To implement control structures with a climate dataset using Python to filter, manipulate, and analyze the data. ¶

Objectives:

1. Apply Conditional Statements: Use if-else to filter climate data based on conditions like temperature or CO₂ levels.
2. Use Loops for Data: Implement loops to process and analyze climate data, such as calculating averages or trends.
3. Filter and Manipulate Data: Use filter() and list comprehensions to filter and modify climate datasets.
4. Analyze Data with Control Structures: Identify patterns, such as the city with the highest carbon footprint, using control structures.

1: Apply Conditional Statements to Filter High Temperature Cities

In [1]:
```
# Sample climate dataset
climate_data = [
    {"city": "City A", "temperature": 25, "carbon_footprint": 500},
    {"city": "City B", "temperature": 30, "carbon_footprint": 350},
    {"city": "City C", "temperature": 22, "carbon_footprint": 600},
    {"city": "City D", "temperature": 15, "carbon_footprint": 200},
    {"city": "City E", "temperature": 28, "carbon_footprint": 450},
]

# High temperature threshold (cities above this temperature will be selected
high_temp_threshold = 26

# Use a list comprehension to filter cities with temperatures higher than th
high_temp_cities = [city for city in climate_data if city["temperature"] > h

# Print the result
print("Cities with high temperatures (> 26°C):")
for city in high_temp_cities:
    print(f"{city['city']} - {city['temperature']}°C")
```

```
Cities with high temperatures (> 26°C):
City B - 30°C
City E - 28°C
```

The above code filters cities where the temperature is above 26°C. We use a list comprehension and an if statement to check each city's temperature.

2: Use Loops to Calculate Average Carbon Footprint

In [2]:
```
# We will continue using the same dataset as above
# Initialize a variable to store the total carbon footprint
total_carbon = 0

# Use a for loop to go through each city in the dataset and add their carbon
for city in climate_data:
    total_carbon += city["carbon_footprint"]  # Add each city's carbon footp

# Calculate the average carbon footprint
average_carbon_footprint = total_carbon / len(climate_data)

# Print the average result
print(f"\nAverage Carbon Footprint: {average_carbon_footprint:.2f} kg CO2")
```

```
Average Carbon Footprint: 420.00 kg CO2
```

The above python code calculates the average carbon footprint by summing all the footprints and dividing by the number of cities. A for loop helps us add up the values.

3: Filter and Manipulate Data to Find Sustainable Cities

In [3]:
```
# Set a sustainability threshold (e.g., 400 kg CO2)
sustainability_threshold = 400

# Use a lambda function with filter() to get cities with a carbon footprint
sustainable_cities = list(filter(lambda city: city["carbon_footprint"] < sus

# Print the result
print("\nSustainable Cities (carbon footprint < 400 kg CO2):")
for city in sustainable_cities:
    print(f"{city['city']} - {city['carbon_footprint']} kg CO2")
```

```
Sustainable Cities (carbon footprint < 400 kg CO2):
City B - 350 kg CO2
City D - 200 kg CO2
```

The above python code filters cities with a carbon footprint below 400 kg CO_2 using a lambda function and the filter() method. Only cities that meet the condition are shown.

4: Analyze Data to Find the City with the Highest Carbon Footprint

In [4]:
```
# Use max() to find the city with the highest carbon footprint
highest_footprint_city = max(climate_data, key=lambda city: city["carbon_foo

# Print the result
print(f"\nCity with the highest carbon footprint:")
print(f"{highest_footprint_city['city']} - {highest_footprint_city['carbon_f
```

```
City with the highest carbon footprint:
City C - 600 kg CO2
```

The above python code finds the city with the highest carbon footprint using the max() function, which compares each city's footprint.

Conclusion:

The above codes are in Python that addresses the objectives as laid below, using control structures to filter, manipulate, and analyze a climate dataset

Objective 1: Filters cities based on temperature using conditional logic.

Objective 2: Uses a loop to calculate the average carbon footprint.

Objective 3: Filters cities below a carbon footprint threshold using lambda.

Objective 4: Finds the city with the highest carbon footprint using max().