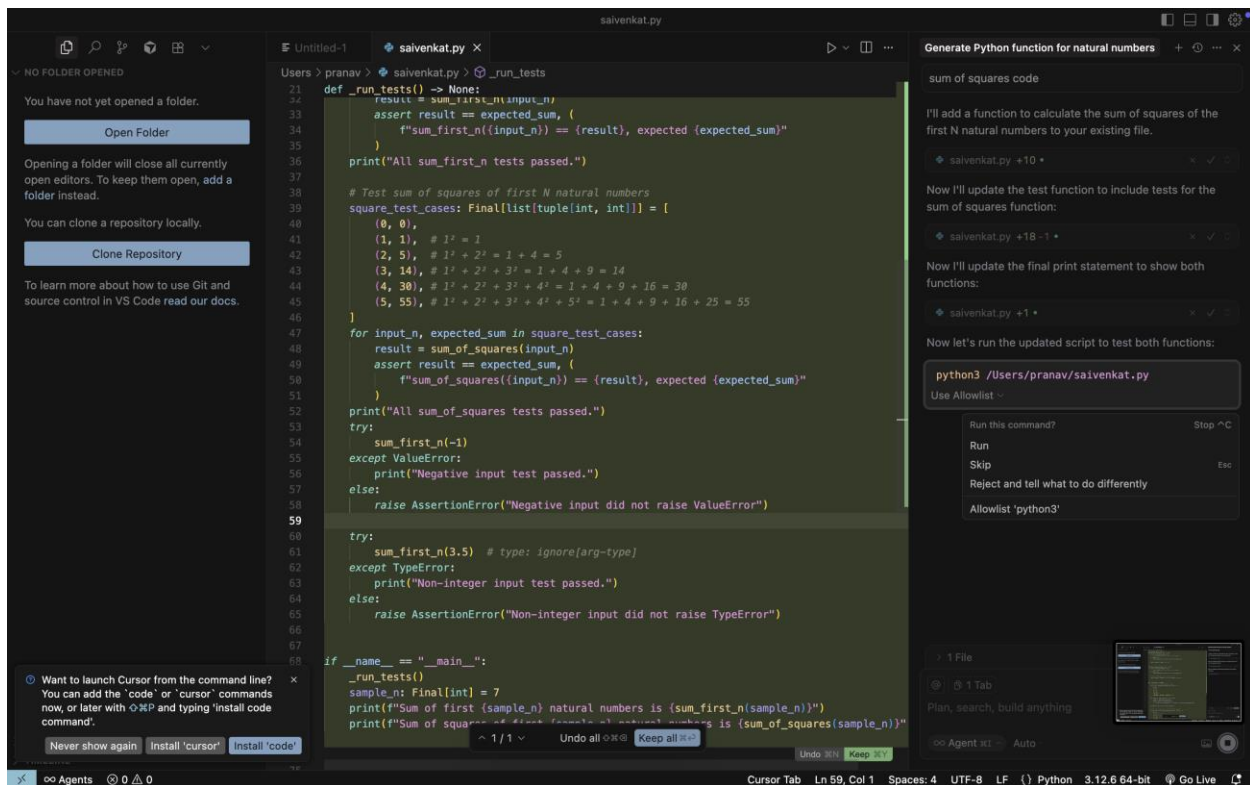
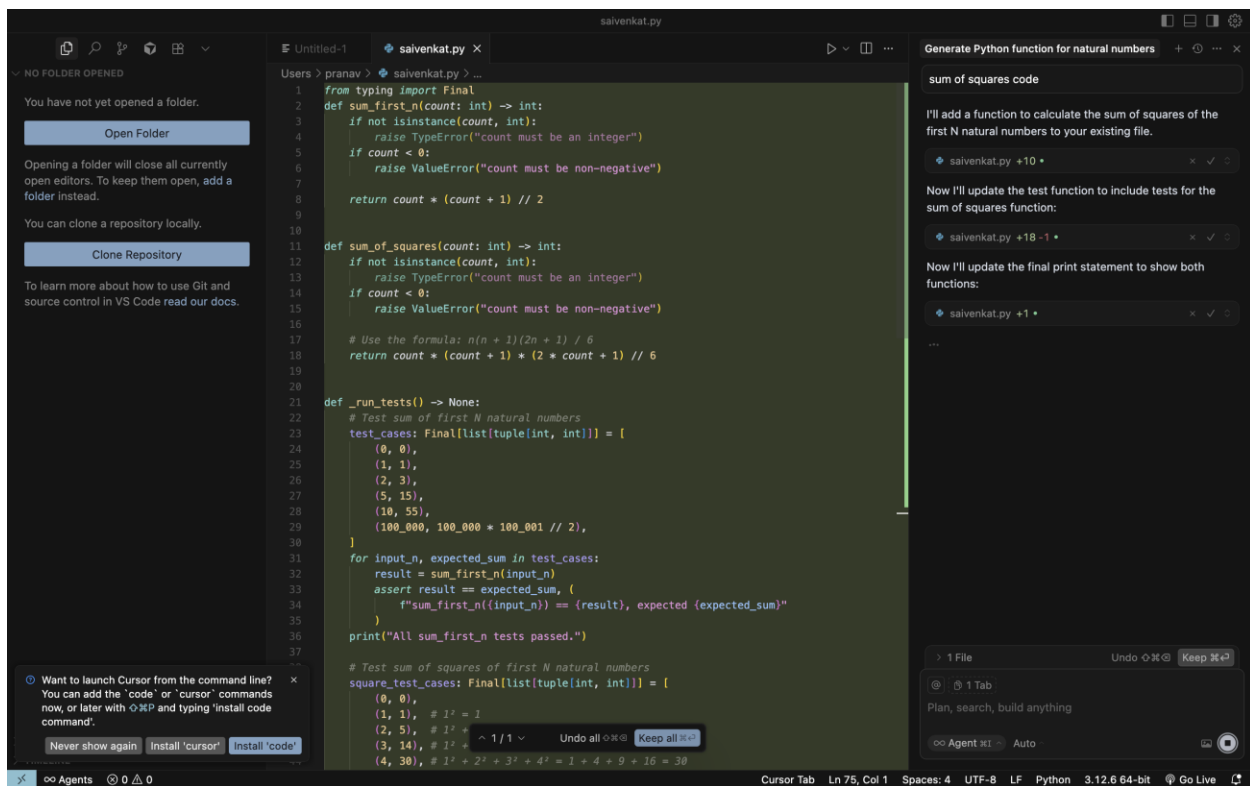
#### Task Description#4

- Install and configure Cursor AI. Use it to generate a Python function (e.g., sum of squares).

#### Expected Output#4

- Screenshots of working environments with few prompts to generate python code





PROMPT GIVEN :

Write a python code to generate sum of squares of the given number and output according to requirements

#### Task Description#5

- Student need to write code to calculate sum of add number and even numbers in the list

#### Expected Output#5

- Refactored code written by student with improved logic

