Data Structures and Analysis in Python: A Practical Approach

Sabbir Ahmed

February 5, 2025

1 Introduction

This report presents a comprehensive implementation of various Python programming concepts, focusing on data structures, numerical computations, data analysis, and visualization. The implementations demonstrate proficiency in using essential Python libraries such as NumPy, Pandas, and Matplotlib.

More information on Github repo

2 Basic Python Data Structures

2.1 List Operations

2.1.1 Objective

Implementation of list operations to remove duplicates and sort numbers in ascending order, demonstrating fundamental list manipulation techniques in Python.

2.1.2 Implementation

```
numlist = [3, 9, 4, 2, 4, 1, 5, 6, 10, 8, 7, 9]

res = []

for i in numlist:
    if i not in res:
        res.append(i)

res.sort()
print("result 1: ", res)

print("result 2: ", sorted(list(set(numlist))))
```

Listing 1: List Operations Implementation

2.1.3 Output

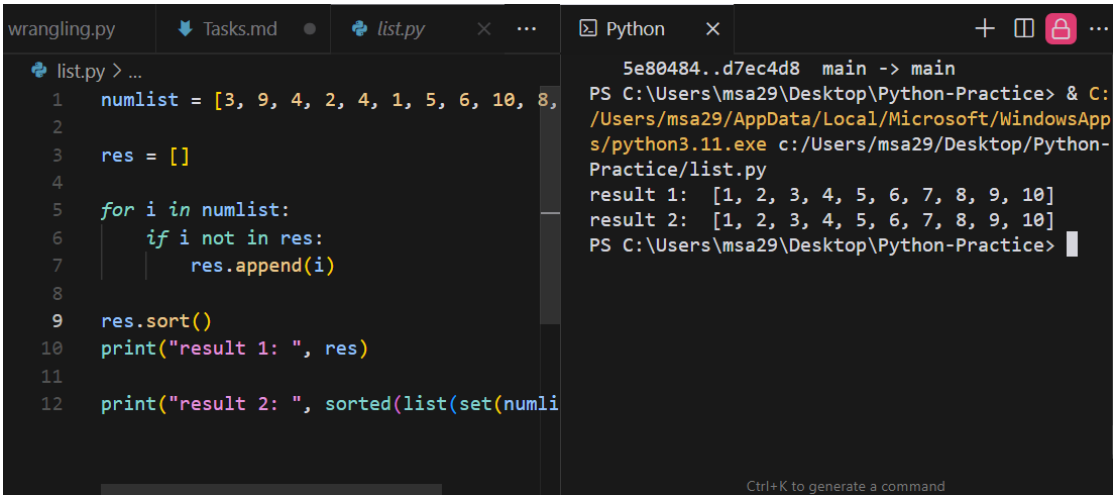

Figure 1: List

2.2 Set Operations

2.2.1 Objective

Demonstration of set operations to find common elements between two lists, showcasing the efficiency of set operations in Python.

2.2.2 Implementation

```
numlist_1 = [3, 9, 4, 2, 4, 1, 5, 6, 10, 8, 7, 9]
numlist_2 = [2, 2, 3, 2, 5, 6, 0, 8, 12 ]

set_1 = set(numlist_1)
set_2 = set(numlist_2)

method_1 = list(set_1 & set_2)
method_2 = list(set_1.intersection(set_2))

print("method_1: ", method_1)
print("method_2: ", method_2)
```

Listing 2: Set Operations Implementation

2.2.3 Output

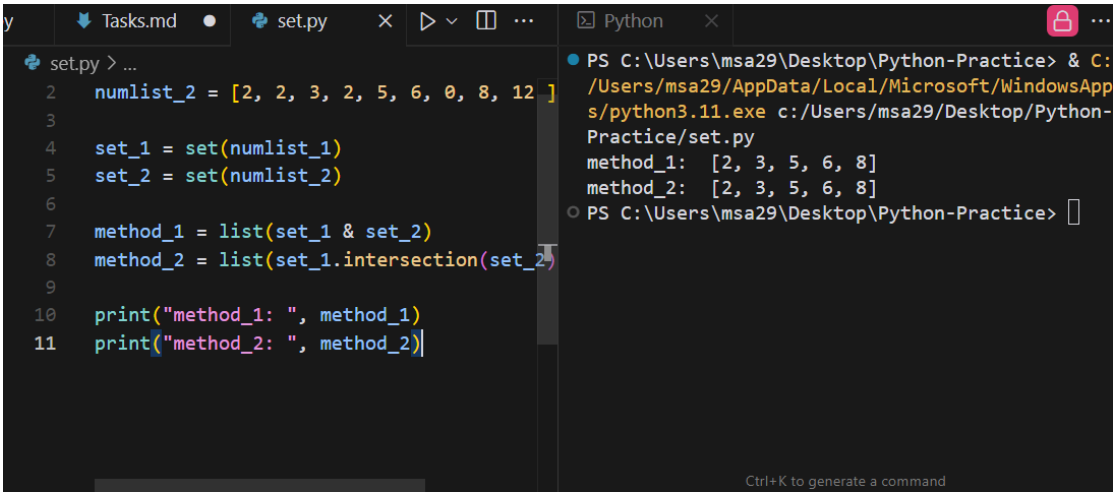

Figure 2: Set

2.3 Tuple Operations

2.3.1 Objective

Creation and manipulation of student records using tuples, demonstrating the immutable nature of tuples and sorting operations.

2.3.2 Implementation

```
students = (
    ("Sabbir", 22, 3.6),
    ("Jiyon", 23, 3.8),
    ("Ishaq", 24, 3.7),
    ("Rakib", 22, 3.5),
    ("Sakib", 23, 3.6)
)

def student_sort(student):
    return student[2]

sorted_students = sorted(students, key=student_sort)

for student in sorted_students:
    print(student)
```

Listing 3: Tuple Operations Implementation

2.3.3 Output

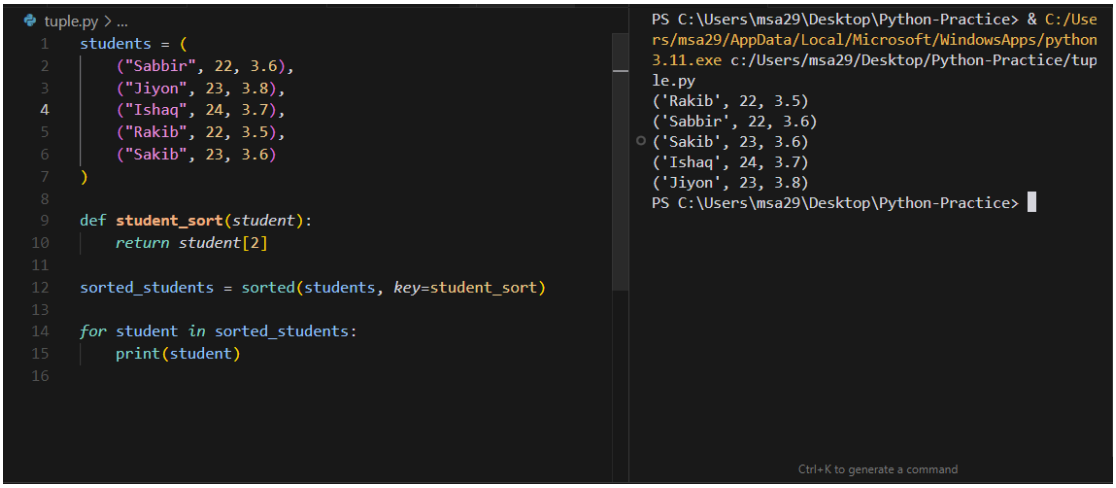

Figure 3: Tuple

2.4 Dictionary Operations

2.4.1 Objective

Implementation of a word occurrence counter using dictionaries, showcasing text analysis capabilities.

2.4.2 Implementation

```
text = "Hello there, I'm Sabbir Ahmed, a passionate and curious
    individual from Dhaka, Bangladesh. As a Computer Science and
    Engineering student at Green University of Bangladesh, I am
    constantly seeking new opportunities to expand my knowledge and
     skills in the software engineering and data science field."

words = text.split()

def word_count(words):
    word_count = {}
    for word in words:
        if word in word_count:
            word_count[word] += 1
        else:
            word_count[word] = 1
    return word_count

text_dict = word_count(words)


for i in text_dict:
    print(i, text_dict[i])
```

Listing 4: Dictionary Operations Implementation

2.4.3 Output

```
Hello 1
there, 1
I’m 1
Sabbir 1
Ahmed, 1
a 2
passionate 1
and 4
curious 1
individual 1
from 1
Dhaka, 1
Bangladesh. 1
As 1
Computer 1
Science 1
Engineering 1
student 1
at 1
Green 1
University 1
of 1
Bangladesh, 1
I 1
am 1
constantly 1
seeking 1
new 1
opportunities 1
to 1
expand 1
my 1
knowledge 1
skills 1
in 1
the 1
software 1
engineering 1
data 1
science 1
field. 1
```

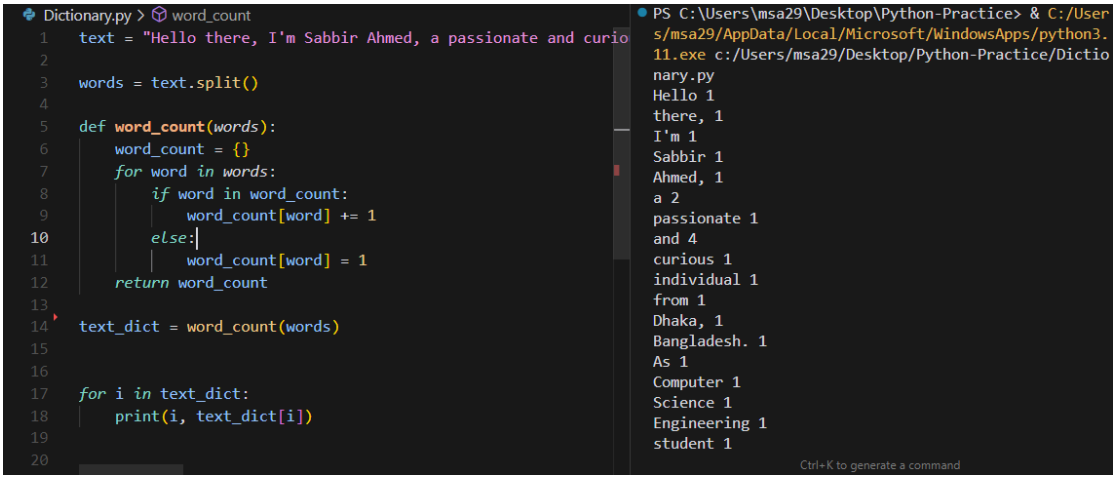

Figure 4: Python Dictionary

3 NumPy Operations

3.1 Matrix Operations

3.1.1 Objective

Generation and manipulation of a 5x5 random integer matrix, demonstrating NumPy's capabilities for matrix operations.

3.1.2 Technical Details

The implementation uses:

- np.random.randint for matrix generation
- np.sum with axis parameter for row-wise calculations

3.1.3 Implementation

```
import numpy as np

matrix = np.random.randint(0, 10, size=(5, 5))

row_sums = np.sum(matrix, axis=1)


print("Matrix:")
print(matrix)
print("\nRow-wise sums:")
print(row_sums)
```

Listing 5: Matrix Operations Implementation

3.1.4 Output

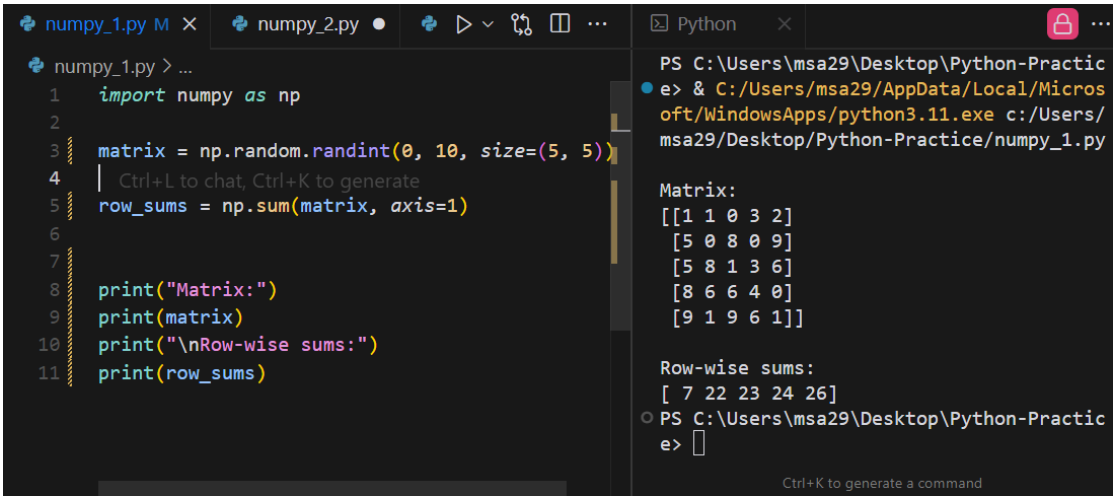

Figure 5: Numpy: row wise sum

3.2 Array Normalization

3.2.1 Objective

Generation and normalization of random arrays, demonstrating advanced array operations using NumPy.

3.2.2 Technical Details

The implementation includes:

- Generation of 100 random values using np.random.rand
- Normalization using the formula: $(x - min(x))/(max(x) - min(x))$

3.2.3 Implementation

```
import numpy as np

arr = np.random.rand(100)

print("Original array:")
print(arr)
print("\nOriginal array min:", np.min(arr))
print("Original array max:", np.max(arr))

normalized_arr = (arr - np.min(arr)) / (np.max(arr) - np.min(arr))

print("\nNormalized array:")
print(normalized_arr)
```

```
print("\nNormalized array min:", np.min(normalized_arr))
print("Normalized array max:", np.max(normalized_arr))
```

Listing 6: Array Normalization Implementation

3.2.4 Output

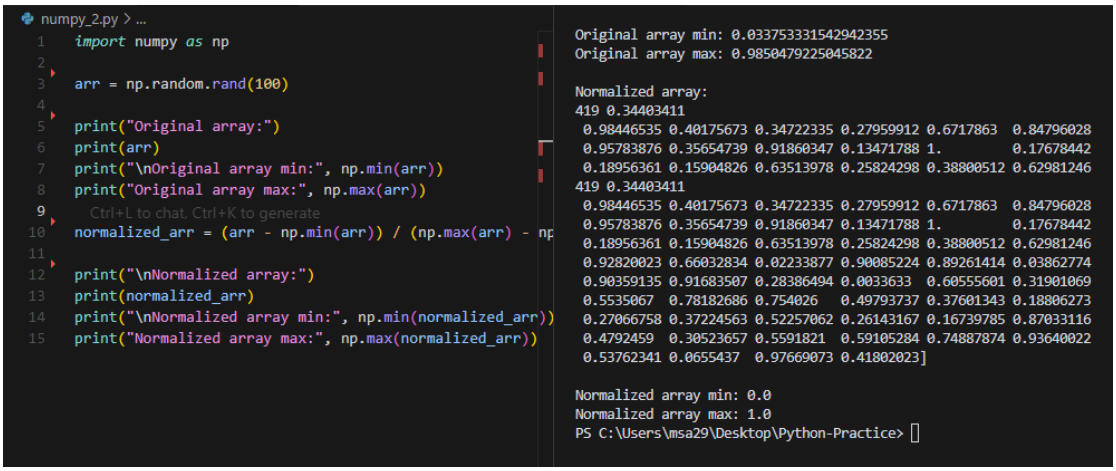

Figure 6: Numpy Matrix Normalization

4 Pandas Data Analysis

4.1 Revenue Analysis

4.1.1 Objective

Analysis of furniture sales data using Pandas, demonstrating data grouping and aggregation techniques.

4.1.2 Technical Details

The implementation features:

- Data grouping by product category
- Revenue and sales calculations
- Advanced aggregation techniques

4.1.3 Implementation

```python
import pandas as pd

df = pd.read_csv('data/Furniture.csv')

revenue_by_product = df.groupby('category').agg({
    'revenue': 'sum',
    'sales': 'sum',
}).round(2)

print("\nDetailed Summary by Product Category:")
print(revenue_by_product)
```

Listing 7: Revenue Analysis Implementation

4.1.4 Output

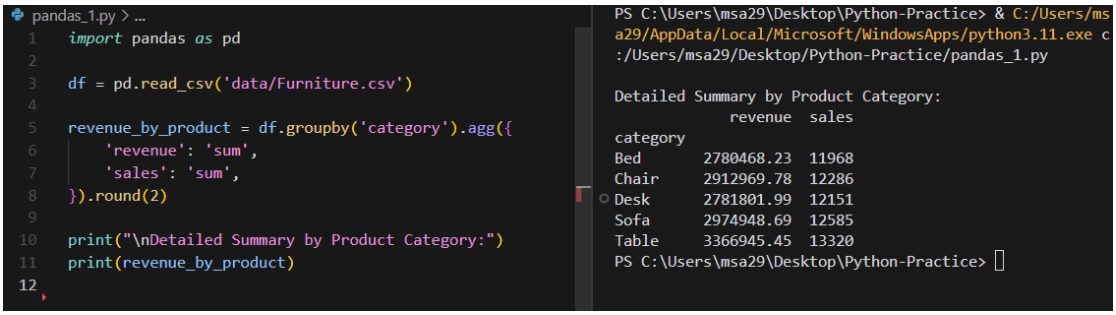

Figure 7: Sales Revenue

4.2 Missing Value Treatment

4.2.1 Objective

Handling missing values in furniture data, demonstrating data cleaning and preprocessing techniques.

4.2.2 Technical Details

Key features include:

- Null value detection using isnull()
- Numeric column separation
- Mean calculation and imputation

4.2.3 Implementation

```python
import pandas as pd

df = pd.read_csv('data/Furniture copy.csv')

print("Null values before filling:")
print(df.isnull().sum())

numeric_cols = df.select_dtypes(include=['float64', 'int64']).
    columns
column_means = df[numeric_cols].mean()

df_filled = df.copy()
df_filled[numeric_cols] = df_filled[numeric_cols].fillna(
    column_means)

print("\nNull values after filling:")
print(df_filled.isnull().sum())

df_filled.to_csv('data/Furniture_filled.csv', index=False)

original_null_rows = df[df.isnull().any(axis=1)]
filled_null_rows = df_filled.loc[original_null_rows.index]
print("\nSample of filled rows (before and after):")
print("\nBefore filling:")
print(original_null_rows)
print("\nAfter filling:")
print(filled_null_rows)
```

Listing 8: Missing Value Treatment Implementation

4.2.4 Output

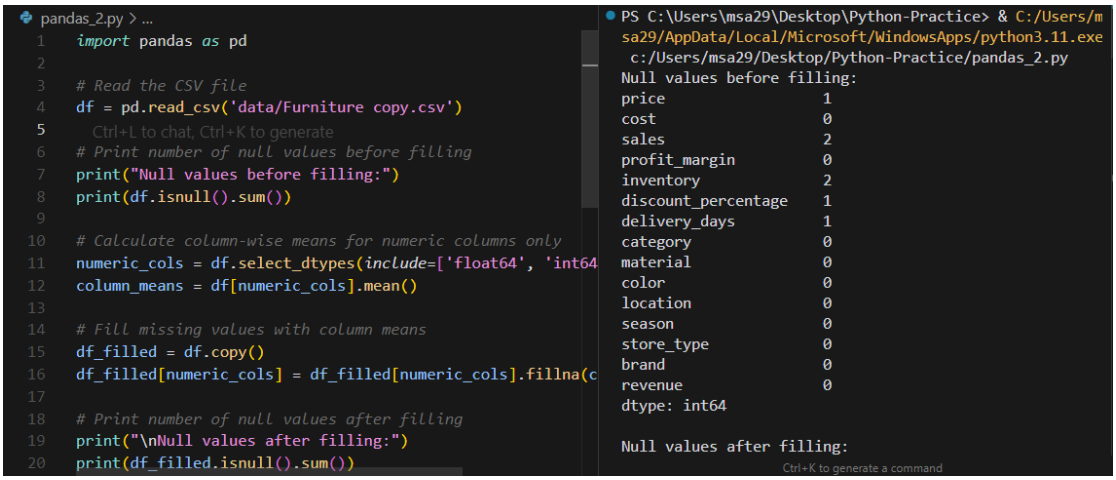

Figure 8: Missing Value Before Filling

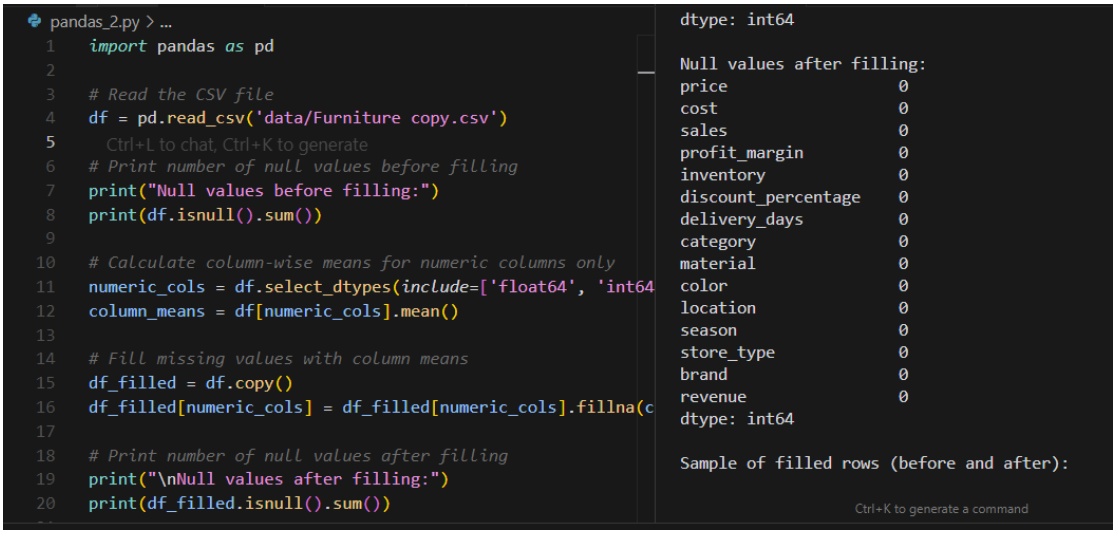

Figure 9: Missing Value after Filling

5 Data Visualization with Matplotlib

5.1 Temperature Variation Analysis

5.1.1 Objective

Visualization of weekly temperature variations using line plots, demonstrating time series data visualization.

5.1.2 Technical Details

Implementation features:

- DateTime conversion and processing
- Daily temperature aggregation
- Custom plot styling and formatting

5.1.3 Implementation

```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('data/first_week_weather.csv')

data['Date_Time'] = pd.to_datetime(data['Date_Time'])

daily_temps = data.groupby(data['Date_Time'].dt.date)['
    Temperature_C'].mean().reset_index()

daily_temps = daily_temps.sort_values('Date_Time').head(7)
```

```
11
12 plt.figure(figsize=(10, 5))
13 plt.plot(daily_temps['Date_Time'], daily_temps['Temperature_C'],
14          marker='o', linestyle='-', color='b', linewidth=2,
   markersize=8)
15
16 plt.title('Average Daily Temperature Variations Over a Week')
17 plt.xlabel('Date')
18 plt.ylabel('Temperature (\textdegree C)')
19 plt.xticks(rotation=45)
20 plt.grid(True)
21 plt.tight_layout()
22 plt.show()
```

Listing 9: Temperature Analysis Implementation

5.1.4 Output

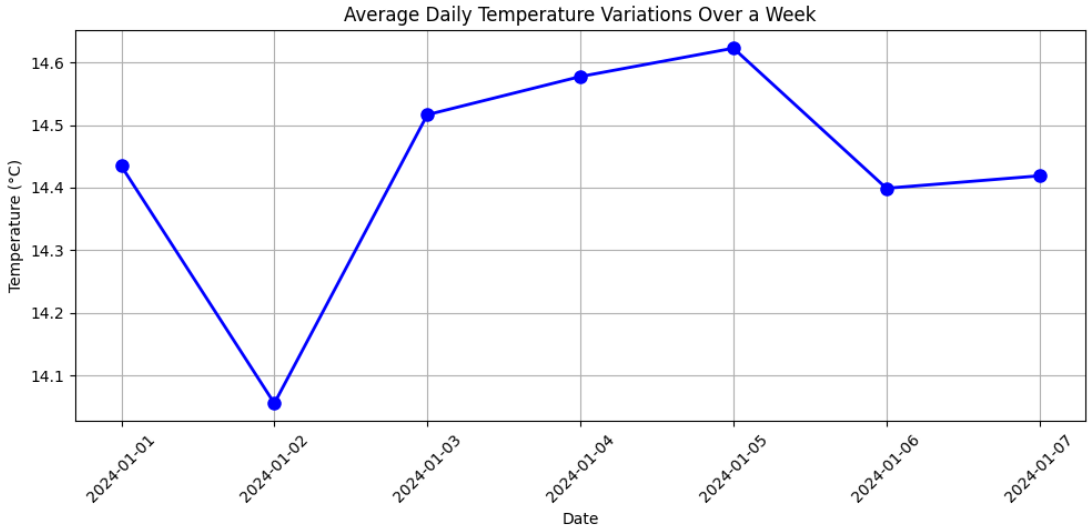

Figure 10: Line chart showing the average daily temperature variations over the first week of 2024. The plot demonstrates clear temperature fluctuations, with a notable peak on January 5th and a minimum on January 2nd. The visualization includes data points marked with blue circles and connected by lines for better trend visibility.

5.2 Regional Sales Analysis

5.2.1 Objective

Creation of bar charts for regional Mac sales analysis, demonstrating categorical data visualization.

5.2.2 Technical Details

Key features include:

- Bar chart creation with value labels
- Custom formatting and styling
- Grid lines and layout optimization

5.2.3 Implementation

```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_csv('data/apple_sales_2024.csv')

regional_sales = data.groupby('Region')['Mac Sales (in million
    units)'].mean()

plt.figure(figsize=(12, 6))
bars = plt.bar(regional_sales.index, regional_sales.values)

plt.title('Average Mac Sales by Region', fontsize=14, pad=20)
plt.xlabel('Region', fontsize=12)
plt.ylabel('Average Mac Sales (Million Units)', fontsize=12)

plt.ylim(5, 6)

plt.xticks(rotation=45)

for bar in bars:
    height = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2., height,
             f'{height:.2f}M',
             ha='center', va='bottom')

plt.grid(axis='y', linestyle='--', alpha=0.7)

plt.tight_layout()

plt.show()
```

Listing 10: Regional Sales Analysis Implementation

5.2.4 Output

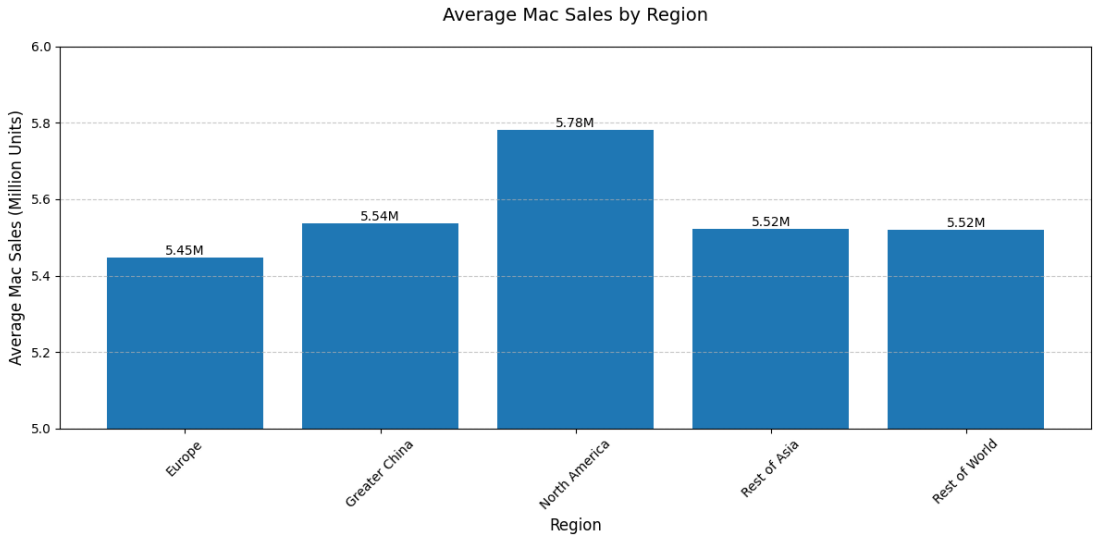

Figure 11: Bar chart comparing average Mac sales across different regions. North America shows the highest sales at 5.78M units, followed by Greater China at 5.54M units. The visualization includes value labels on top of each bar and a consistent grid pattern for easy comparison.

6 Conclusion

This report demonstrates the implementation of various Python programming concepts, from basic data structures to advanced data analysis and visualization techniques. The implementations showcase the versatility of Python and its libraries in handling different types of data processing and analysis tasks.

7 References

1. NumPy Documentation - https://numpy.org/doc/
2. Pandas Documentation - https://pandas.pydata.org/docs/
3. Matplotlib Documentation - https://matplotlib.org/stable/contents.html
4. Python Documentation - https://docs.python.org/3/
5. GeeksforGeeks - Python Lists - https://www.geeksforgeeks.org/python-ways-to-remove-d
6. W3Schools - Python Sets - https://www.w3schools.com/python/ref_set_intersection.asp