Tutorial VII - NumPy and Pandas for Scientific Computing

Programming with Python

Analyzing Climate Data

Imagine you're a climate scientist working on a project to analyze temperature data from weather stations across the country. You've been given a large dataset, and you need to use NumPy to process and analyze this data efficiently.

To solve this task, you'll need to use NumPy to perform various operations on the temperature data. Here, you'll need some new functions you haven't seen yet.

- `np.mean()`: calculate the mean of an array
- `np.max()`: calculate the maximum of an array
- `np.min()`: calculate the minimum of an array
- `np.argmax()`: find the index of the maximum of an array
- `np.argmin()`: find the index of the minimum of an array

If you want to find those informations along an axis, you can use the same function with `axis` as an additional argument. Here, `axis=0` is for the columns and `axis=1` is for the rows.

For example, to find the maximum of each row, you can use `np.max(data_set, axis=1)`. To find the index of the maximum of each column, you can use `np.argmax(data_set, axis=0)`.

```
import numpy as np

# Let's analyze temperature data from 10 weather stations over 30 days
temp_data = np.random.randint(0, 40, size=(10, 30))

# Example: Calculate the average temperature for the first station
first_station_avg = np.mean(temp_data[0,:])
print(f"Average temperature for the first station: {first_station_avg:.2f}°C")

# TODO: a) Calculate the average temperature for each station and print it. Make sure to
 ↪  round the result to 2 decimal places!
# Hint: Use np.mean() with axis=1
# Your code here


# TODO: b) Find the highest temperature recorded and the station index and print it
# Hint: Use np.max() and np.argmax()
# Your code here

# TODO: c) Find the lowest temperature recorded and the station index and print it
# Hint: Use np.min()
# Your code here

# TODO: d) Calculate the overall average temperature and print it
# Hint: Use np.mean() on the entire temp_data array
```

```
# Your code here

# Example: Identify days above 30°C for the first station
hot_days = np.sum(temp_data[0,:] > 30)
print(f"The first station had {hot_days} days above 30°C")

# TODO: e) Count the number of days above 30°C for each station
# Your code here

# TODO: f) Find the hottest and coldest stations and determine the index of the station
 ↪  with the highest average temperature and the station with the lowest average
 ↪  temperature.
# Hint: Use np.argmax() and np.argmin()
# Your code here
```

```
Average temperature for the first station: 20.47°C
The first station had 5 days above 30°C
```

NASA GISTEMP Climate Change Analysis

In this exercise, you'll use Pandas to analyze real global temperature anomaly data from NASA, helping to understand trends in climate change over time.

The dataset is provided by the GISS Team, 2024: GISS Surface Temperature Analysis (GISTEMP), version 4. NASA Goddard Institute for Space Studies. Dataset at https://data.giss.nasa.gov/gistemp/.

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

# First, we load the NASA GISTEMP dataset for global temperature anomalies.
url = "https://data.giss.nasa.gov/gistemp/tabledata_v4/GLB.Ts+dSST.csv"
temp_anomaly_data = pd.read_csv(url, skiprows=1) # skiprows=1 ensures that the first
 ↪  column is not read as a row index

# Convert all numeric columns (except 'Year') to float, replacing '***' with NaN
numeric_columns = temp_anomaly_data.columns.drop('Year')
temp_anomaly_data[numeric_columns] = temp_anomaly_data[numeric_columns].replace('***',
 ↪  float('nan')).astype(float)

# TODO: a) Display the first 5 rows to learn basic information about the DataFrame. For
 ↪  your work, you only need the 'Year' and and the data from all months. Drop the rest
 ↪  of the columns.
# Your code here

# TODO: b) Calculate and print the average temperature anomaly for each year.
# Hint: To do so, you first need to pd.melt()` the DataFrame to convert months to a
 ↪  single column.
# Your code here

# TODO: c) Find the year with the highest temperature anomaly and the year with the
 ↪  lowest.
# Hint: Use the `idxmax()` and `idxmin()` methods
# Your code here

# TODO: e) Save the melted DataFrame to a Excel file with the name
 ↪  'temp_anomaly_data.xlsx' for next lecture.
# Your code here
```

That’s it!

You can find the solutions to these exercises online in the associated GitHub repository, but we will also quickly go over them in next week’s tutorial. To access the solutions, click on the Github button on the lower right and search for the folder with today’s lecture and tutorial. Alternatively, you can ask ChatGPT or Claude to explain them to you. Remember, the goal is not just to complete the exercises, but to understand the concepts and improve your programming abilities.