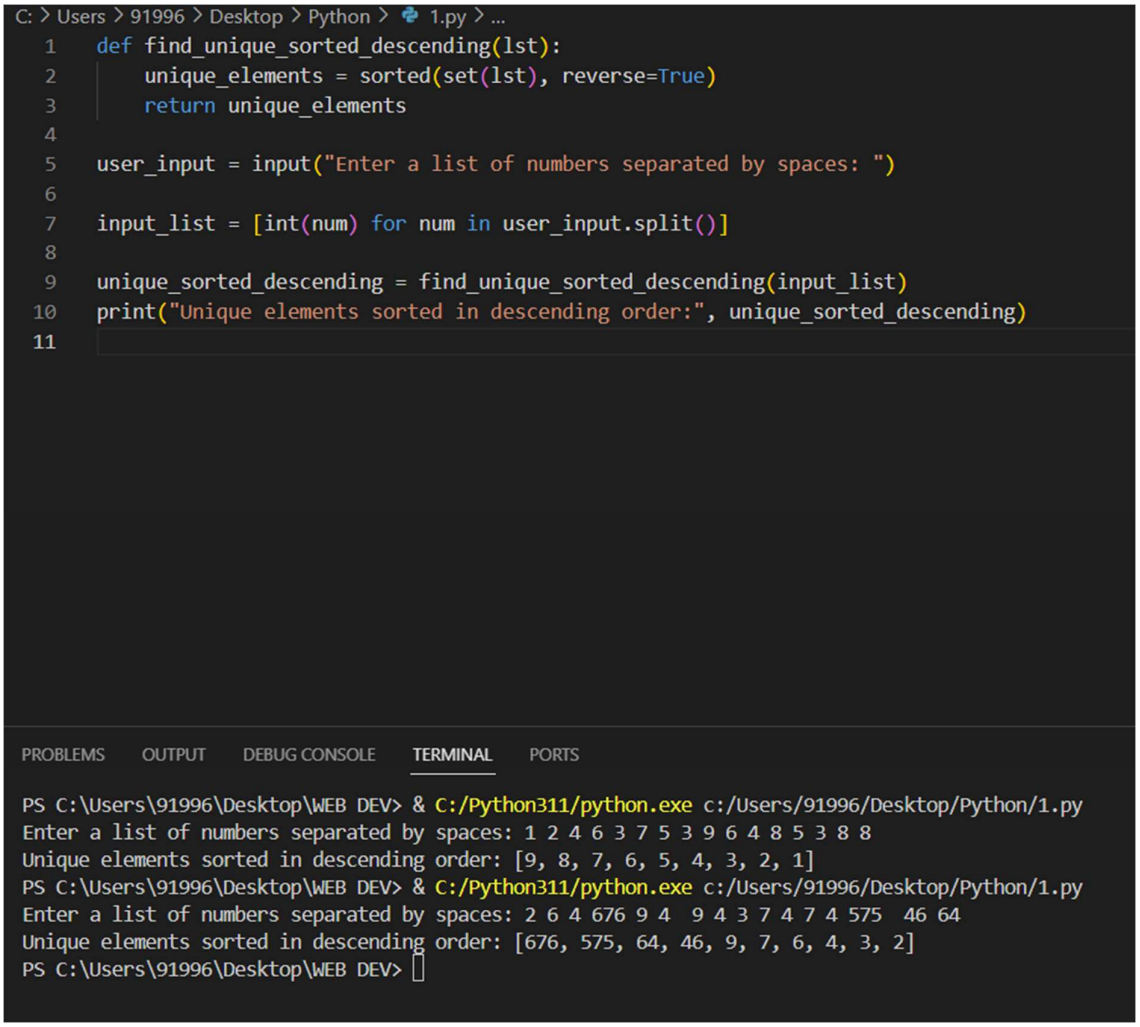
Part – 1 Python Fundamentals

Data Structures and Manipulation:

o Implement a function to find unique elements from a list and sort them in descending order.



o Convert a nested dictionary structure into a flattened dictionary.

C: > Users > 91996 > Desktop > Python > 1.py > ...

```python
def flatten_nested_dict(nested_dict, parent_key='', sep='_'):
    """Convert a nested dictionary structure into a flattened dictionary."""
    items = []
    for k, v in nested_dict.items():
        new_key = parent_key + sep + k if parent_key else k
        if isinstance(v, dict):
            items.extend(flatten_nested_dict(v, new_key, sep=sep).items())
        else:
            items.append((new_key, v))
    return dict(items)

# Example usage:

nested_dict = {
    'a': 1,
    'b': {
        'c': 2,
        'd': {
            'e': 3
        }
    }
}
flattened_dict = flatten_nested_dict(nested_dict)
print("Flattened dictionary:", flattened_dict)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\91996\Desktop\WEB DEV> & C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Flattened dictionary: {'a': 1, 'b_c': 2, 'b_d_e': 3}
PS C:\Users\91996\Desktop\WEB DEV>
```

Object-Oriented Programming (OOP):

o Define a class Calculator that supports basic arithmetic operations (addition, subtraction, multiplication, division). Include error handling (e.g., division by zero).

C: > Users > 91996 > Desktop > Python > 1.py > ...

```python
class Calculator:
    def add(self, x, y):
        return x + y

    def subtract(self, x, y):
        return x - y

    def multiply(self, x, y):
        return x * y

    def divide(self, x, y):
        try:
            result = x / y
        except ZeroDivisionError:
            return "Error: Division by zero is not allowed."
        else:
            return result
try:
    num1 = float(input("Enter the first number: "))
    operator = input("Enter the operator (+, -, *, /): ")
    num2 = float(input("Enter the second number: "))
except ValueError:
    print("Error: Please enter valid numbers.")
    exit()
calc = Calculator()

if operator == '+':
    print("Result:", calc.add(num1, num2))
elif operator == '-':
    print("Result:", calc.subtract(num1, num2))
elif operator == '*':
    print("Result:", calc.multiply(num1, num2))
elif operator == '/':
    print("Result:", calc.divide(num1, num2))
else:
    print("Error: Invalid operator.")
```

```
Error: Please enter valid numbers.
PS C:\Users\91996\Desktop\WEB DEV> & C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Enter the first number: 69
Enter the operator (+, -, *, /): +
Enter the second number: 3
Result: 72.0
PS C:\Users\91996\Desktop\WEB DEV>
PS C:\Users\91996\Desktop\WEB DEV> C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Enter the first number: 72
Enter the operator (+, -, *, /): /
Enter the second number: 0
Result: Error: Division by zero is not allowed.
PS C:\Users\91996\Desktop\WEB DEV> C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Enter the first number: 96
Enter the operator (+, -, *, /): *
Enter the second number: 3
Result: 288.0
PS C:\Users\91996\Desktop\WEB DEV>
```

o Demonstrate inheritance by extending the Calculator class to

a ScientificCalculator that adds trigonometric functions.

C: > Users > 91996 > Desktop > Python > 1.py > ...

```python
import math

class Calculator:
    def add(self, x, y):
        return x + y

    def subtract(self, x, y):
        return x - y

    def multiply(self, x, y):
        return x * y

    def divide(self, x, y):
        try:
            result = x / y
        except ZeroDivisionError:
            return "Error: Division by zero is not allowed."
        else:
            return result

class ScientificCalculator(Calculator):
    def sin(self, x):
        return math.sin(math.radians(x))

    def cos(self, x):
        return math.cos(math.radians(x))

    def tan(self, x):
        return math.tan(math.radians(x))

```

C: > Users > 91996 > Desktop > Python > 1.py > ...

```
try:
    num1 = float(input("Enter the number: "))
    operator = input("Enter the operation (+, -, *, /, sin, cos, tan): ")
except ValueError:
    print("Error: Please enter a valid number.")
    exit()

sci_calc = ScientificCalculator()


if operator in ['+', '-', '*', '/']:
    try:
        num2 = float(input("Enter the second number: "))
    except ValueError:
        print("Error: Please enter a valid number.")
        exit()
    calc = Calculator()
    if operator == '+':
        print("Result:", calc.add(num1, num2))
    elif operator == '-':
        print("Result:", calc.subtract(num1, num2))
    elif operator == '*':
        print("Result:", calc.multiply(num1, num2))
    elif operator == '/':
        print("Result:", calc.divide(num1, num2))
elif operator in ['sin', 'cos', 'tan']:
    if operator == 'sin':
        print("Result:", sci_calc.sin(num1))
    elif operator == 'cos':
        print("Result:", sci_calc.cos(num1))
    elif operator == 'tan':
```

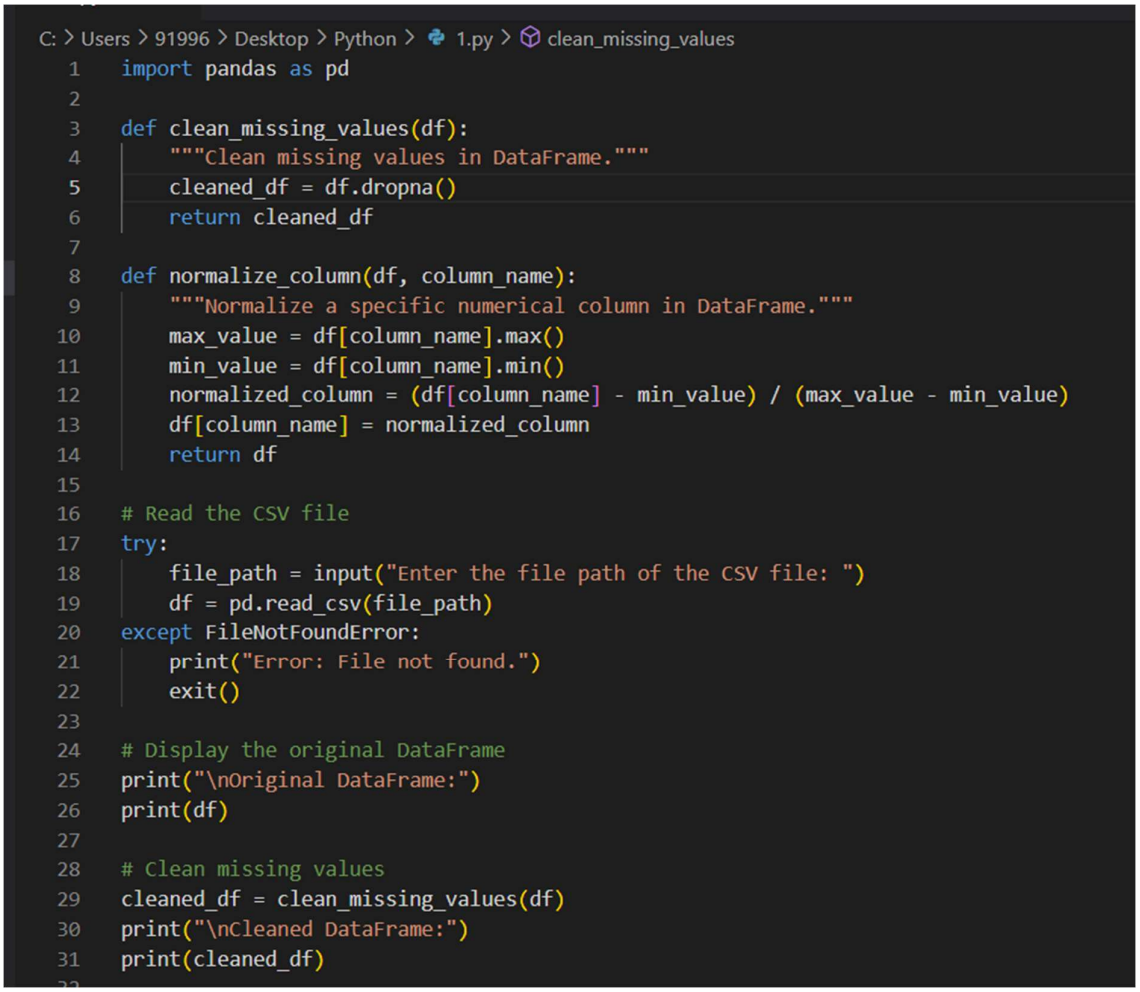
```
        print("Result:", sci_calc.sin(num1))
    elif operator == 'cos':
        print("Result:", sci_calc.cos(num1))
    elif operator == 'tan':
        print("Result:", sci_calc.tan(num1))
else:
    print("Error: Invalid operator.")

```

```
>>     # Use calculator methods
PS C:\Users\91996\Desktop\WEB DEV> & C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Enter the number: 30
Enter the operation (+, -, *, /, sin, cos, tan): sin
Result: 0.49999999999999994
PS C:\Users\91996\Desktop\WEB DEV> C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Enter the number: 60
Enter the operation (+, -, *, /, sin, cos, tan): cos
Result: 0.5000000000000001
PS C:\Users\91996\Desktop\WEB DEV> C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Enter the number: 45
Enter the operation (+, -, *, /, sin, cos, tan): tan
Result: 0.9999999999999999
PS C:\Users\91996\Desktop\WEB DEV>
```

File I/O:

o Read a CSV file, clean missing values, and normalize a specific numerical column.

o Write the cleaned data into a new JSON file.



```python
# Normalize a specific numerical column
try:
    column_name = input("Enter the name of the column to normalize: ")
    cleaned_df = normalize_column(cleaned_df, column_name)
except KeyError:
    print("Error: Column not found.")
    exit()

# Display the normalized DataFrame
print("\nNormalized DataFrame:")
print(cleaned_df)

# Write the cleaned data to a JSON file
output_file_path = "cleaned_data.json"
cleaned_df.to_json(output_file_path, orient='records', lines=True)
print(f"\nCleaned data has been written to '{output_file_path}' as JSON.")

```

```
PS C:\Users\91996\Desktop\WEB DEV> & C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Enter the file path of the CSV file: c:/Users/91996/Desktop/sample_data.csv

Original DataFrame:
    Name   Age  Salary
0   John  30.0   50000
1  Alice   NaN   60000
2    Bob  40.0   70000

Cleaned DataFrame:
   Name   Age  Salary
0  John  30.0   50000
2   Bob  40.0   70000
Enter the name of the column to normalize: Age
c:\Users\91996\Desktop\Python\1.py:13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[column_name] = normalized_column

Normalized DataFrame:
   Name  Age  Salary
0  John  0.0   50000
2   Bob  1.0   70000

Cleaned data has been written to 'cleaned_data.json' as JSON.
PS C:\Users\91996\Desktop\WEB DEV>
```

Created JSON File: