Basic Programming

Lesson 11. Lesson 10 continued + Error handling

Alexandra Ciobotaru

7 ian 2021

Syllabus

- **Lesson 1. Computers, Programming and Cognitive Science. From pseudocode to programming languages.**
- **Lesson 2. Variables in Python. Basic calculus. Using Math library, type() and help() functions.**
- **Lesson 3. Collections: Lists, Tuples.**
- **Lesson 4. Strings. Working with strings.**
- **Lesson 5. Branching and decisions: Logical operators, If-Statements, Nested conditions. Loops: For and While**
- **Lesson 6. Lesson 5 continued** Quiz 1 (25%).
- **Lesson 7. Creating functions. Recursive functions. Matrices.**
- **Lesson 8. Collections: Dictionaries. JSON construction.**
- **Lesson 9. Working with files. Reading and Writing.**
- **Lesson 10. Analyzing dataframes. Pandas and Matplotlib Python libraries.**
- **Lesson 11. Lesson 10 continued + error handling**. Quiz 2 (25%).
- Lesson 12. Object-Oriented Programming (cont.). Error handling. Best practices when programming.
- Lab (20%) + final exam (30%).

What is Matplotlib?

- A comprehensive library for creating static, animated, and interactive visualizations in Python
- Installation: pip install matplotlib
- Importing: import matplotlib.pyplot as plt

- You can create plots with: matplotlib, seaborn and pandas as well!

Bar plot

- You can use **.plot.bar()** method to plot the graph vertically in form of rectangular bars

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

- Create a dataframe from a random numpy array

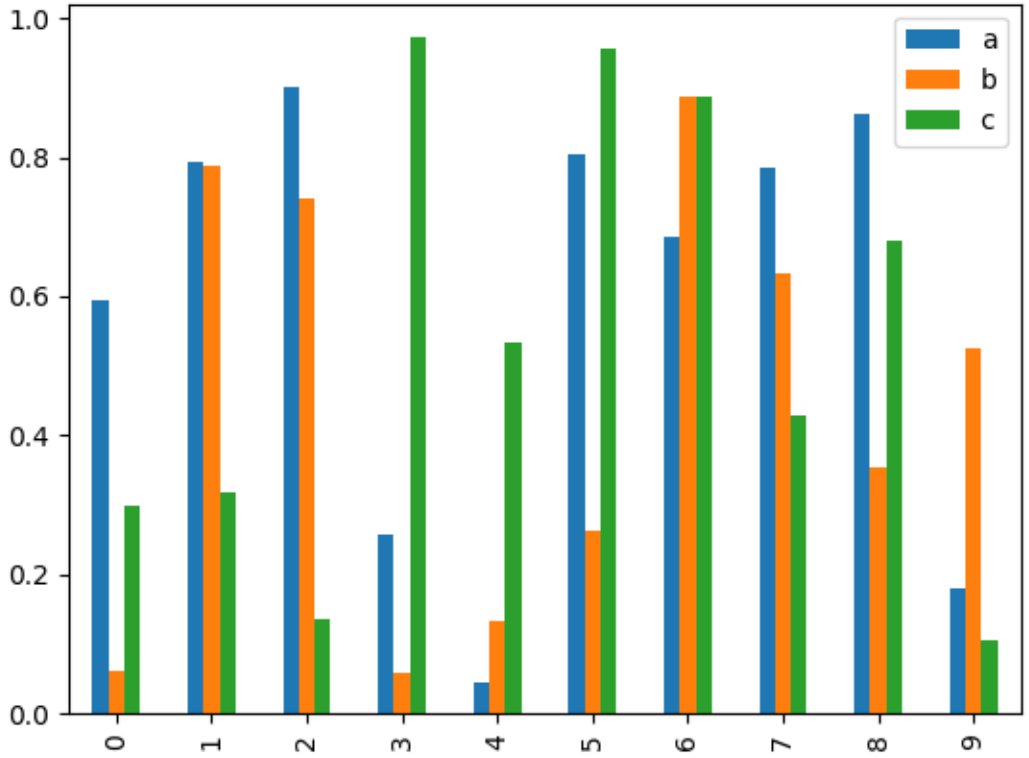
```
rnd_arr = np.random.rand(10, 3)
print(rnd_arr)
df = pd.DataFrame(rnd_arr, columns=['a', 'b', 'c'])
df.plot.bar()
plt.show()
```

- Plotting two columns from dataframe:

```
df.plot.bar(x = ‘name_of_column1’, ‘name_of_column2’)
```

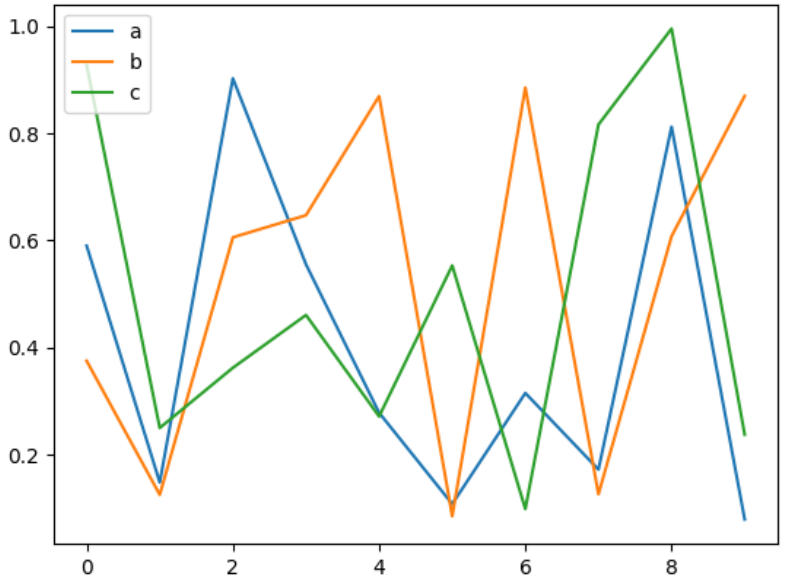
Bar plot

	a	b	c
0	0.594736	0.060920	0.299605
1	0.792587	0.788698	0.317061
2	0.900321	0.740707	0.136934
3	0.257342	0.058175	0.972296
4	0.045137	0.131768	0.534247
5	0.803600	0.262312	0.957628
6	0.684835	0.888481	0.887564
7	0.784256	0.633894	0.429599
8	0.862750	0.354024	0.679965
9	0.181168	0.524034	0.106147



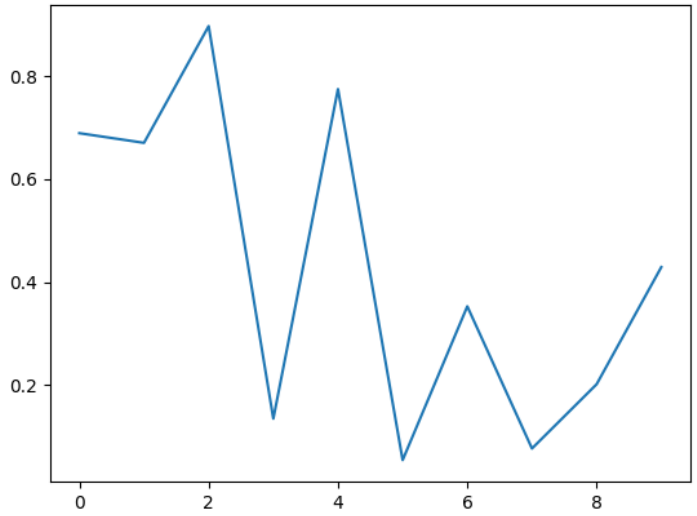
Line plot

- On the same dataframe as previous, if I write df.plot.line() instead of df.plot.bar():



- If I want, I can select only one column to plot:

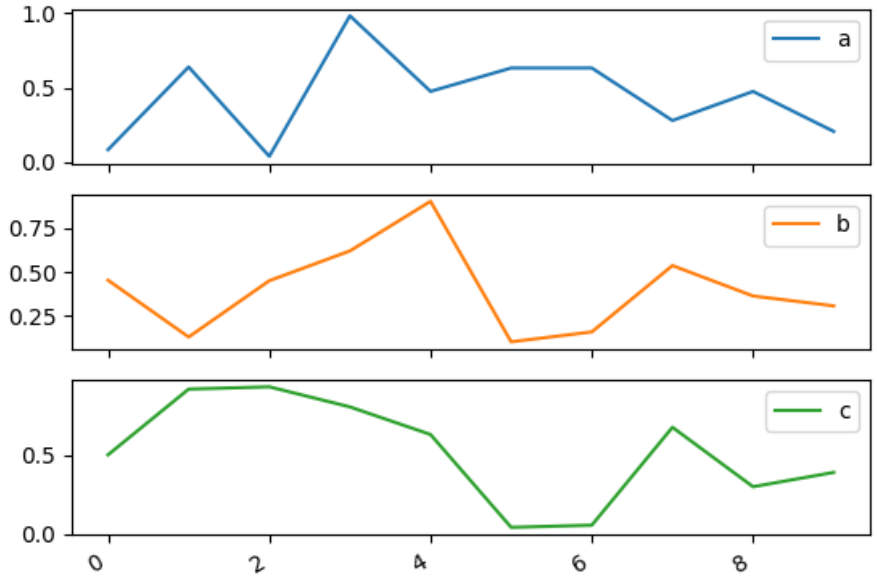
```
df['a'].plot.line()
```



Disclaimer: plots are different because of creating another random array on each run

- We can also see each line in dataframe separately:

```
axes = df.plot.line(subplots=True)
```

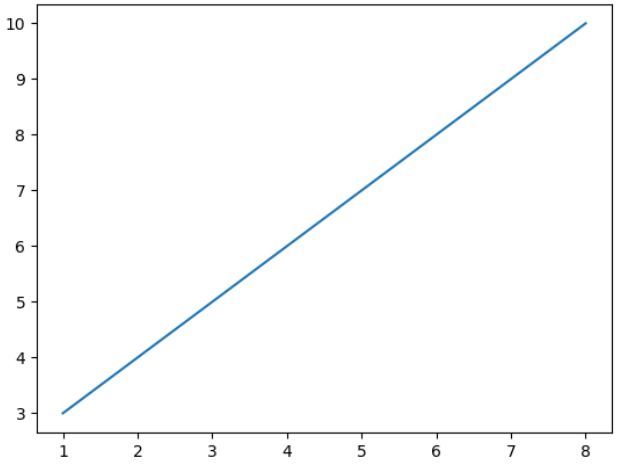


Creating line plots only with matplotlib

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

By setting coordonates for each point:

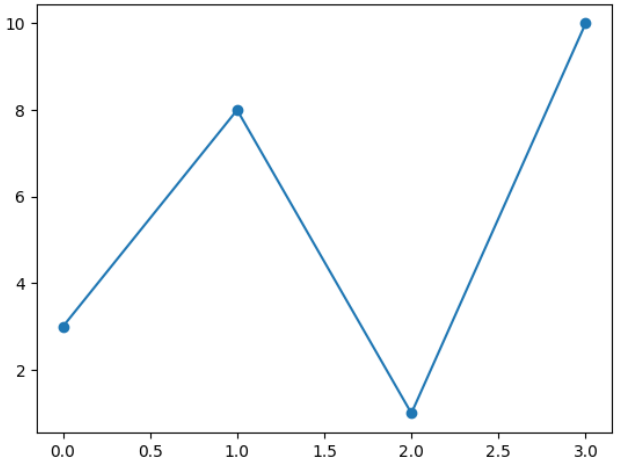
```
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])
plt.plot(xpoints, ypoints)
plt.show()
```



By setting only y coordinates.

And you can also use markers.

```
ypoints = np.array([3, 8, 1, 10])
plt.plot(ypoints, marker = 'o')
plt.show()
```



More on matplotlib: https://www.w3schools.com/python/matplotlib_intro.asp

Pie chart

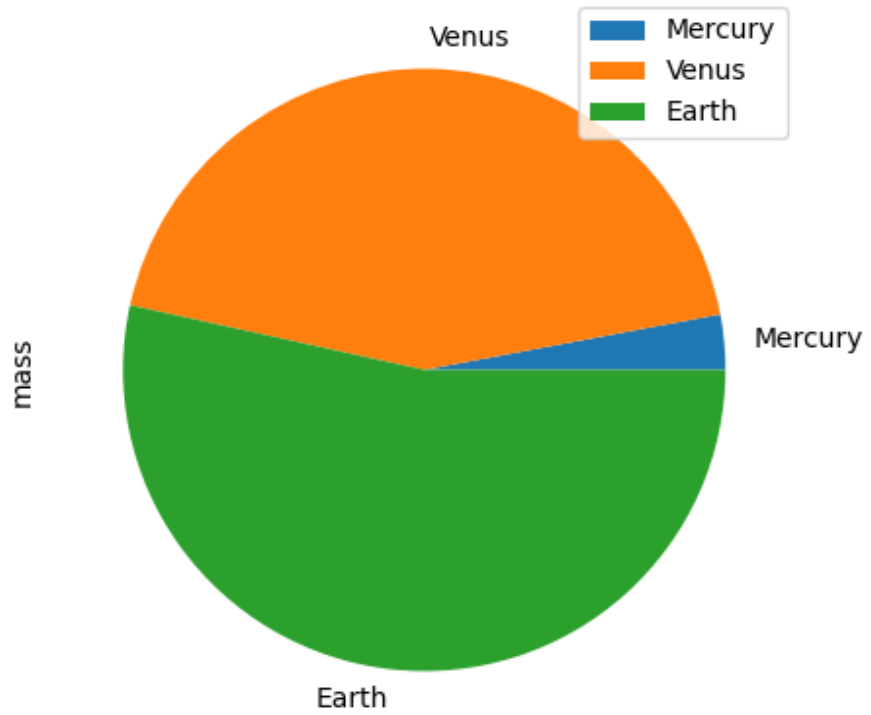
- Here is a dataframe holding information about some planets mass and radius:

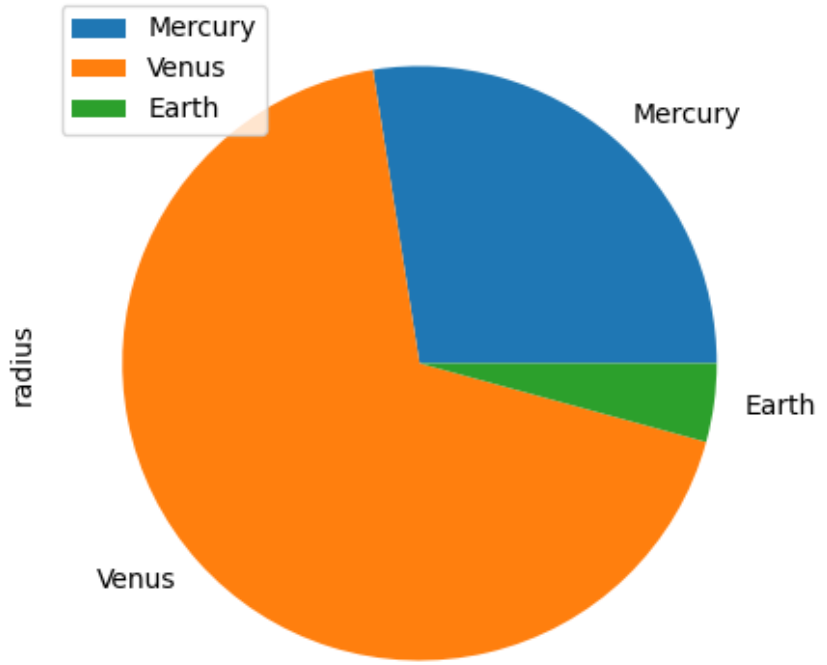
```
df = pd.DataFrame({'mass': [0.330, 4.87 , 5.97],
                   'radius': [2439.7, 6051.8, 378.1]},
                  index=['Mercury', 'Venus', 'Earth'])
```

- You can use .plot.pie method to get a pie plot:

```
plot = df.plot.pie(y='mass', figsize=(5, 5))
```

Pie chart





More about pandas plots: https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html

Error Handling

- When an error occurs, or exception as we call it, Python will normally stop and generate an error message.
- These exceptions can be handled using the **try – except** statement:

```
try:
  print(x)
except:
  print("An exception occurred")
```

- What do you think was the error?

it was: NameError: name 'x' is not defined

Error name

Error explanation

You can have many exceptions

- You can define as many exception blocks as you want, e.g. if you want to execute a special block of code for a special kind of error you need to know the name of your error:

```
try:
  print(x)
except NameError:
  print("Variable x is not defined")
except:
  print("Something else went wrong")
```

- But how do I know what is the exception I want to catch?

```
try:
    print(x)
except Exception as e:
    print("Error name is:", type(e).__name__)
    print("Error explanation is:", e)
```

Error handling

- The **try** block lets you test a block of code for errors.
- The **except** block lets you handle the error.
- The **finally** block, if specified, lets you execute some code, regardless of the result of the try-except blocks:

```
try:
  print(x)
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished")
```

- This can be useful to close objects and clean up resources.

Exercise

- Try to open a file that does not exist.
- What error does the execution raise?
- Can you catch it? Print the error name and explanation.
- Open the same file in writing mode (this will create an empty file with that name) and include a finally block to close the file.

Thank you!