COP 3035

Intro Programming in Python

Summer 2024

Lecture 22 – part 1

Exam 4 – 08/02/24

Lecture 22 – part 2

Review

Review

Python Packages

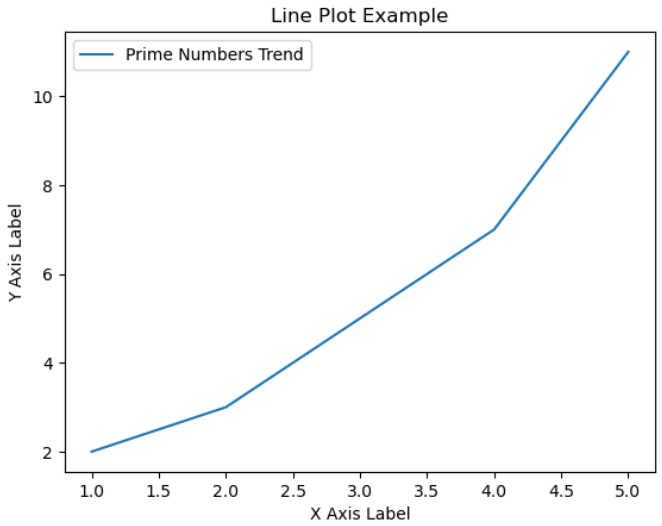
Matplotlib

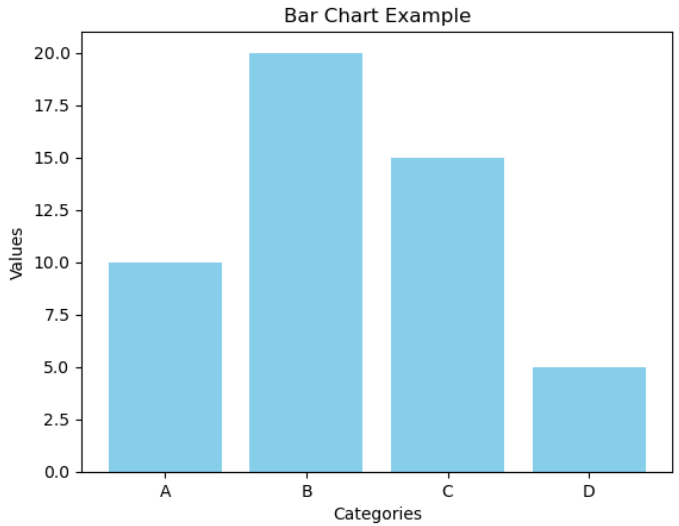
Basic plotting

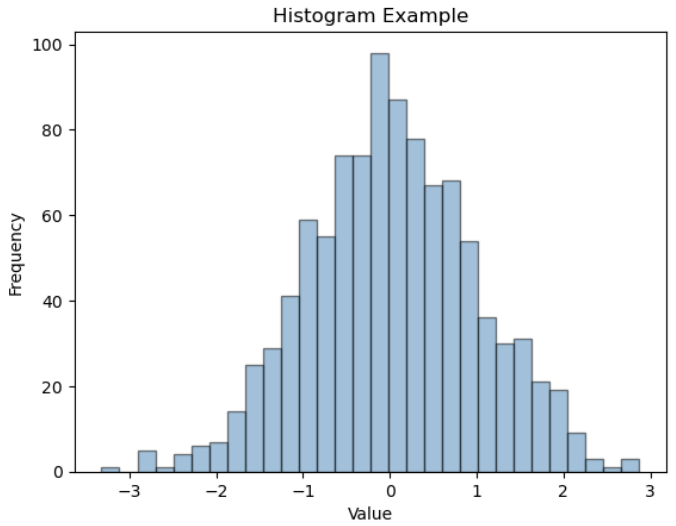
Line Plot : Basic syntax and customization (color, linestyle, marker).

Bar Chart: Comparing data side-by-side.

Histogram: Visualizing distributions.







https://matplotlib.org/stable/users/explain/quick_start.html

Common Parameters

data:	An optional parameter that allows specifying the data source (Dictionary).
color:	Specifies the color of the line. You can use named colors, hex codes, or RGB/A tuples.
label:	Sets the label for this line, which will appear in the legend.
linewidth or **lw**:	Sets the width of the line.
markersize or **ms**:	Determines the size of the markers.
linestyle or **ls**:	Defines the style of the line, such as solid, dashed, or none.
marker:	Chooses the marker style for the points, like circles, squares, etc.

```
plt.plot(y)     # Assumes x as range(len(y))
plt.plot(x, y)  # Plots y vs x
plt.plot(x, y, 'bo-')  # Blue circles with solid lines
plt.plot(x, y, color='green', marker='o', linestyle='dashed’,
         linewidth=2, markersize=12)
```

Example using the data parameter:

```
import matplotlib.pyplot as plt

# Sample data as a dictionary
data = {
    'x': range(1, 11),
    'y1': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'y2': [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
}

# Plotting with explicit formatting
plt.plot('x', 'y1', data=data, marker='o', color='red', linestyle='-', label='Ascending')
plt.plot('x', 'y2', data=data, marker='^', color='blue', linestyle='--', label='Descending')

plt.legend()
plt.show()
```

Exercises

1. Draw a plot of a simple list [1,2,3,4,5]. Then add a title and a grid.
2. Draw a plot using two lists X and Y. Add title, grid, axis titles and a line label.
3. Draw three plots for $y_1=x$, $y_2=x^2$ and $y_3=x^3$, for the first 50 numbers. Use different colors and markers.
4. Generate a line plot with three different mathematical functions:

   y1 = sin(x) ,

   y2 = cos(x) and

   y3 = 2sin(x)cos(x),

   for x ranging from 0 to 2π.

   Use different line styles and colors for each function. Include a title and legends. Label the axis.

5. Complete the tutorial for matplotlib.pyplot at: https://matplotlib.org/stable/tutorials/pyplot.html#sphx-glr-tutorials-pyplot-py

Lecture 22 – part 4

NumPy

What is NumPy ?

- **NumPy** stands for Numerical Python.
- It's a library for Python that supports **large, multi-dimensional arrays and matrices**.
- Provides a wide range of **mathematical functions** to operate on these arrays **efficiently**.
- **Simplifies** complex numerical operations, making code more readable and concise.
- Works well with **other libraries** in the Python data ecosystem (e.g., pandas, matplotlib).
- **Common Use Cases:** Data analysis, Machine Learning, Image processing, simulations

Installation:

```
pip install numpy
```

Importing:

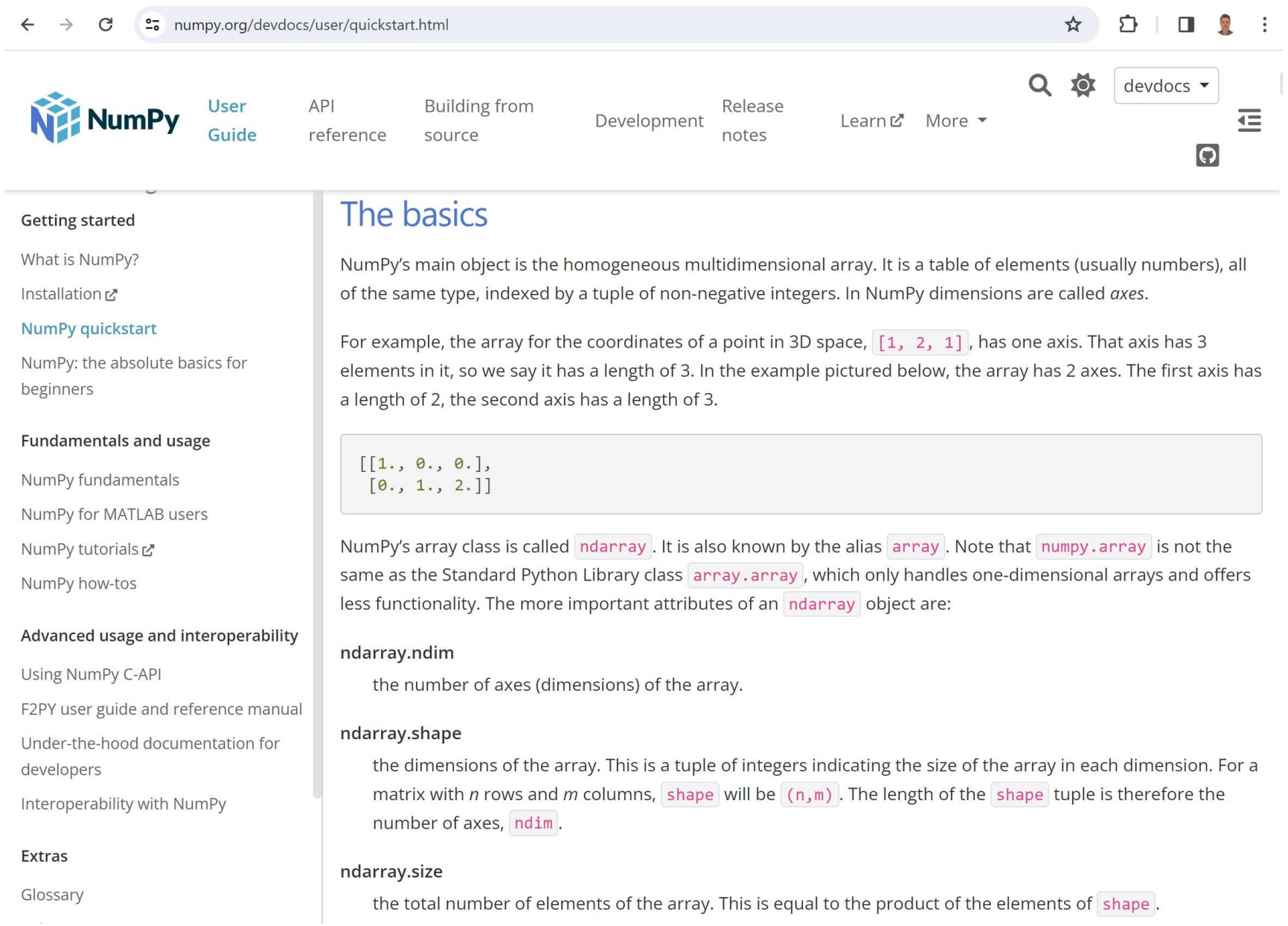
```
import numpy as np
```

Creating an array:

```
arr = np.array([1, 2, 3])
```

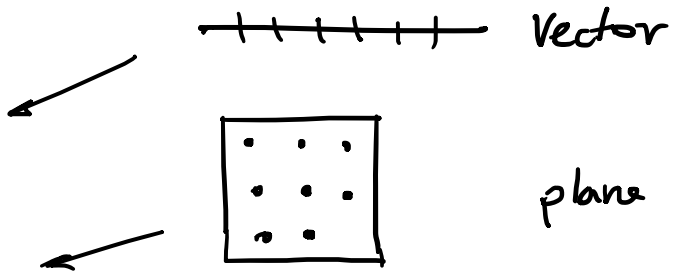
NumPy's array class is called **ndarray**

https://numpy.org/devdocs/user/quickstart.html



NumPy dimensions

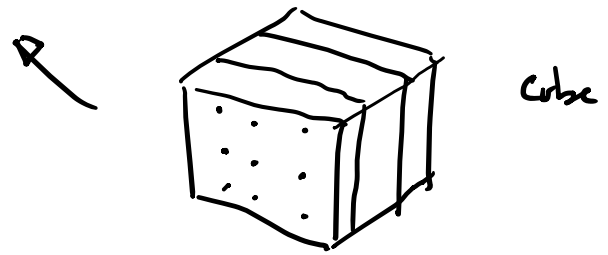
- NumPy dimensions are called ***axes***




```
one_d_array = np.array([1, 2, 3, 4, 5])

two_d_array = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

three_d_array = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]], [[13, 14, 15], [16, 17, 18]]])
```



Basics

Feature	Description	Example Code
np.array	Creates an array.	A = np.array([1, 2, 3])
.ndim	Number of array dimensions.	A.ndim
.shape	Dimensions of the array.	A.shape
.size	Number of elements in the array.	A.size
np.zeros	Creates an array filled with zeros.	np.zeros((2, 3))
np.ones	Creates an array filled with ones.	np.ones((3, 3))
np.arange	Creates an array with a range of values.	np.arange(0, 10, 2)
np.linspace	Creates an array with evenly spaced values.	np.linspace(0, 1, 5)

Operations

Feature	Description	Example Code
-	Subtraction element-wise.	A - B
**	Exponentiation element-wise.	A ** 2
*	Multiplication element-wise.	A * B
<	Element-wise comparison.	A < B
.dot()	Dot product of two arrays.	A.dot(B) or np.dot(A, B)

Math

Feature	Description	Example Code
np.add()	Element-wise addition.	np.add(A, B)
np.exp()	Exponential of all elements.	np.exp(A)
np.sqrt()	Square root of each element.	np.sqrt(A)
np.sin()	Sine of each element.	np.sin(A)
np.cos()	Cosine of each element.	np.cos(A)

Indexing and Slicing

Feature	Description	Example Code
Single element indexing	Access a single element by its position.	arr[2] or arr[2, 3]
Slice along one dimension	Select a range of elements from an array.	arr[0:5] or arr[:5]
Slice along two dimensions	Select a rectangle of elements.	arr[1:4, 0:3]
Stride for slicing	Select elements with a step size between them.	arr[::2] or arr[0:5:2]
Negative slicing	Use negative indices to slice from the end of the array.	arr[-3:] or arr[1:-1]
Boolean indexing	Select elements based on a boolean condition.	arr[arr > 5]
Fancy indexing	Index with integer arrays.	arr[[1, 3, 4]] or arr[[1, 2], [3, 4]]
Mixing integer and slice	Combine integer indexing and slicing.	arr[1, :2] or arr[2:3, :1]
Ellipsis (...)	Used to replace multiple colons.	arr[..., 1] (same as arr[:, 1])

NumPy Array Manipulations

.ravel()

- Flattens an array into a contiguous 1D array.
- Returns a view of the original array whenever possible, making it memory efficient.
- Changes to the returned array may affect the original array.
- Example: `flattened_array = arr.ravel()`

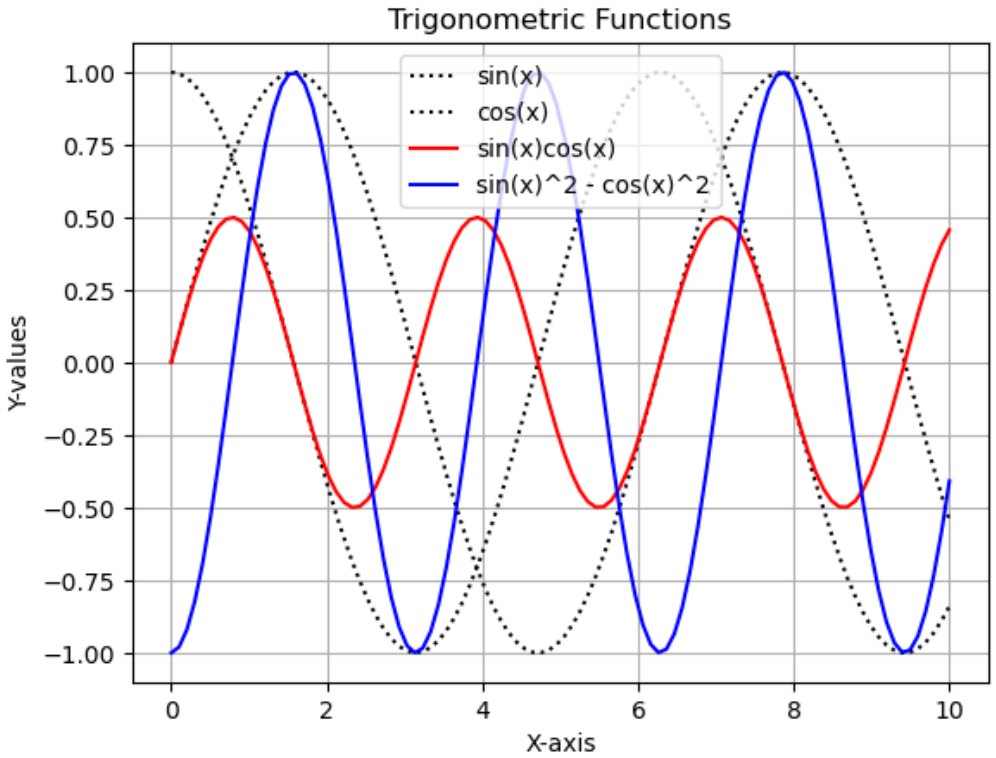
.reshape()

- Gives a new shape to an array without changing its data.
- Returns a new array with the specified shape.
- Can return a view or a copy, depending on the memory layout.
- Use -1 to automatically calculate the size of one dimension.
- Example: `reshaped_array = arr.reshape(2, 6)`

.resize()

- Alters the size and shape of an array in-place.
- Can expand or shrink the array; new elements are filled with zeros.
- Does not return a value; modifies the original array.
- Example: `arr.resize((2, 6))`

Example: Using Numpy, Matplotlib and Pandas



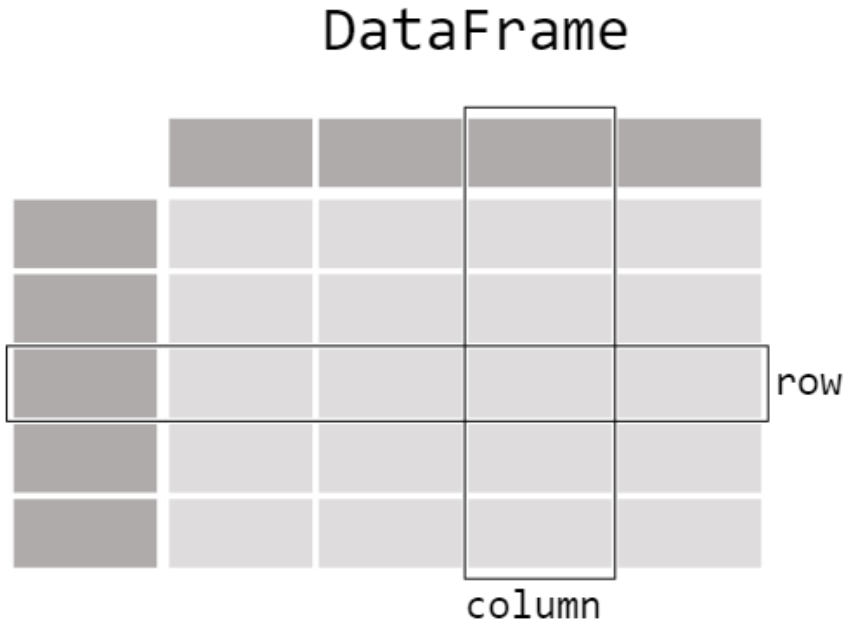
Lecture 22 – part 5

Pandas

Pandas

Pandas is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language.

- Provides essential data structures like **DataFrame** and **Series** for efficient data manipulation and analysis.
- Supports **various file formats** including CSV, Excel, and SQL, facilitating easy data reading and writing.
- Offers **tools** for handling missing data, merging, joining, and filtering datasets effectively.
- Includes **robust features for time series** data manipulation such as resampling and frequency conversion.
- Backed by a **strong community** with extensive documentation, making it a reliable choice for data scientists and analysts.

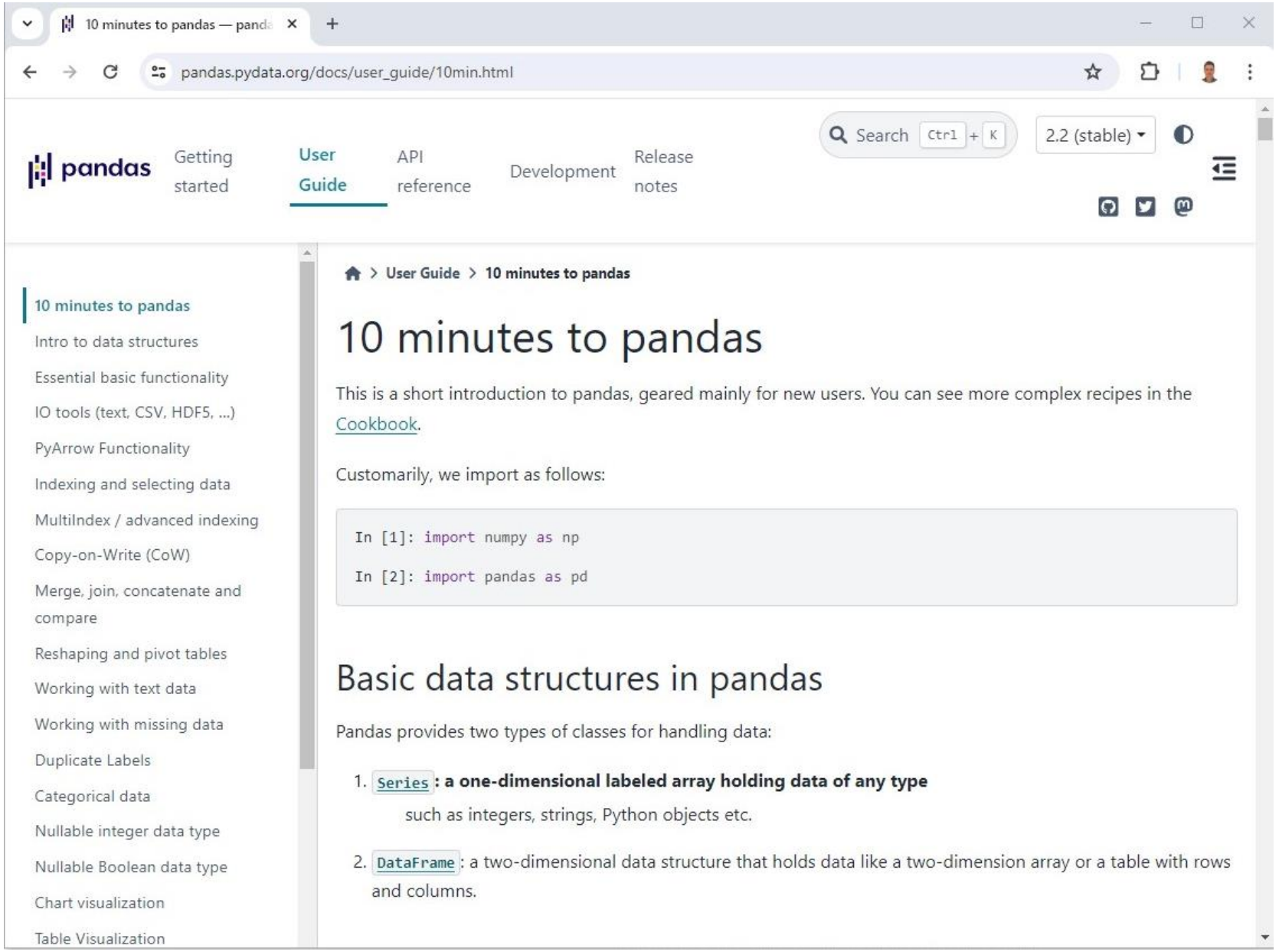


Installation

```
pip install pandas
```

Import

```
import pandas as pd
```



https://pandas.pydata.org/docs/user_guide/10min.html