

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

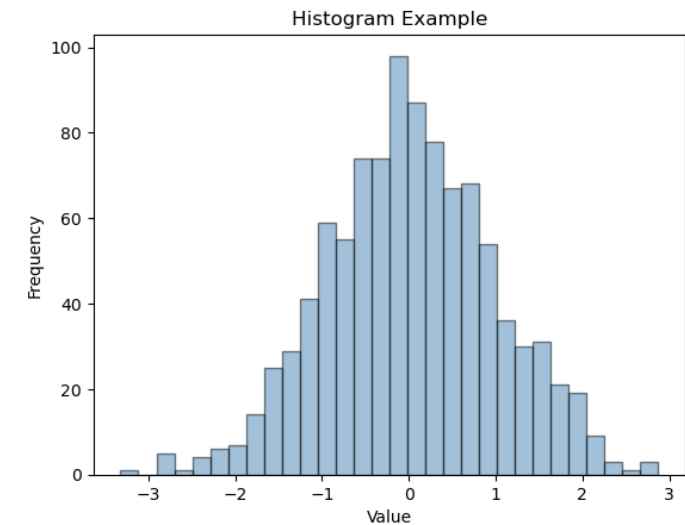
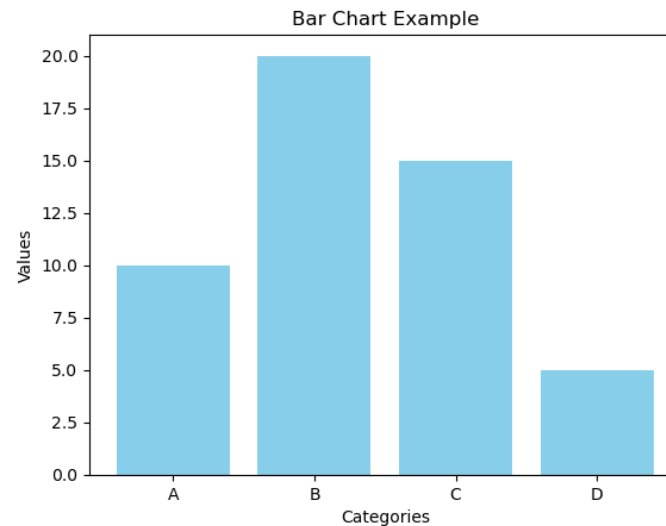
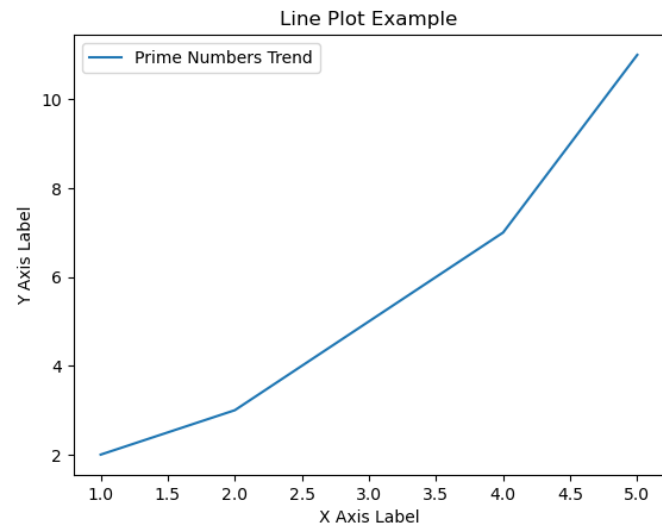
Matplotlib

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:
 $y_1 = \sin(x)$,
 $y_2 = \cos(x)$ and
 $y_3 = 2\sin(x)\cos(x)$,
 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**



User
Guide

API
reference

Building from
source

Development

Release
notes

Learn ↗ More ▾



devdocs ▾



Getting started

What is NumPy?

Installation ↗

NumPy quickstart

NumPy: the absolute basics for
beginners

Fundamentals and usage

NumPy fundamentals

NumPy for MATLAB users

NumPy tutorials ↗

NumPy how-tos

Advanced usage and interoperability

Using NumPy C-API

F2PY user guide and reference manual

Under-the-hood documentation for
developers

Interoperability with NumPy

Extras

Glossary

The basics

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of non-negative integers. In NumPy dimensions are called *axes*.

For example, the array for the coordinates of a point in 3D space, `[1, 2, 1]`, has one axis. That axis has 3 elements in it, so we say it has a length of 3. In the example pictured below, the array has 2 axes. The first axis has a length of 2, the second axis has a length of 3.

```
[[1., 0., 0.],  
 [0., 1., 2.]]
```

NumPy's array class is called `ndarray`. It is also known by the alias `array`. Note that `numpy.array` is not the same as the Standard Python Library class `array.array`, which only handles one-dimensional arrays and offers less functionality. The more important attributes of an `ndarray` object are:

`ndarray.ndim`

the number of axes (dimensions) of the array.

`ndarray.shape`

the dimensions of the array. This is a tuple of integers indicating the size of the array in each dimension. For a matrix with n rows and m columns, `shape` will be `(n,m)`. The length of the `shape` tuple is therefore the number of axes, `ndim`.

`ndarray.size`

the total number of elements of the array. This is equal to the product of the elements of `shape`.

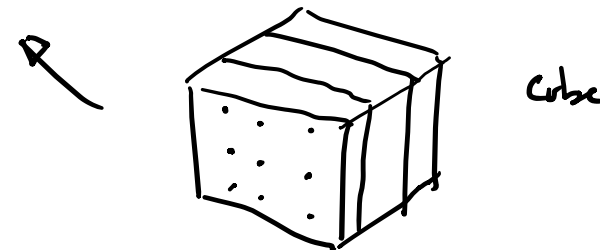
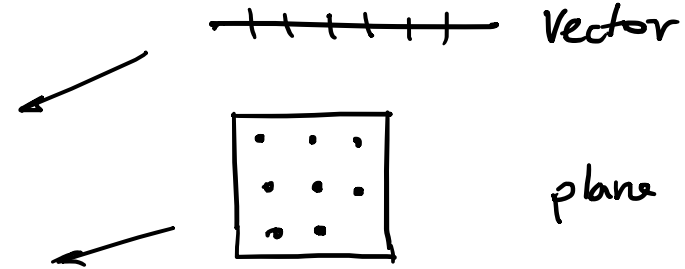
NumPy dimensions

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

Operations

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

Math

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

Indexing and Slicing

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

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()**

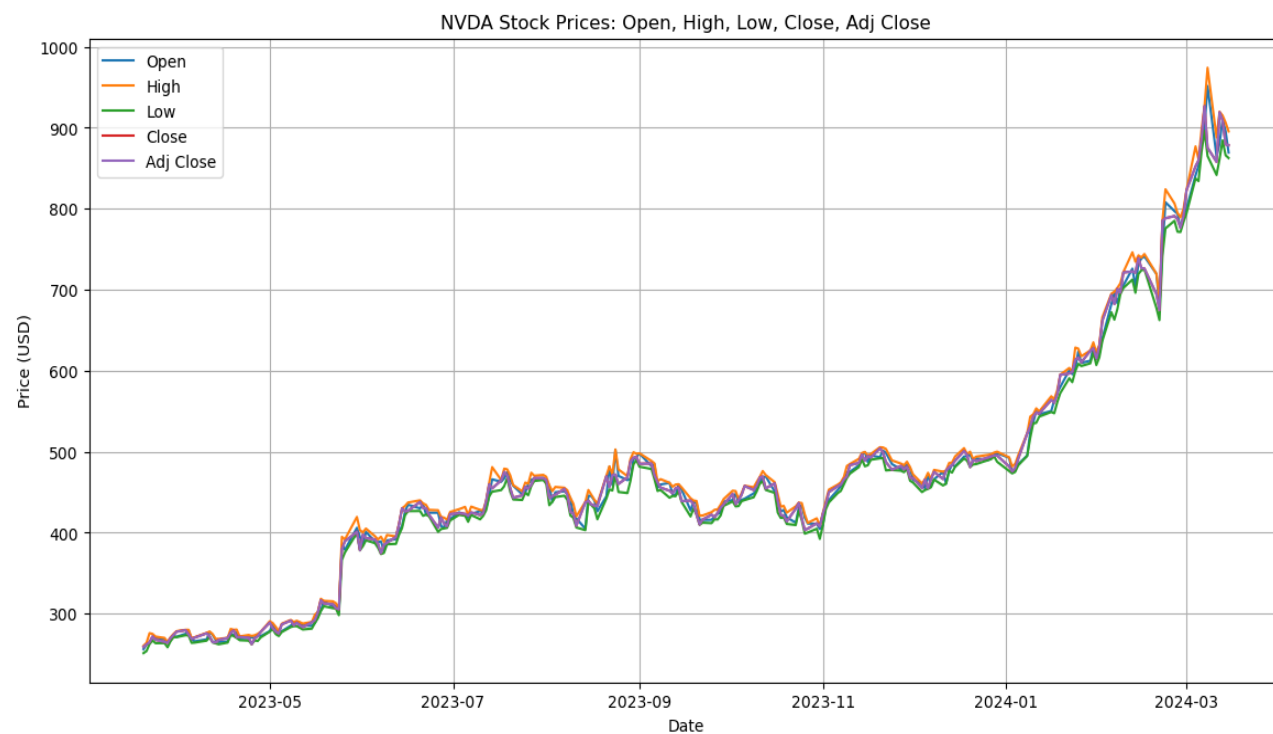
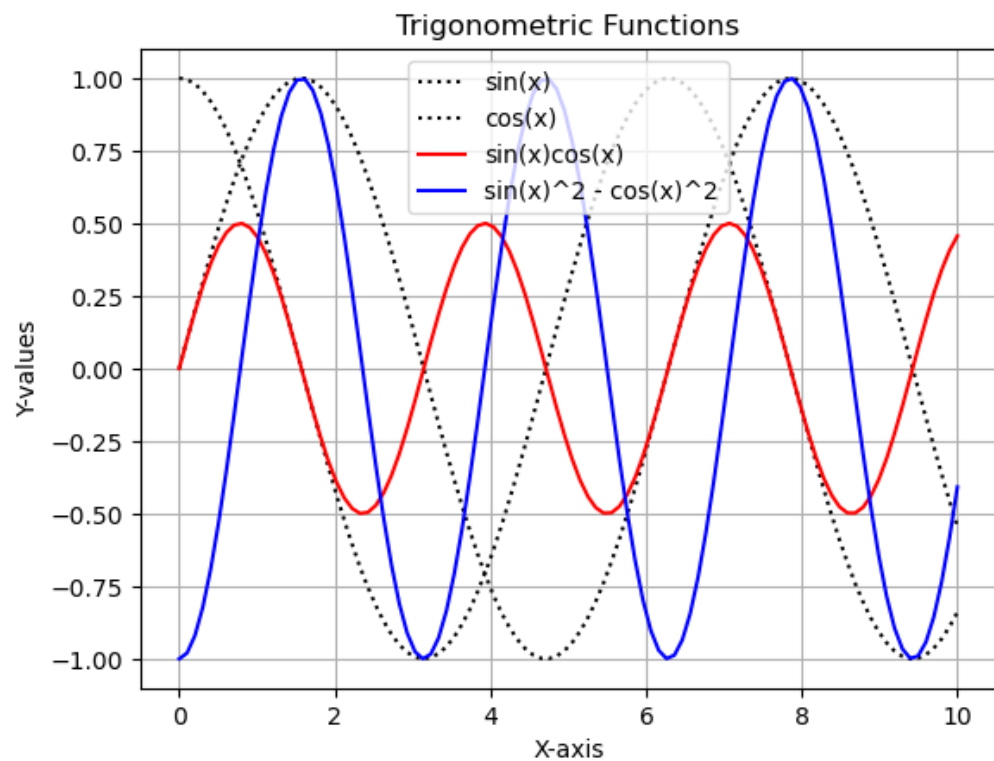
.reshape()

- 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



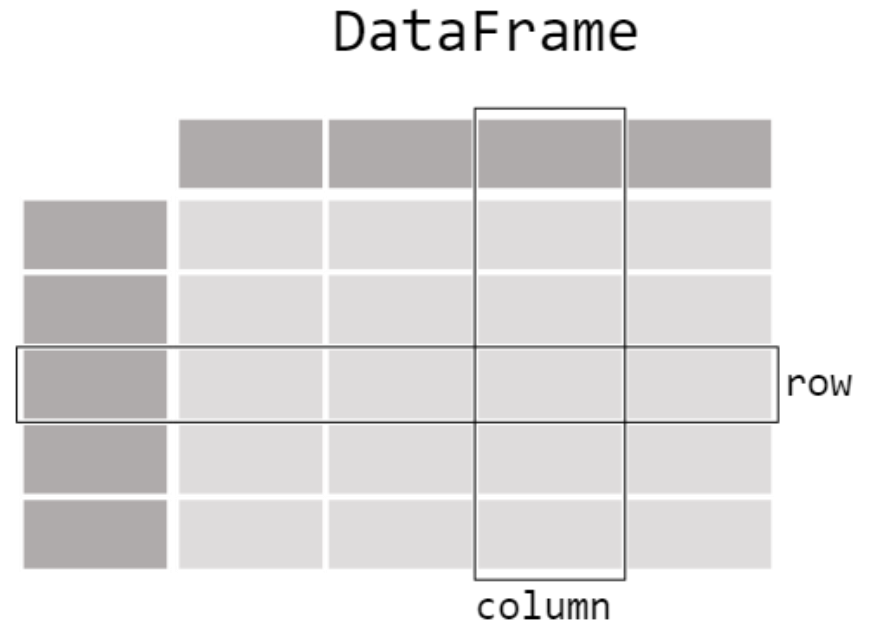
Lecture 22 – part 5

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
```

The screenshot shows a web browser displaying the pandas documentation page titled "10 minutes to pandas". The browser's address bar shows the URL "pandas.pydata.org/docs/user_guide/10min.html". The page features a navigation bar with links to "Getting started", "User Guide" (which is highlighted), "API reference", "Development", and "Release notes". A search bar and a version selector set to "2.2 (stable)" are also present. On the left side, a sidebar lists various topics, with "10 minutes to pandas" selected. The main content area has a breadcrumb trail "User Guide > 10 minutes to pandas" and a large heading "10 minutes to pandas". Below the heading, a paragraph introduces the tutorial as a short introduction for new users, with a link to the "Cookbook". It then states, "Customarily, we import as follows:", followed by a code block containing two lines of Python code:

```
In [1]: import numpy as np
In [2]: import pandas as pd
```

. The next section is titled "Basic data structures in pandas" and explains that pandas provides two types of classes for handling data. It lists two items: 1. **Series**: a one-dimensional labeled array holding data of any type such as integers, strings, Python objects etc. 2. **DataFrame**: a two-dimensional data structure that holds data like a two-dimension array or a table with rows and columns.

10 minutes to pandas

Intro to data structures

Essential basic functionality

IO tools (text, CSV, HDF5, ...)

PyArrow Functionality

Indexing and selecting data

MultilIndex / advanced indexing

Copy-on-Write (CoW)

Merge, join, concatenate and compare

Reshaping and pivot tables

Working with text data

Working with missing data

Duplicate Labels

Categorical data

Nullable integer data type

Nullable Boolean data type

Chart visualization

Table Visualization

10 minutes to pandas

This is a short introduction to pandas, geared mainly for new users. You can see more complex recipes in the [Cookbook](#).

Customarily, we import as follows:

```
In [1]: import numpy as np
In [2]: import pandas as pd
```

Basic data structures in pandas

Pandas provides two types of classes for handling data:

1. **Series**: a one-dimensional labeled array holding data of any type such as integers, strings, Python objects etc.
2. **DataFrame**: a two-dimensional data structure that holds data like a two-dimension array or a table with rows and columns.

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