Introduction to Programming with Python

Dr. Anatol Wegner

Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science (CAIDAS)
Julius-Maximilians-Universität Würzburg
Würzburg, Germany

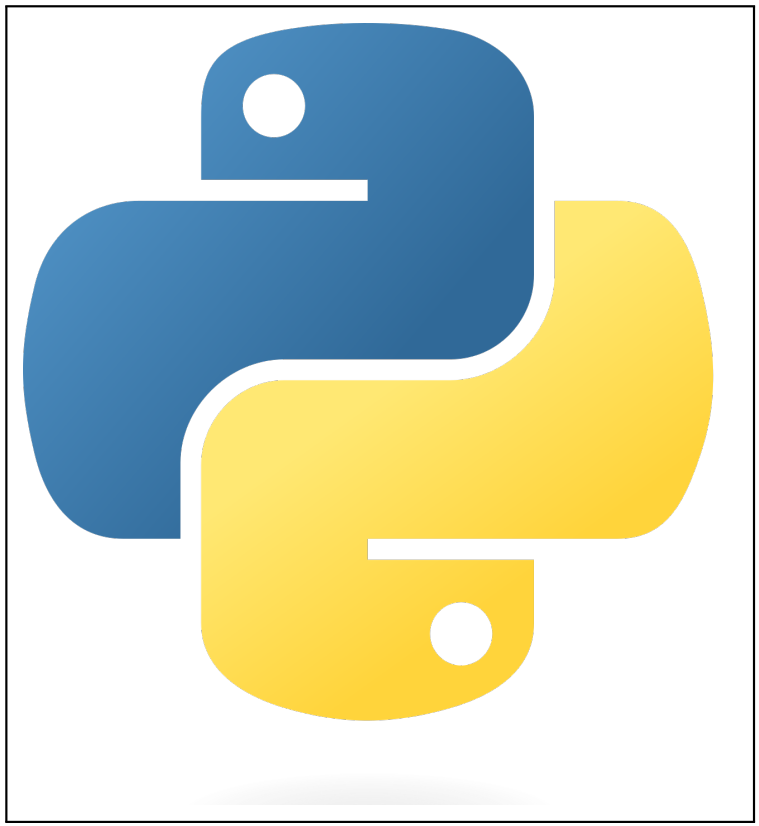
anatol.wegner@uni-wuerzburg.de

Lecture 05
Data analysis with Python

November 29, 2024

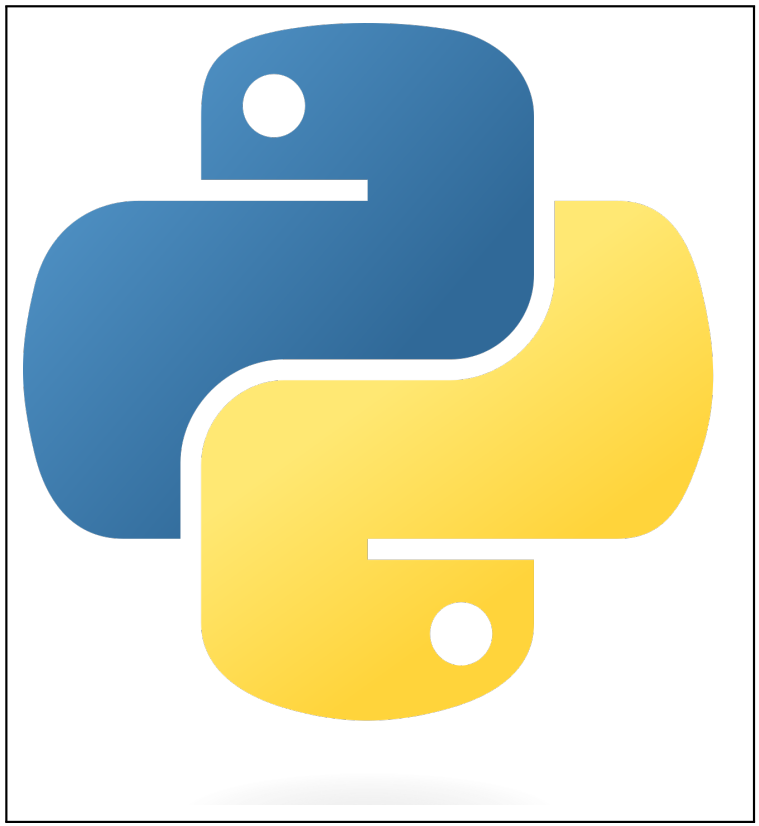
Recap

- we introduced modules in Python
- we learned how to import modules
- In-built modules (math, random, os ...)
- Creating our own modules
- Installing 3rd party libraries
- Creating Python environments
- Opening files in 'r', 'w', 'a' and 'b'
- Reading files and lines
- Writing and appending to files



Recap

- we introduced modules in Python
- we learned how to import modules
- In-built modules (math, random, os ...)
- Creating our own modules
- Installing 3rd party libraries
- Creating Python environments
- Opening files in 'r', 'w', 'a' and 'b'
- Reading files and lines
- Writing and appending to files



Today

- Data analysis (pandas)
- Data visualization (seaborn)

Key Libraries for Data Analysis in Python

- `pandas` - Data manipulation and analysis. (pandas.pydata.org)
- `seaborn`- Statistical data visualization. (seaborn.pydata.org)
- `numpy` - Numerical computations. (numpy.org)
- `matplotlib` - Data visualization with plots. (matplotlib.org)

Introduction to `pandas`

Key Functionalities:

- Data manipulation and analysis
- Handles structured data like CSVs, Excel files, and SQL queries
- Tools for cleaning, transforming, and summarizing data

Core Data Structures:

- `DataFrame`: 2D labeled table, similar to a spreadsheet
- `Series`: 1D labeled array, representing a single column or row

Learn more: pandas.pydata.org

Loading and Saving Data with `pandas`

Loading Data:

- `pd.read_csv('file.csv')`: Reads a CSV file into a DataFrame.
- `pd.read_excel('file.xlsx')`: Reads an Excel file.
- `pd.read_sql(query, connection)`: Reads data from a SQL database.
- `pd.read_json('file.json')`: Reads JSON data into a DataFrame.

Example:

```
# Load CSV file
data = pd.read_csv('file.csv')

# Save as Excel
data.to_excel('file.xlsx', index=False)
```

Loading and Saving Data with `pandas`

Saving Data:

- `data.to_csv('file.csv')`: Saves the DataFrame as a CSV file.
- `data.to_excel('file.xlsx')`: Exports the DataFrame to an Excel file.
- `data.to_json('file.json')`: Saves the DataFrame in JSON format.
- `data.to_sql('table_name', connection)`: Writes the DataFrame to a SQL table.

Example:

```
# Load CSV file
data = pd.read_csv('file.csv')

# Save as Excel
data.to_excel('file.xlsx', index=False)
```

Inspecting Data with `pandas`

Basic Inspection Methods:

- `data.head(n)`: Displays the first `n` rows of the DataFrame (default is 5).
- `data.tail(n)`: Displays the last `n` rows of the DataFrame.
- `data.info()`: Provides a summary of the DataFrame, including column types, counts, and memory usage.
- `data.describe()`: Calculates summary statistics for numeric columns.
- `data.shape`: Returns the dimensions of the DataFrame (`rows`, `columns`).

Inspecting Data with `pandas`

Example:

```
# Inspect the data
print(data.head())
print(data.info())
print(data.describe())

# Check the shape
print(f"Rows: {data.shape[0]}, Columns: {data.shape[1]}")
```

Usage:

- `head()` and `tail()`: Quick look at the beginning or end of the dataset.
- `info()`: Understand data types and missing values.
- `describe()`: Get key statistics like mean, median, and standard deviation.

Understanding Indices and Slicing

Code Example:

```
# Access a single column
column = data['column_name']

# Access rows by index
row = data.iloc[0]

# Slice rows
subset = data[10:20]
```

Concepts:

- **Indices**: Every row and column in a DataFrame has an index.
- **Slicing**: Use square brackets to slice rows or select columns.
- `data.iloc`: Access rows by their integer index position.
- `data['column_name']`: Access a column as a Series.

Basic Statistical Functions in `pandas`

Summary Statistics:

- Mean of a column:

```
mean_value = data['col'].mean()
```

- Standard deviation:

```
std_dev = data['col'].std()
```

- Median:

```
median_value = data['col'].median()
```

- Maximum and minimum values:

```
max_value = data['col'].max()
min_value = data['col'].min()
```

Basic Statistical Functions in `pandas`

Aggregations:

- Sum of values:

```
total = data['col'].sum()
```

- Count of non-null values:

```
count = data['col'].count()
```

Other Useful Functions:

- Correlation:

```
correlation = data.corr()
```

- Unique values in a column:

```
unique_values = data['col'].unique()
```

- Value counts (frequencies):

```
value_counts = data['col'].value_counts()
```

Modifying Data with `pandas`

Adding and Updating Columns:

- Add a new column:

```
data['new_col'] = data['col1'] + data['col2']
```

- Update values conditionally:

```
data.loc[data['col1'] > 10, 'col1'] = 10
```

Dropping Rows and Columns:

- Drop a column:

```
data = data.drop('col_name', axis=1)
```

- Drop rows by index:

```
data = data.drop(index=[0, 1])
```

Renaming Columns:

- `data = data.rename(columns={'old_name': 'new_name'})`

Data Preprocessing with `pandas`

Handling Missing Data:

- Drop rows with missing values:

```
data = data.dropna()
```

- Fill missing values:

```
data['col'] = data['col'].fillna(value)
```

Data Transformation:

- Apply a function to a column:

```
data['col'] = data['col'].apply(myfunction)
```

- Normalize numeric columns:

```
(data['col'] - data['col'].mean()) / data['col'].std()
```

Handling Duplicates:

- Remove duplicate rows:

```
data = data.drop_duplicates()
```

`groupby()` in Pandas

- `groupby()` is a powerful function in Pandas used for grouping data based on one or more columns.
- It is often used in combination with aggregation functions (e.g., `mean()`, `sum()`, `count()`).

How Does It Work?

- Splits the data into groups based on the values of a column(s).
- Applies a function (e.g., aggregation or transformation) to each group.
- Combines the results into a new DataFrame or Series.

Example: Calculate Mean Fare by Passenger Class (Titanic Dataset)

```
# Group by 'Pclass' and calculate mean fare
grouped_data = data.groupby('Pclass')['fare'].mean()
```

Output:

```
Pclass
1    84.154687
2    20.662183
3    13.675550
```

Summary of `pandas`

Core Features Covered:

- Loading data using `read_csv`
- Inspecting data with `head()` and `info()`
- Working with indices and slicing
- Adding and modifying columns

Explore further at: pandas.pydata.org

Summary of `pandas`

Core Features Covered:

- Loading data using `read_csv`
- Inspecting data with `head()` and `info()`
- Working with indices and slicing
- Adding and modifying columns

Explore further at: pandas.pydata.org

Practice Session 1

- Data analysis with `pandas`

`https://gitlab2.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_wise_infhaf_notebooks/-/blob/main/PythonIntroNotebooks/Lecture_05.ipynb`

Introduction to Seaborn

Seaborn is a Python visualization library based on Matplotlib that provides a high-level interface for creating attractive and informative statistical graphics.

- Simplifies the creation of complex visualizations.
- Built-in support for Pandas DataFrames.
- Comes with attractive default themes and color palettes.
- Makes it easy to visualize distributions, relationships, and categorical data.

Website: `https://seaborn.pydata.org/`

Core Data Structures in Seaborn

Seaborn works well with Pandas DataFrames. You can pass a DataFrame directly to most Seaborn functions.
Key Data Structures:

- **Pandas DataFrame**: Seaborn handles DataFrames directly, making it easy to plot data.
- **Arrays**: Seaborn also supports plotting from NumPy arrays or Python lists, though DataFrames are preferred.

Example:

```
import seaborn as sns
import pandas as pd

# Load the Titanic dataset
data = sns.load_dataset('titanic')
```

Simple Plot with Seaborn

Seaborn makes it easy to create basic plots. Here's how to create a simple scatter plot:

```
import seaborn as sns

# Load a dataset
data = sns.load_dataset('titanic')

# Create a scatter plot
sns.scatterplot(x='age', y='fare', data=data)
```

This code creates a scatter plot showing the relationship between 'age' and 'fare' for passengers in the Titanic dataset.

Common Plot Types in Seaborn

Seaborn simplifies common visualizations like:

- **Scatter plot**: `sns.scatterplot()`
- **Line plot**: `sns.lineplot()`
- **Bar plot**: `sns.barplot()`
- **Histogram**: `sns.histplot()`
- **Box plot**: `sns.boxplot()`
- **Heatmap**: `sns.heatmap()`

These plots allow you to explore different aspects of your data in a visually appealing way.

Customizing Plots with Seaborn

Seaborn allows for easy customization of plots:

- **Titles and labels**: Use Matplotlib functions like `plt.title()`, `plt.xlabel()`.
- **Themes**: Seaborn offers built-in themes that make your plots look better.

```
sns.set_theme(style="whitegrid")
```

- **Color palettes**: Seaborn provides several color palettes for different plot types.

```
sns.set_palette("coolwarm")
```

These options make it easier to create polished, professional-looking plots.

Plotting survival rate by passenger class on the Titanic

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load Titanic dataset
data = sns.load_dataset('titanic')

# Create a bar plot for survival rate by class
sns.barplot(x='Pclass', y='Survived', data=data, palette='vi

# Add title and labels
plt.title("Survival Rate by Passenger Class")
plt.xlabel("Passenger Class")
plt.ylabel("Survival Rate")

# Display the plot
plt.show()
```

Conclusion and Further Exploration

Seaborn is a powerful tool for visualizing data with minimal code. It simplifies complex visualizations, handles Pandas DataFrames well, and offers many useful default styles.

- Explore more plot types (e.g., violin plots, pair plots).
- Experiment with different color palettes.
- Use Seaborn for exploratory data analysis (EDA).

For more details, visit the official Seaborn documentation: `https://seaborn.pydata.org/`

Conclusion and Further Exploration

Seaborn is a powerful tool for visualizing data with minimal code. It simplifies complex visualizations, handles Pandas DataFrames well, and offers many useful default styles.

- Explore more plot types (e.g., violin plots, pair plots).
- Experiment with different color palettes.
- Use Seaborn for exploratory data analysis (EDA).

For more details, visit the official Seaborn documentation:
`https://seaborn.pydata.org/`

Practice Session 2

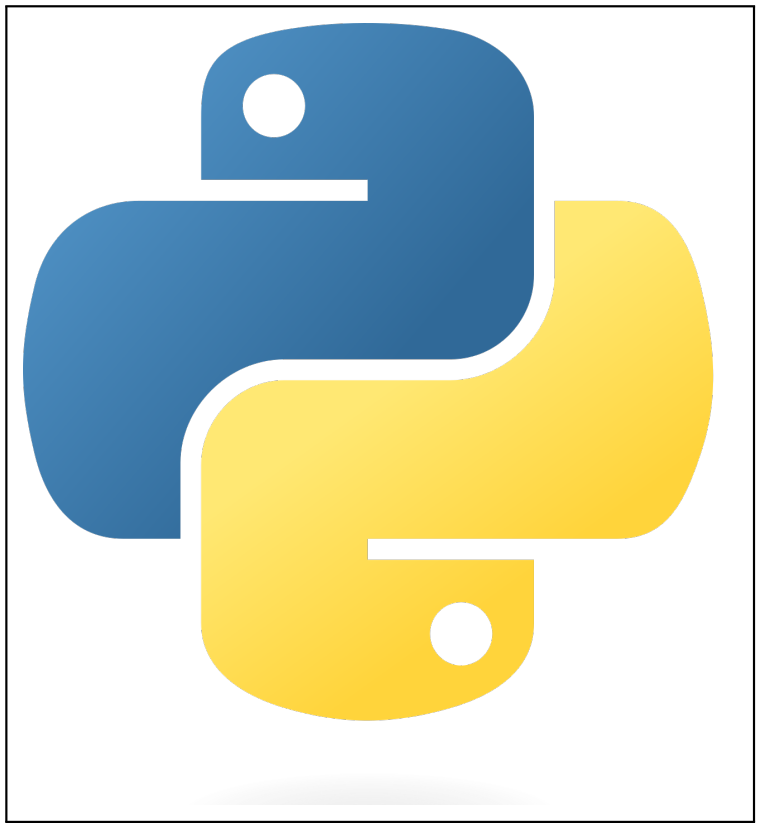
- Creating plots with `seaborn`

`https://gitlab2.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_wise_infhaf_notebooks/-/blob/main/PythonIntroNotebooks/Lecture_05.ipynb`

In summary

1. Pandas: Data Analysis and Manipulation

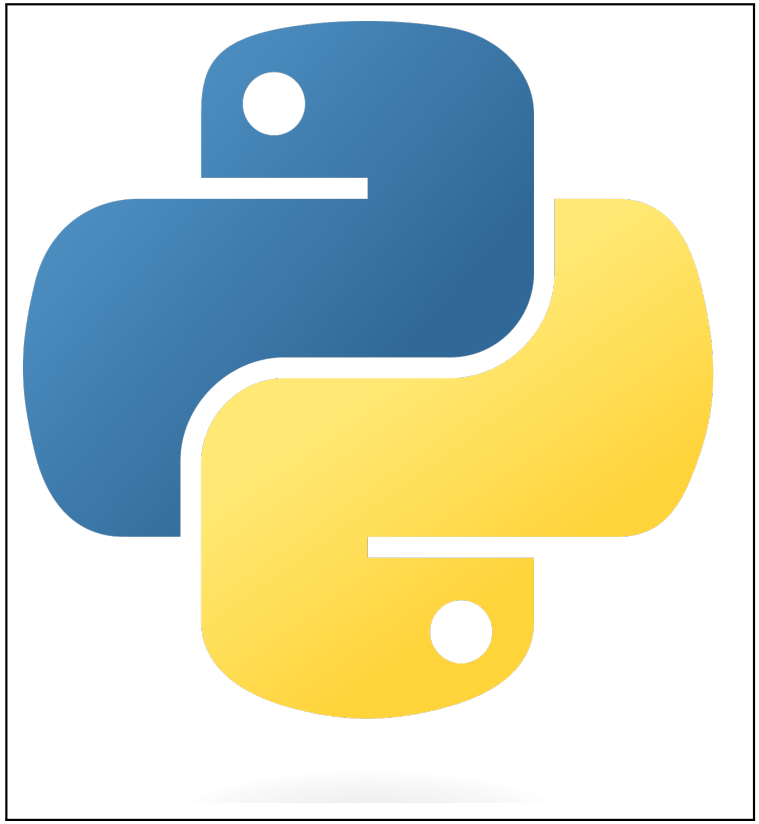
- Core Data Structures: `Series`, `DataFrame`.
- Data Import and Export: Functions like `read_csv()`, `to_csv()`.
- Data Exploration: `head()`, `info()`, `describe()`.
- Data Preprocessing: Handling missing data, filtering rows, and modifying columns.
- Grouping and Aggregation: Using `groupby()` for insights.
- Statistical Functions: `mean()`, `median()`, `std()`.



In summary

2. Seaborn: Data Visualization

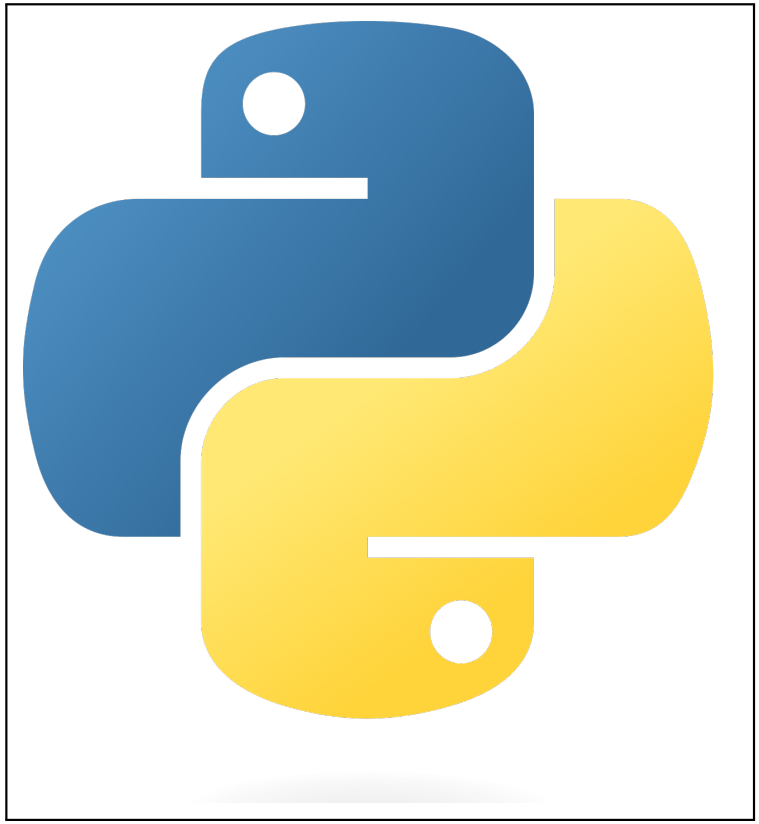
- High-Level Interface: Simplified plotting for statistical graphics.
- Common Plot Types: Scatter plots, bar plots, histograms, box plots.
- Customization: Themes, color palettes, and labels.
- Integration with Pandas: Directly using DataFrames for visualization.



In summary

2. Seaborn: Data Visualization

- High-Level Interface: Simplified plotting for statistical graphics.
- Common Plot Types: Scatter plots, bar plots, histograms, box plots.
- Customization: Themes, color palettes, and labels.
- Integration with Pandas: Directly using DataFrames for visualization.



Exercise Session

- Data analysis and visualization

`https://gitlab2.informatik.uni-wuerzburg.de/ml4nets_notebooks/2024_wise_infhaf_notebooks/-/blob/main/PythonIntroNotebooks/Exercise_L05.ipynb`

Self-Study Questions: Pandas and Seaborn

1. **What is the difference between a Series and a DataFrame in Pandas?**
2. **How do you handle missing data in a DataFrame (e.g., filling or dropping missing values)?**
3. **What is the purpose of the `groupby()` function in Pandas? Provide an example.**
4. **How can you filter rows in a DataFrame based on a condition (e.g., passengers over 30 years old)?**
5. **What is the role of Seaborn in data visualization? How does it simplify plotting?**
6. **How do you create a scatter plot using Seaborn? Provide an example.**
7. **How would you create a bar plot in Seaborn to show the average survival rate by passenger class?**
8. **How can you customize a Seaborn plot by changing its color palette or theme?**