python-1

November 24, 2024

1 Name: Rohan Shrestha

2 University ID: 2418112

[3]:
```
# List of temperature measured at Tribhuwan International Airport
temperatures = [
    8.2, 17.4, 14.1, 7.9, 18.0, 13.5, 9.0, 17.8, 13.0, 8.5,
    16.5, 12.9, 7.7, 17.2, 13.3, 8.4, 16.7, 14.0, 9.5, 18.3, 13.4, 8.1,
    17.9, 14.2, 7.6, 17.0, 12.8, 8.0, 16.8, 13.7, 7.8, 17.5, 13.6, 8.7,
    17.1, 13.8, 9.2, 18.1, 13.9, 8.3, 16.4, 12.7, 8.9, 18.2, 13.1, 7.8,
    16.6, 12.5
]

# Create empty lists for classifications
cold = []
mild = []
comfortable = []

# Iterate over the temperatures list and add each temperature to the␣
 ↪appropriate category
for temp in temperatures:
    if temp < 10:
        cold.append(temp)
    elif 10 <= temp <= 15:
        mild.append(temp)
    elif 15 < temp <= 20:
        comfortable.append(temp)

# Print the lists to verify the classifications
print("Cold temperatures (<10°C):", cold)
print("Mild temperatures (10°C - 15°C):", mild)
print("Comfortable temperatures (15°C - 20°C):", comfortable)
```

```
Cold temperatures (<10°C): [8.2, 7.9, 9.0, 8.5, 7.7, 8.4, 9.5, 8.1, 7.6, 8.0,
7.8, 8.7, 9.2, 8.3, 8.9, 7.8]
Mild temperatures (10°C - 15°C): [14.1, 13.5, 13.0, 12.9, 13.3, 14.0, 13.4,
14.2, 12.8, 13.7, 13.6, 13.8, 13.9, 12.7, 13.1, 12.5]
```

```
Comfortable temperatures (15°C - 20°C): [17.4, 18.0, 17.8, 16.5, 17.2, 16.7,
18.3, 17.9, 17.0, 16.8, 17.5, 17.1, 18.1, 16.4, 18.2, 16.6]
```

[5]:
```python
# Count the number of times each temperature classification occurred
cold_count = len(cold)
mild_count = len(mild)
comfortable_count = len(comfortable)

# Print the results
print("Number of cold days (<10°C):", cold_count)
print("Number of mild days (10°C - 15°C):", mild_count)
print("Number of comfortable days (15°C - 20°C):", comfortable_count)
```

```
Number of cold days (<10°C): 16
Number of mild days (10°C - 15°C): 16
Number of comfortable days (15°C - 20°C): 16
```

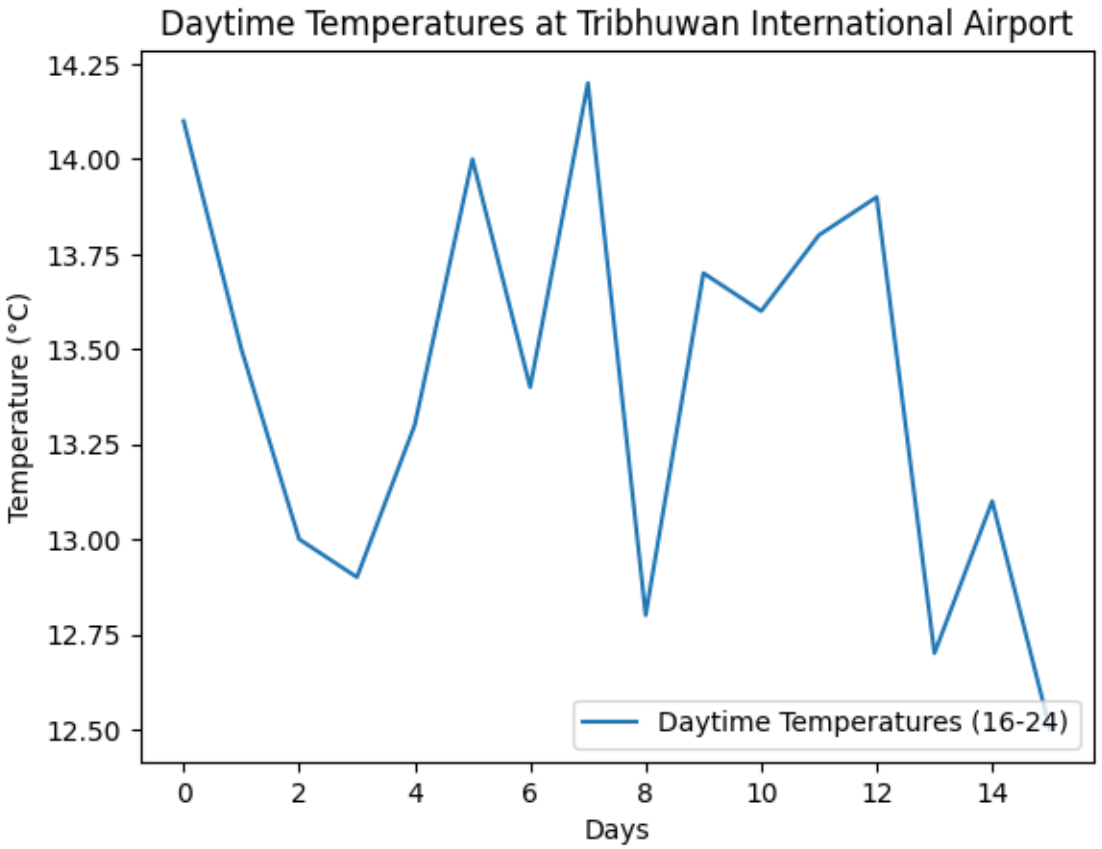
[6]:
```python
# Convert each temperature from Celsius to Fahrenheit
temperatures_fahrenheit = [(temp * 9 / 5) + 32 for temp in temperatures]

# Print the converted temperatures
print("Temperatures in Fahrenheit:", temperatures_fahrenheit)
```

```
Temperatures in Fahrenheit: [46.76, 63.32, 57.379999999999995, 46.22, 64.4,
56.3, 48.2, 64.04, 55.4, 47.3, 61.7, 55.22, 45.86, 62.959999999999994, 55.94,
47.120000000000005, 62.059999999999995, 57.2, 49.1, 64.94, 56.120000000000005,
46.58, 64.22, 57.56, 45.68, 62.6, 55.04, 46.4, 62.24, 56.66, 46.04, 63.5, 56.48,
47.66, 62.78, 56.84, 48.56, 64.58, 57.02, 46.94, 61.519999999999996, 54.86,
48.02, 64.75999999999999, 55.58, 46.04, 61.88, 54.5]
```

[7]:
```python
# Create empty lists for night, day, and evening temperatures
night_temps = []
evening_temps = []
day_temps = []

# Iterate over the temperatures list, assigning values to each time-of-day list␣
 ↪based on
#their position
for i in range(0, len(temperatures), 3):
    night_temps.append(temperatures[i])
    evening_temps.append(temperatures[i + 1])
    day_temps.append(temperatures[i + 2])

# Calculate the average daytime temperature
average_day_temp = sum(day_temps) / len(day_temps)

# Print the average day-time temperature
print("Average Day-time Temperature:", average_day_temp)
```

```
Average Day-time Temperature: 13.40625
```

[29]:
```python
import matplotlib.pyplot as plt

# Plot Day vs Temperature
plt.xlabel('Days')
plt.ylabel('Temperature (°C)')
plt.title('Daytime Temperatures at Tribhuwan International Airport')
plt.plot(day_temps, label="Daytime Temperatures (16-24)")
plt.legend(loc="lower right")
plt.show()
```



Exercise - Recursion

Task 1: Sum of Nested Lists

[9]:
```python
def sum_nested_list(nested_list):
    total = 0
    for element in nested_list:
        if isinstance(element, list):
            total += sum_nested_list(element)
```

```
        else:
            total += element
    return total

# Test the function
nested_list = [1, [2, [3, 4], 5], 6, [7, 8]]
print("Nested List Sum: ",sum_nested_list(nested_list))
```

```
Nested List Sum:  36
```

Task 2: Generate All Permutations of a String

[10]:
```
def generate_permutations(s):
    if len(s) == 0:
        return ['']
    perms = []
    for i in range(len(s)):
        char = s[i]
        remaining = s[:i] + s[i+1:]
        for p in generate_permutations(remaining):
            perms.append(char + p)
    return perms

# Test the function
print("Permutation: ",generate_permutations("abc"))
```

```
Permutation:  ['abc', 'acb', 'bac', 'bca', 'cab', 'cba']
```

Task 3: Directory Size Calculation

[12]:
```
def calculate_directory_size(directory):
    total_size = 0
    for item in directory:
        if isinstance(directory[item], dict):
            total_size += calculate_directory_size(directory[item])
        else:
            total_size += directory[item]
    return total_size

directory_structure = {
    "file1.txt": 200,
    "file2.txt": 300,
    "subdir1": {
        "file3.txt": 400,
        "file4.txt": 100
    },
    "subdir2": {
        "subsubdir1": {
            "file5.txt": 250
```

```
        },
        "file6.txt": 150
    }
}

# Test the function
print("Directory Size: ",calculate_directory_size(directory_structure))
```

```
Directory Size:  1400
```

3 Exercises - Dynamic Programming

Task 1: Coin Change Problem

[14]:
```
def min_coins(coins, amount):
    dp = [float('inf')] * (amount + 1)
    dp[0] = 0
    for coin in coins:
        for i in range(coin, amount + 1):
            dp[i] = min(dp[i], dp[i - coin] + 1)
    return dp[amount] if dp[amount] != float('inf') else -1

# Test the function
print(min_coins([1, 2, 5], 11))
```

```
3
```

Task 2: Longest Common Subsequence (LCS)

[15]:
```
def longest_common_subsequence(s1, s2):
    m, n = len(s1), len(s2)
    dp = [[0] * (n + 1) for _ in range(m + 1)]
    for i in range(1, m + 1):
        for j in range(1, n + 1):
            if s1[i - 1] == s2[j - 1]:
                dp[i][j] = dp[i - 1][j - 1] + 1
            else:
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1])
    return dp[m][n]

# Test the function
print(longest_common_subsequence("abcde", "ace"))  # Output: 3
```

```
3
```

Task 3: 0/1 Knapsack Problem

[16]:
```python
def knapsack(weights, values, capacity):
    n = len(weights)
    dp = [[0] * (capacity + 1) for _ in range(n + 1)]
    for i in range(1, n + 1):
        for w in range(capacity + 1):
            if weights[i - 1] <= w:
                dp[i][w] = max(dp[i - 1][w], dp[i - 1][w - weights[i - 1]] + values[i - 1])
            else:
                dp[i][w] = dp[i - 1][w]
    return dp[n][capacity]

# Test the function
print(knapsack([1, 3, 4, 5], [1, 4, 5, 7], 7))  # Output: 9
```

```
9
```

[ ]: