Lecture VII - NumPy and Pandas for Scientific Computing Programming with Python

Dr. Tobias Vlćek

Quick Recap of the last Lecture

NumPy Module

- NumPy is a fundamental package for scientific computing in Python
- It provides support for large, multi-dimensional arrays and matrices
- It also provides a wide range of mathematical functions to operate on these arrays
- · Much faster than standard Python lists

Creating Arrays

- Use the np.array(list) function to create an array from a list
- Use the np.zeros(shape) function to create an array of zeros
- Use the np.ones(shape) function to create an array of ones
- Use the np.arange(start, stop, step) function to create an array of evenly spaced values
- Use the np.linspace(start, stop, num) function to create an array of evenly spaced values
- Use the np.random.rand(shape) function to create an array of random values
- Use the np.random.randint(low, high, size) function to create an array of random integers

Pandas Module

- Pandas is a powerful data manipulation and analysis library
- It provides data structures like DataFrames and Series for data manipulation
- · It also provides tools for data cleaning, analysis, and visualization

Creating DataFrames

- Use the pd.DataFrame(data, index, columns) function to create a DataFrame from a dictionary
- Use the pd.DataFrame(data, index, columns) function to create a DataFrame from a dictionary
- Use the pd.DataFrame(data, index, columns) function to create a DataFrame from a dictionary

Basic Operations

- Use the df.head() method to display the first few rows of a DataFrame
- Use the df.tail() method to display the last few rows of a DataFrame
- Use the df.info() method to display information about a DataFrame
- Use the df.describe() method to display summary statistics about a DataFrame
- Use the df.columns attribute to access the column names of a DataFrame
- Use the df.index attribute to access the index of a DataFrame
- Use the df.values attribute to access the values of a DataFrame

Subsetting DataFrames

- Use the df.loc[row_indexer, column_indexer] method to access a specific element of a DataFrame
- Use the df.iloc[row_indexer, column_indexer] method to access a specific element of a DataFrame

Filtering DataFrames

- Use the df [df ['column'] > value] method to filter a DataFrame
- Use the df [df ['column'].isin(values)] method to filter a DataFrame

Grouping DataFrames

• Use the df.groupby('column').sum() method to group a DataFrame and calculate the sum of a column

- Use the df.groupby('column').mean() method to group a DataFrame and calculate the mean of a column
- Use the df.groupby('column').count() method to group a DataFrame and count the number of elements in a column
- Use the df.groupby('column').size() method to group a DataFrame and count the number of elements in a column

Excel Files

- Excel files can be read using the pd.read_excel(file_path) function
- Excel files can be written using the df.to_excel(file_path) method