

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.

```
C: > Users > 91996 > Desktop > Python > 1.py > ...
1  def find_unique_sorted_descending(lst):
2      unique_elements = sorted(set(lst), reverse=True)
3      return unique_elements
4
5  user_input = input("Enter a list of numbers separated by spaces: ")
6
7  input_list = [int(num) for num in user_input.split()]
8
9  unique_sorted_descending = find_unique_sorted_descending(input_list)
10 print("Unique elements sorted in descending order:", unique_sorted_descending)
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\91996\Desktop\WEB DEV> & C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Enter a list of numbers separated by spaces: 1 2 4 6 3 7 5 3 9 6 4 8 5 3 8 8
Unique elements sorted in descending order: [9, 8, 7, 6, 5, 4, 3, 2, 1]
PS C:\Users\91996\Desktop\WEB DEV> & C:/Python311/python.exe c:/Users/91996/Desktop/Python/1.py
Enter a list of numbers separated by spaces: 2 6 4 676 9 4 9 4 3 7 4 7 4 575 46 64
Unique elements sorted in descending order: [676, 575, 64, 46, 9, 7, 6, 4, 3, 2]
PS C:\Users\91996\Desktop\WEB DEV>
```

- o Convert a nested dictionary structure into a flattened dictionary.

```
C:\Users\91996\Desktop\Python> 1.py > ...
1  def flatten_nested_dict(nested_dict, parent_key='', sep='_'):
2      """Convert a nested dictionary structure into a flattened dictionary."""
3      items = []
4      for k, v in nested_dict.items():
5          new_key = parent_key + sep + k if parent_key else k
6          if isinstance(v, dict):
7              items.extend(flatten_nested_dict(v, new_key, sep=sep).items())
8          else:
9              items.append((new_key, v))
10     return dict(items)
11
12 # Example usage:
13
14 nested_dict = {
15     'a': 1,
16     'b': {
17         'c': 2,
18         'd': {
19             'e': 3
20         }
21     }
22 }
23 flattened_dict = flatten_nested_dict(nested_dict)
24 print("Flattened dictionary:", flattened_dict)
25
```

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 > ...
1  class Calculator:
2      def add(self, x, y):
3          return x + y
4
5      def subtract(self, x, y):
6          return x - y
7
8      def multiply(self, x, y):
9          return x * y
10
11     def divide(self, x, y):
12         try:
13             result = x / y
14         except ZeroDivisionError:
15             return "Error: Division by zero is not allowed."
16         else:
17             return result
18
19     try:
20         num1 = float(input("Enter the first number: "))
21         operator = input("Enter the operator (+, -, *, /): ")
22         num2 = float(input("Enter the second number: "))
23     except ValueError:
24         print("Error: Please enter valid numbers.")
25         exit()
26
27     calc = Calculator()
28
29     if operator == '+':
30         print("Result:", calc.add(num1, num2))
31     elif operator == '-':
32         print("Result:", calc.subtract(num1, num2))
33     elif operator == '*':
34         print("Result:", calc.multiply(num1, num2))
35     elif operator == '/':
36         print("Result:", calc.divide(num1, num2))
37     else:
38         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 > ...
1  import math
2
3  class Calculator:
4      def add(self, x, y):
5          |   return x + y
6
7      def subtract(self, x, y):
8          |   return x - y
9
10     def multiply(self, x, y):
11         |   return x * y
12
13     def divide(self, x, y):
14         |   try:
15             |       result = x / y
16         |   except ZeroDivisionError:
17             |       return "Error: Division by zero is not allowed."
18         |   else:
19             |       return result
20
21     class ScientificCalculator(Calculator):
22         def sin(self, x):
23             |   return math.sin(math.radians(x))
24
25         def cos(self, x):
26             |   return math.cos(math.radians(x))
27
28         def tan(self, x):
29             |   return math.tan(math.radians(x))
30
```

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

```
31 try:
32     num1 = float(input("Enter the number: "))
33     operator = input("Enter the operation (+, -, *, /, sin, cos, tan): ")
34 except ValueError:
35     print("Error: Please enter a valid number.")
36     exit()
37
38 sci_calc = ScientificCalculator()
39
40
41 if operator in ['+', '-', '*', '/']:
42     try:
43         num2 = float(input("Enter the second number: "))
44     except ValueError:
45         print("Error: Please enter a valid number.")
46         exit()
47     calc = Calculator()
48     if operator == '+':
49         print("Result:", calc.add(num1, num2))
50     elif operator == '-':
51         print("Result:", calc.subtract(num1, num2))
52     elif operator == '*':
53         print("Result:", calc.multiply(num1, num2))
54     elif operator == '/':
55         print("Result:", calc.divide(num1, num2))
56 elif operator in ['sin', 'cos', 'tan']:
57     if operator == 'sin':
58         print("Result:", sci_calc.sin(num1))
59     elif operator == 'cos':
60         print("Result:", sci_calc.cos(num1))
61     elif operator == 'tan':
62         print("Result:", sci_calc.tan(num1))
63 else:
64     print("Error: Invalid operator.")
65
```

```
58     print("Result:", sci_calc.sin(num1))
59 elif operator == 'cos':
60     print("Result:", sci_calc.cos(num1))
61 elif operator == 'tan':
62     print("Result:", sci_calc.tan(num1))
63 else:
64     print("Error: Invalid operator.")
65
```

```
77 # 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.

```
C: > Users > 91996 > Desktop > Python > 1.py > clean_missing_values
1  import pandas as pd
2
3  def clean_missing_values(df):
4      """Clean missing values in DataFrame."""
5      cleaned_df = df.dropna()
6      return cleaned_df
7
8  def normalize_column(df, column_name):
9      """Normalize a specific numerical column in DataFrame."""
10     max_value = df[column_name].max()
11     min_value = df[column_name].min()
12     normalized_column = (df[column_name] - min_value) / (max_value - min_value)
13     df[column_name] = normalized_column
14     return df
15
16 # Read the CSV file
17 try:
18     file_path = input("Enter the file path of the CSV file: ")
19     df = pd.read_csv(file_path)
20 except FileNotFoundError:
21     print("Error: File not found.")
22     exit()
23
24 # Display the original DataFrame
25 print("\nOriginal DataFrame:")
26 print(df)
27
28 # Clean missing values
29 cleaned_df = clean_missing_values(df)
30 print("\nCleaned DataFrame:")
31 print(cleaned_df)
32
```



```

32
33 # Normalize a specific numerical column
34 try:
35     column_name = input("Enter the name of the column to normalize: ")
36     cleaned_df = normalize_column(cleaned_df, column_name)
37 except KeyError:
38     print("Error: Column not found.")
39     exit()
40
41 # Display the normalized DataFrame
42 print("\nNormalized DataFrame:")
43 print(cleaned_df)
44
45 # Write the cleaned data to a JSON file
46 output_file_path = "cleaned_data.json"
47 cleaned_df.to_json(output_file_path, orient='records', lines=True)
48 print(f"\nCleaned data has been written to '{output_file_path}' as JSON.")
49

```

```

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:

```

1.py  {} cleaned_data.json 1 X
{} cleaned_data.json > ...
1  {"Name": "John", "Age": 0.0, "Salary": 50000}
2  {"Name": "Bob", "Age": 1.0, "Salary": 70000}
3

```