Python for Beginners with applications

Lesson 8:

Introduction to Pandas library

Write code from scratch in a clear & concise way, with a complete basic course.

From beginners to intermediate.

Abdesselam Filali *2025-07-01 17h00 Algiers time*

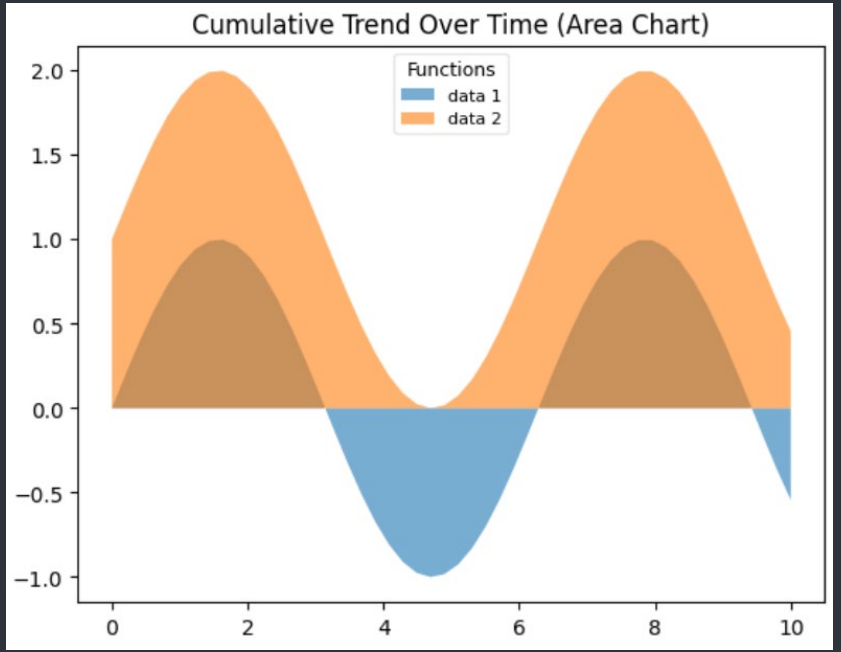

1.
Data visualisation trick

Visualization: fill_between & legend

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 50)
y1 = np.sin(x)
y2 = np.sin(x) + 1

plt.fill_between(x, y1, alpha=0.6, label='data 1')
plt.fill_between(x, y2, alpha=0.6, label='data 2')
plt.title('Cumulative Trend Over Time (Area Chart)')
plt.legend(loc=1)
plt.show()
```



Visualization: fill_between & legend

```
plt.legend(

        loc=,
        ncol=,
        fontsize=,
        framealpha=,
        title='Functions',
        title_fontsize=

)
```

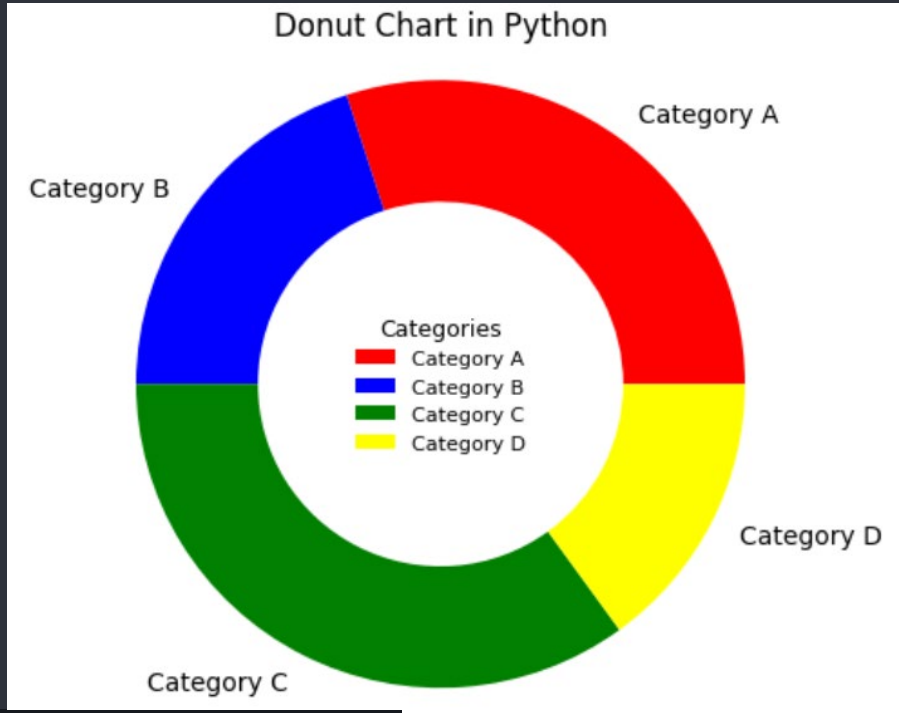
Legend : location

Location String	Location Code
'best' (Axes only)	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

Legend donut chart

Visualization: fill_between & legend

```
labels = ['Category A',
          'Category B',
          'Category C',
          'Category D']
sizes = [30, 20, 35, 15]
colors = ['red', 'blue', 'green', 'yellow']
plt.pie(sizes, labels=labels,
          colors=colors,
          wedgeprops={'width': 0.4})
plt.axis('equal')
plt.title('Donut Chart in Python')
plt.legend(loc=10,
          fontsize=8,
          framealpha=0,
          title='Categories',
          title_fontsize='9')
plt.show()
```



2.

Introduction to Pandas library

What is Pandas?

Built on top of the Python programming language.

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, it is used for working with data sets.

It has functions for analyzing, cleaning, exploring, and manipulating data.

The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Pandas allows us to analyze big data and make conclusions based on statistical theories.

Pandas can clean messy data sets and make them readable and relevant.
Relevant data is very important in data science.

https://pandas.pydata.org/
https://www.w3schools.com/python/pandas/default.asp

What Can Pandas Do?

Pandas gives you answers about the data. Like:

- Is there a correlation between two or more columns?
- What is average value?
- Max value?
- Min value?

Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values. This is called cleaning the data.

Where is the Pandas Codebase?

The source code for Pandas is located at this github repository
https://github.com/pandas-dev/pandas

Pandas Library

Installation & importing

installation

```
pip install pandas
```

importing

```
import pandas as pd
```

Checking Pandas Version

```
import pandas as pd
print(pd.__version__)
```

Loading a data file

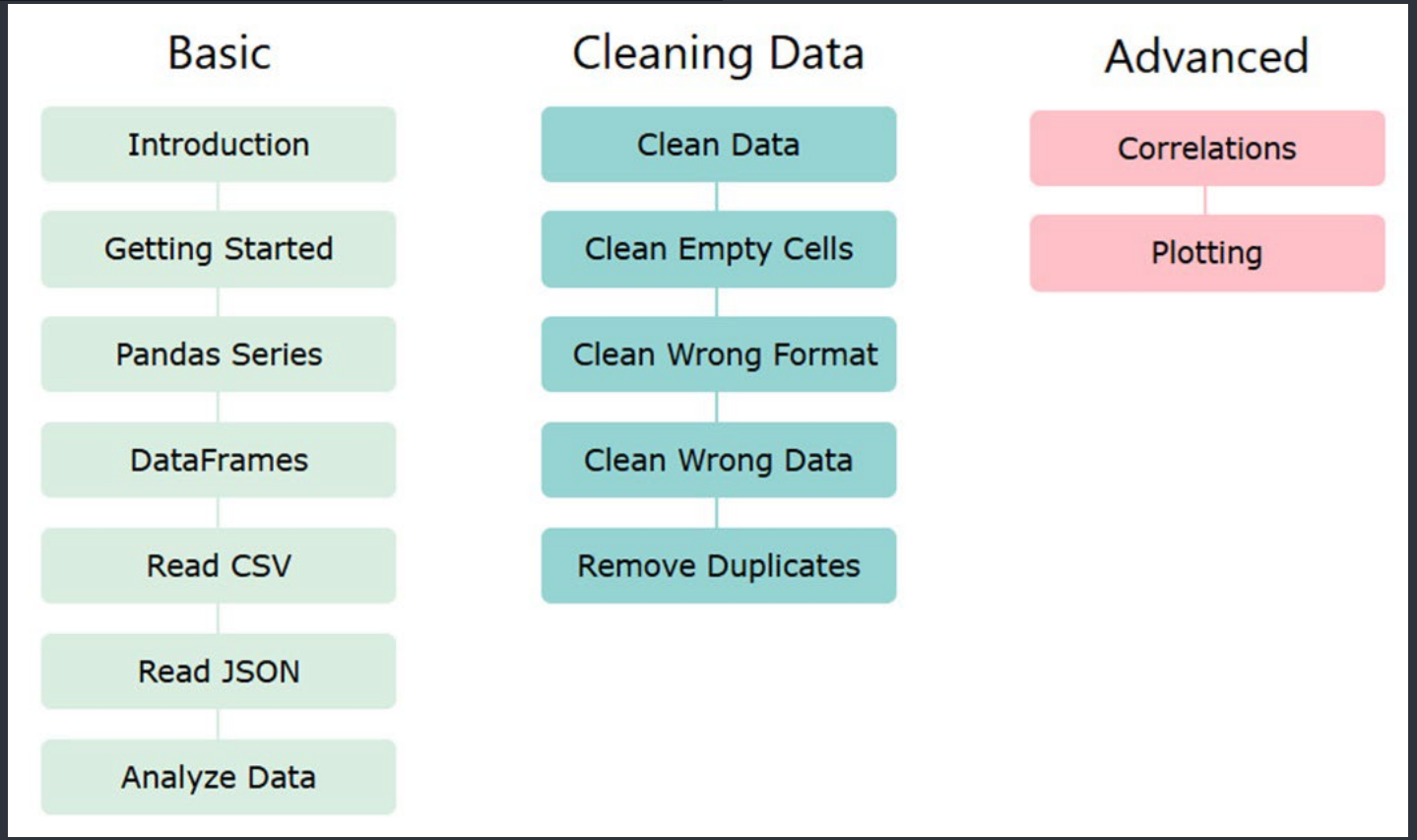
```
import pandas as pd
df = pd.read_csv('data.csv’)
print(df.to_string())
```

https://pandas.pydata.org/

pandas

Pandas Library

Pandas Roadmap



https://pandas.pydata.org/

pandas

installation

```
pip install pandas
```

importing

```
import pandas as pd
```

Checking Pandas Version

```
import pandas as pd
print(pd.__version__)
```

Loading a data file

```
import pandas as pd
df = pd.read_csv('data.csv’)
print(df.to_string())
```

What is a Series?

What is a Series?
A Pandas Series is like a column in a table.

It is a one-dimensional array holding data of any type.

Example:

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.Series(a)
print(myvar)
```

Labels ?

If nothing else is specified, the values are labeled with their index number. First value has index 0, second value has index 1 etc.

This label can be used to access a specified value. ype.

Example:

```
print(myvar[0])
```

Create labels ?

With the index argument, you can name your own labels.

Example:

```
import pandas as pd
a = [1, 7, 2]
myvar = pd.Series(a, index = ["x", "y", "z"])
print(myvar)
```

Labels ?

Key/Value Objects as Series

You can also use a key/value object, like a dictionary, when creating a Series.

Example:

```
import pandas as pd
calories = {"day1": 420, "day2": 380, "day3": 390}
myvar = pd.Series(calories)
print(myvar)
```

Note :

The keys of the dictionary become the labels.

Definition ?

Data sets in Pandas are usually multi-dimensional tables, called DataFrames.
Series is like a column, a DataFrame is the whole table.

Example:

```
import pandas as pd
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
myvar = pd.DataFrame(data)
print(myvar)
```

Definition ?

Data sets in Pandas are usually multi-dimensional tables, called DataFrames.
Series is like a column, a DataFrame is the whole table.

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

Example:

```
import pandas as pd
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}
df = pd.DataFrame(data)
print(df)
```

	calories	duration
0	420	50
1	380	40
2	390	45

Read CSV Files

A simple way to store big data sets is to use CSV files (comma separated files).

CSV files contains plain text and is a well know format that can be read by everyone including Pandas.

Example:

```
import pandas as pd
df = pd.read_csv('data.csv')
print(df.to_string())
```

```
    calories  duration
0        420        50
1        380        40
2        390        45
```

Read CSV Files

Big data sets are often stored, or extracted as JSON.

JSON is plain text, but has the format of an object, and is well known in the world of programming, including Pandas.

Example:

```
import pandas as pd
df = pd.read_json('data.json’)
print(df.to_string())
```

Viewing the data:

One of the most used method for getting a quick overview of the DataFrame, is the head() method.

The head() method returns the headers and a specified number of rows, starting from the top.

Example:

```
import pandas as pd
df = pd.read_csv('data.csv’)
print(df.head(10))
```