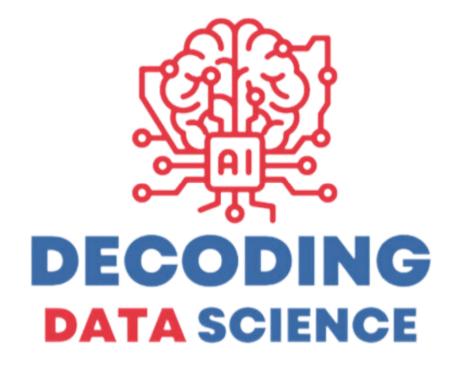


ADVANCED PYTHON

JOIN OUR AI COMMUNITY: HTTPS://NAS.IO/ARTIFICIALINTELLIGENCE

NumPy Cheat Sheet DECODING DATA SCIENCE



JOIN OUR AI COMMUNITY: HTTPS://NAS.IO/ARTIFICIALINTELLIGENCE

1. Basic Commands

Importing NumPy and checking its version:

```python

import numpy as np

print(np.\_\_version\_\_)

• • •

## 2. Array Creation

Creating NumPy arrays from lists and with initial placeholders:

```
``python
 # From a list
arr = np.array([1, 2, 3, 4, 5])

Array of zeros
arr = np.zeros((3, 3))

Array of ones
arr = np.ones((3, 3))

Array with a range of values
arr = np.arange(0, 10)

Array of random values
arr = np.random.rand(3, 3)
```

• • •

## 3. Array Attributes

Getting an array's shape and data type:

```
```python
arr = np.array([[1, 2, 3], [4, 5, 6]])
     # Shape
     print(arr.shape)
     # Data type
     print(arr.dtype)
```

4. Indexing and Slicing

Indexing and slicing one-dimensional and multi-dimensional arrays:

```python
arr = np.array([1, 2, 3, 4, 5])
# Get the first element
 print(arr[0])
# Get the last element

# Get the last element print(arr[-1])

# Get a slice from the second to the fourth element print(arr[1:4])

• • •

## 5. Array Manipulation

Various ways to manipulate arrays such as reshaping, stacking, and splitting:

```python
arr = np.array([[1, 2, 3], [4, 5, 6]])
Reshape
arr_reshaped = arr.reshape((3, 2))

Vertical stack

arr_stack = np.vstack([arr, arr])

6. Arithmetic Operations

Performing addition, subtraction, multiplication, division, and dot product on arrays:

```
```python
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
 # Addition print
 (arr1 + arr2)
 # Subtraction print
 (arr1 - arr2)
 # Multiplication print
 (arr1 * arr2)
 # Division print
 (arr1 / arr2)
```

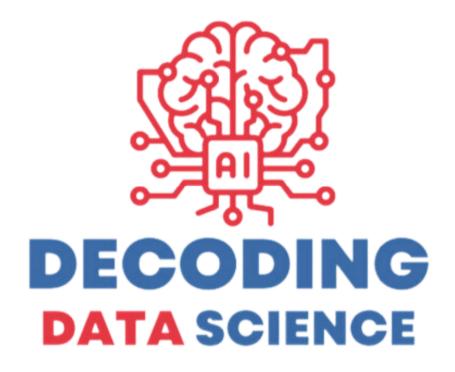
### 7. Statistical Operations

Calculating the mean, median, and standard deviation of an array:

```
```python
arr = np.array([1, 2, 3, 4, 5])
         # Mean
    print(np.mean(arr))
         # Median
    print(np.median(arr))

# Standard deviation
    print(np.std(arr))
```

Matplotlib Cheat Sheet DECODING DATA SCIENCE



JOIN OUR AI COMMUNITY: HTTPS://NAS.IO/ARTIFICIALINTELLIGENCE

1. Basic Commands

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

- Importing Matplotlib:

import matplotlib.pyplot as plt

- Checking Matplotlib version:

import matplotlib

print(matplotlib.__version__)

2. Basic Plotting

Matplotlib provides functionalities for various types of plots.

- Line Plot: plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
- Scatter Plot: plt.scatter([1, 2, 3, 4], [1, 4, 9, 16])
- Bar Plot: plt.bar(['group_a', 'group_b', 'group_c'], [1, 10, 5])
 - Histogram: plt.hist([1, 2, 2, 3, 4, 4, 4, 5, 5, 5, 5])

3. Figure and Axes

A figure in matplotlib means the whole window in the user interface. Axis are the number-line-like objects and they take care of generating the graph limits.

- Creating Figure and Axes: fig, ax = plt.subplots()
- Setting Figure Size: fig, ax = plt.subplots(figsize=(6, 4))
 - Setting Axis Labels and Title:
 - ax.set_xlabel('x')
 - ax.set_ylabel('y')
 - ax.set_title('Title')

4. Customizing Plots

Matplotlib allows you to customize various aspects of your plots.

- Changing Line Style and Color:

plt.plot([1, 2, 3, 4], [1, 4, 9, 16], linestyle='--', color='r')

- Adding Grid:

plt.grid(True)

- Setting Axis Limits:

plt.xlim(0, 5)

plt.ylim(0, 20)

5. Multiple Plots

Matplotlib provides functionalities to create multiple plots in a single figure.

- Subplots:

fig, axs = plt.subplots(2)

- Sharing Axis:

fig, axs = plt.subplots(2, sharex=True, sharey=True)

6. Text and Annotations

Matplotlib provides functionalities to add text and annotations to the plots.

- Adding Text:

plt.text(0.5, 0.5, 'Hello')

- Adding Annotations:

plt.annotate('Hello', xy=(0.5, 0.5), xytext=(0.6, 0.6), arrowprops=dict(facecolor='black', shrink=0.05))

7. Saving Figures

Matplotlib provides the savefig() function to save the current figure to a file.

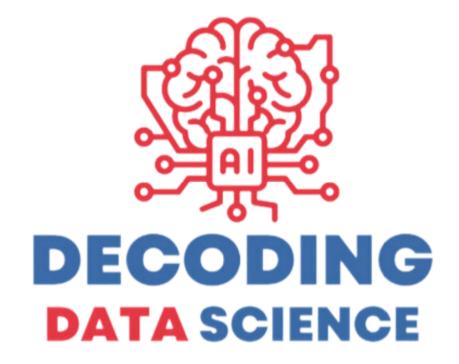
- Saving Figures as PNG, PDF, SVG, and more:

plt.savefig('figure.png')

plt.savefig('figure.pdf')

plt.savefig('figure.svg')

Pandas Cheat Sheet DECODING DATA SCIENCE



JOIN OUR AI COMMUNITY: HTTPS://NAS.IO/ARTIFICIALINTELLIGENCE

1. Basic Commands

Pandas is a software library for Python that provides tools for data manipulation and analysis. It's important to ensure that the correct version of pandas is installed for compatibility with your code.

- Importing pandas:
- import pandas as pd
- Checking pandas version:
 - print(pd.__version__)

2. Dataframe Creation

Dataframes are two-dimensional labeled data structures with columns potentially of different types.

You can think of it like a spreadsheet or SQL table.

```
- From a list:

my_list = [1, 2, 3, 4, 5]

df = pd.DataFrame(my_list, columns=['column_name'])

- From a dictionary:

my_dict = {'A': [1, 2, 3], 'B': [4, 5, 6]}

df = pd.DataFrame(my_dict)
```

3. Data Selection

Pandas provides different methods for data selection.

- Selecting a column:df['A']
- Selecting multiple columns:df[['A', 'B']]
 - Selecting rows:df.loc[0] # row labeldf.iloc[0] # row index
- Selecting specific value:
 df.at[0, 'A'] # row label and column name
 df.iat[0, 0] # row index and column index

4. Data Manipulation

Pandas provide various ways to manipulate a dataset.

- Adding a column:

- Deleting a column: df.drop('C', axis=1, inplace=True)
- - Applying a function to a column:

df['A'].apply(lambda x: x*2)

5. Data Cleaning

Data cleaning is the process of detecting and correcting (or removing) corrupt or inaccurate records from a dataset.

- Checking for null values: df.isnull()
 - Dropping null values:df.dropna(inplace=True)
- Filling null values:df.fillna(value=0, inplace=True)
- Replacing values:
 df.replace(1, 10, inplace=True)

6. Grouping & Aggregation

Grouping involves combining data based on some criteria, while aggregation is the process of turning the results of a query into a single row.

```
- Group by:
```

df.groupby('A')

- Aggregation:

df.agg({'A': ['min', 'max', 'mean', 'sum']})

7. Merging, Joining, and Concatenating

Pandas provides various ways to combine DataFrames including merge and join.

```
- Concatenating:
  df1 = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
df2 = pd.DataFrame({'A': [7, 8, 9], 'B': [10, 11, 12]})
             df = pd.concat([df1, df2])
                      - Merging:
  df1 = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
  df2 = pd.DataFrame({'A': [1, 2, 3], 'C': [7, 8, 9]})
          df = pd.merge(df1, df2, on='A')
                      - Joining:
  df1 = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
         df2 = pd.DataFrame({'C': [7, 8, 9]})
                  df = df1.join(df2)
```

JOIN OUR AI COMMUNITY: HTTPS://NAS.IO/ARTIFICIALINTELLIGENCE

8. Working with Dates

Pandas provides powerful functionalities for working with dates.

- Convert to datetime:
df['date'] = pd.to_datetime(df['date'])

- Extracting date parts:

df['year'] = df['date'].dt.year

df['month'] = df['date'].dt.month

df['day'] = df['date'].dt.day

9. File I/O

Pandas can seamlessly read from and write to a variety of file formats.

- Reading a CSV file:

df = pd.read_csv('file.csv')

- Writing to a CSV file:
df.to_csv('file.csv', index=False)

Similarly for other file formats like Excel (read_excel, to_excel), JSON (read_json, to_json), SQL (read_sql, to_sql), etc.

Join Our Al Community

Being part of an AI community like ours offers multiple benefits. You can network with likeminded individuals, learn from experienced professionals, and stay up-to-date with the latest AI trends and developments.

Whether you're a beginner looking to start your journey in AI, or an experienced professional looking to enhance your skills, our community has something to offer.

Join us and be a part of this exciting journey.

Learn more, Grow more!

Click the link in the footer to join us today!