Programming for Data Science: Matplotlib

Instructor: Jakramate Bootkrajang

Outline

- Matplotlib
- Pandas

Matplotlib

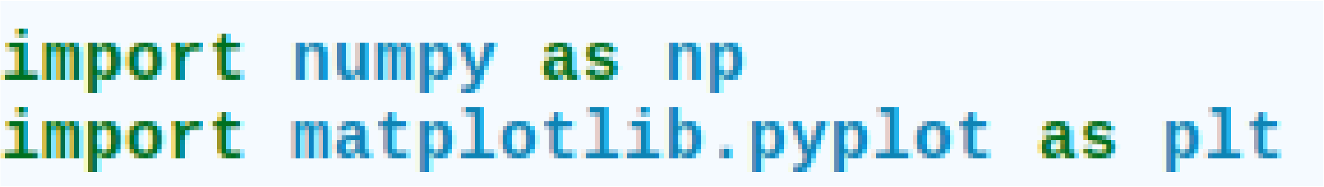
- Matplotlib is probably the most used Python package for 2D-graphics.
- It provides both a quick way to visualize data from Python and publication-quality figures in many formats.

Pyplot

- Pyplot is Matplotlib's sub-module which takes care of actual plotting
- pyplot provides a interface to the matplotlib plotting library.
- It is modeled closely after Matlab™.
- Therefore, the majority of plotting commands in pyplot have Matlab™ analogs with similar arguments

Importing pyplot

- Using import keyword

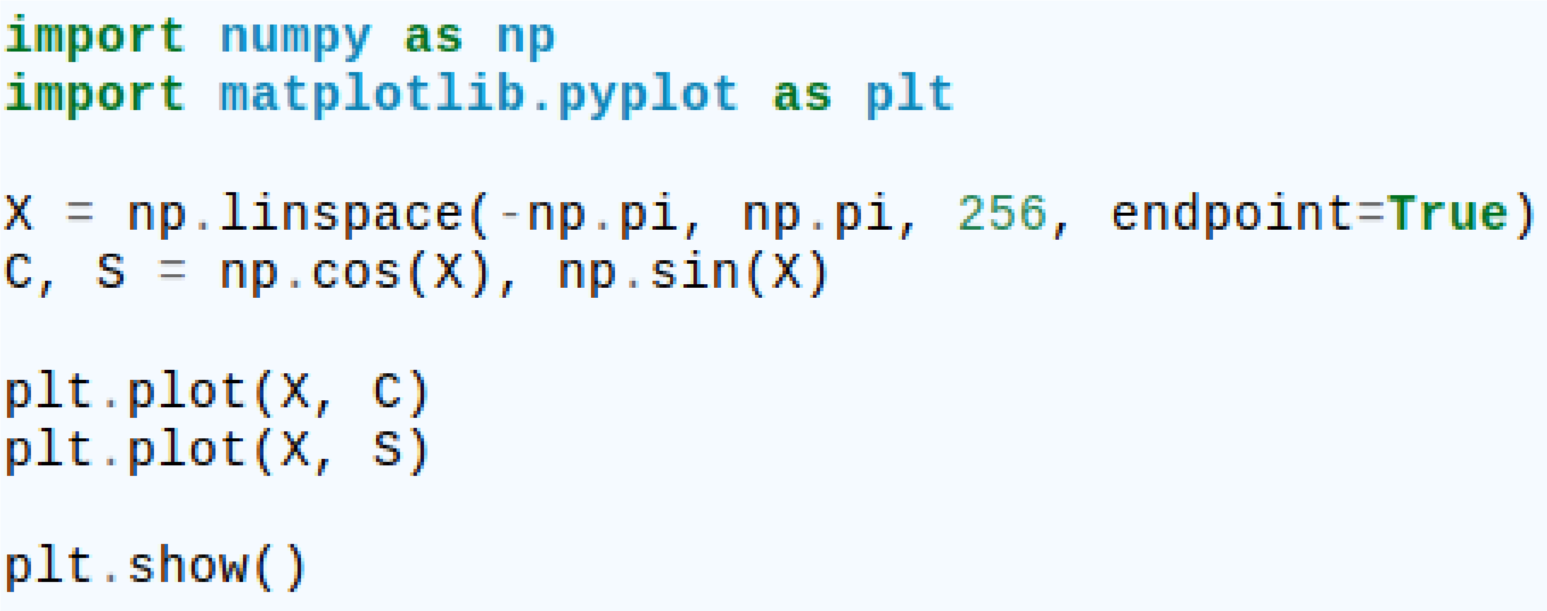


```
import numpy as np
import matplotlib.pyplot as plt
```

- The numpy / pyplot combo is usually used together

Plotting plot() function

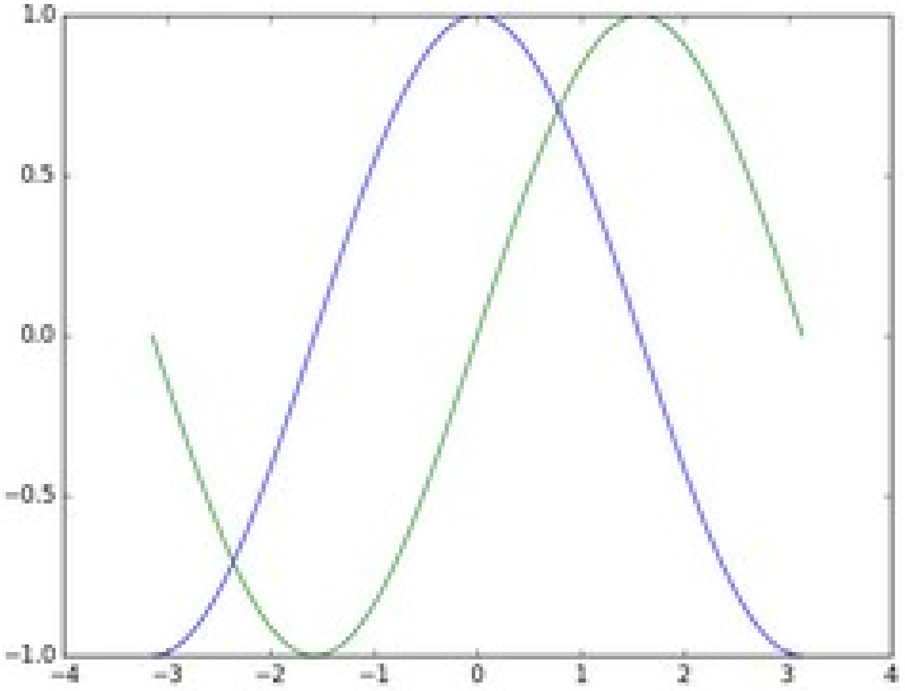
- plot() is used to plot a graph given list of x coordinate and list of y coordinate



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

X = np.linspace(-np.pi, np.pi, 256, endpoint=True)
C, S = np.cos(X), np.sin(X)

plt.plot(X, C)
plt.plot(X, S)

plt.show()
```

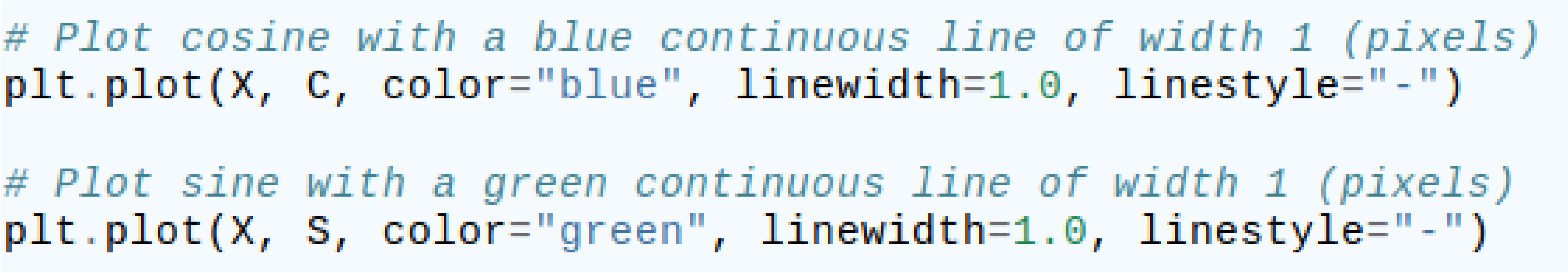
To display the plot

- Must call plt.show() function to actually display the plot



Customising the plot

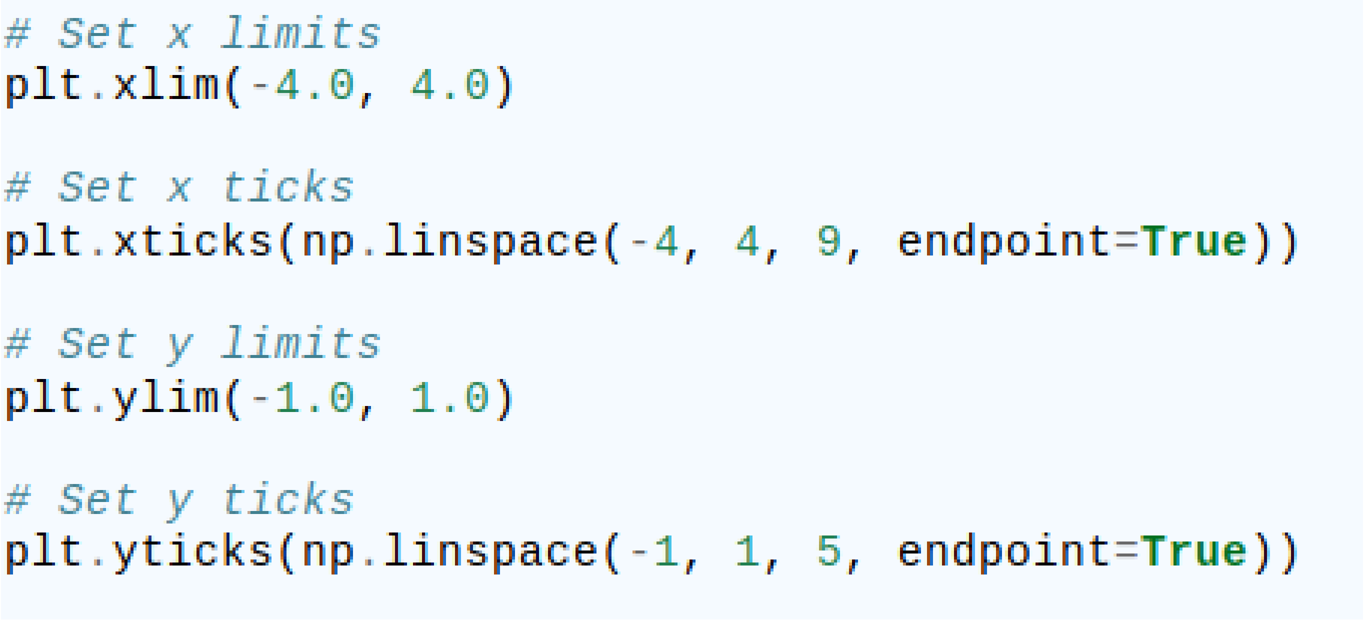
- Setting color using **color** keyword argument
- Setting line width using **linewidth**
- Setting linestyle to select line style



```
# Plot cosine with a blue continuous line of width 1 (pixels)
plt.plot(X, C, color="blue", linewidth=1.0, linestyle="-")

# Plot sine with a green continuous line of width 1 (pixels)
plt.plot(X, S, color="green", linewidth=1.0, linestyle="-")
```

More customisation

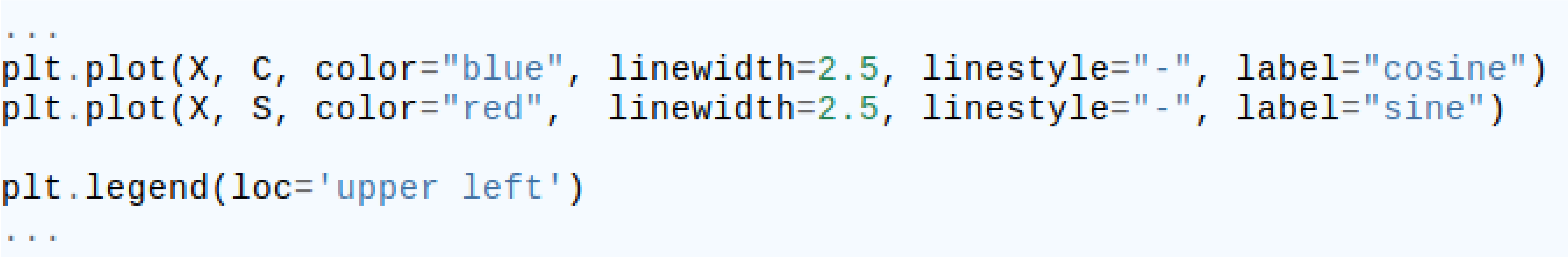
- xlim(), ylim() to get x-axis/y-axis limits
- xticks(), yticks() to set axis ticks



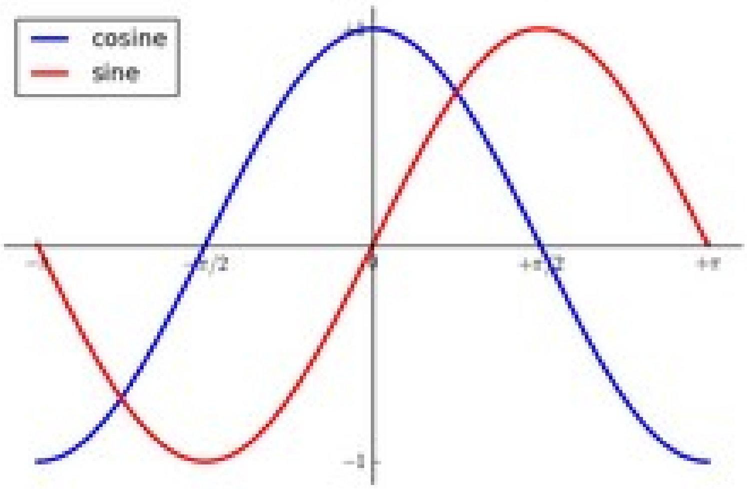
```
# Set x limits
plt.xlim(-4.0, 4.0)

# Set x ticks
plt.xticks(np.linspace(-4, 4, 9, endpoint=True))

# Set y limits
plt.ylim(-1.0, 1.0)

# Set y ticks
plt.yticks(np.linspace(-1, 1, 5, endpoint=True))
```

Adding legend

- Legend can be added using **label** keyword

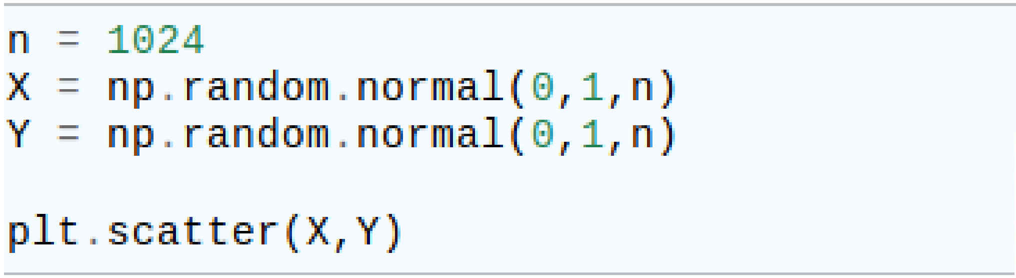


```
...
plt.plot(X, C, color="blue", linewidth=2.5, linestyle="-", label="cosine")
plt.plot(X, S, color="red",  linewidth=2.5, linestyle="-", label="sine")

plt.legend(loc='upper left')
...
```

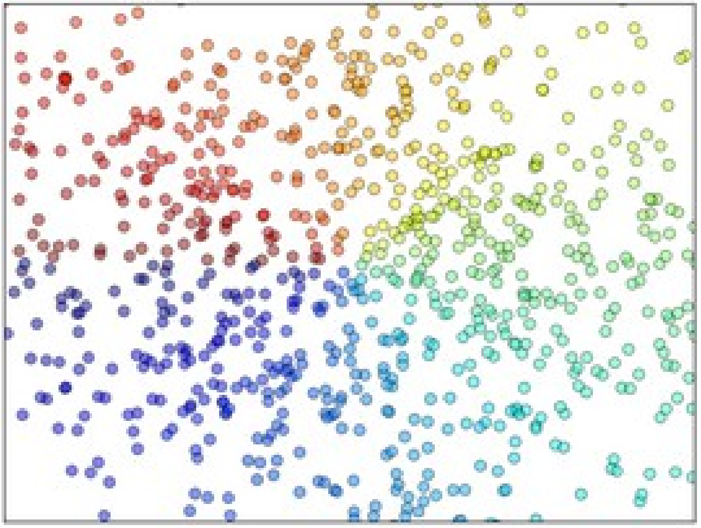


Other types of plots

- Scatter plot

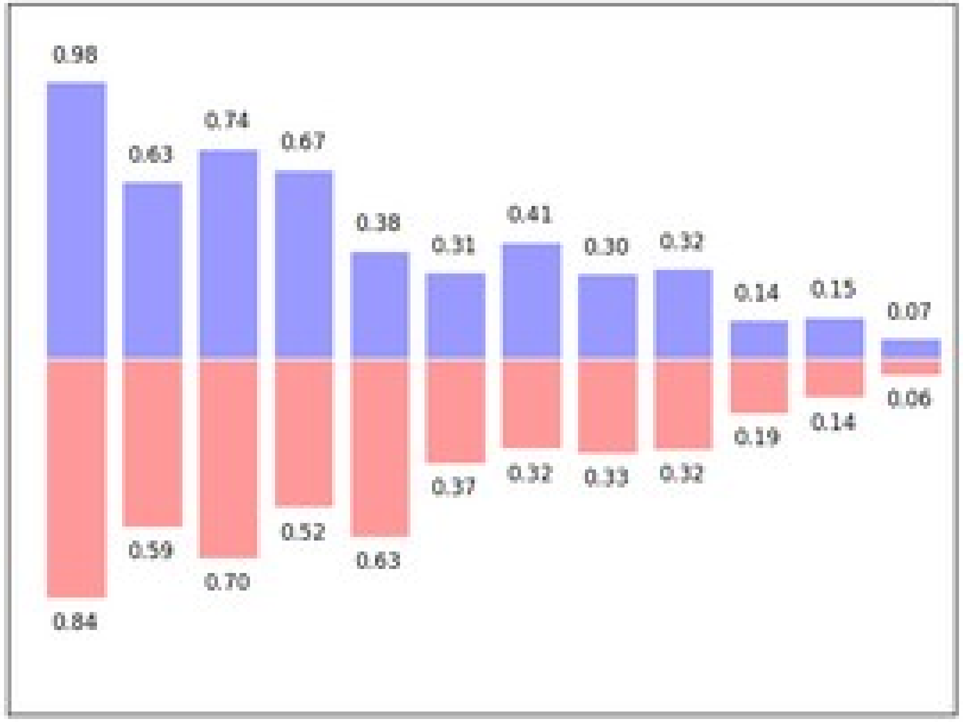


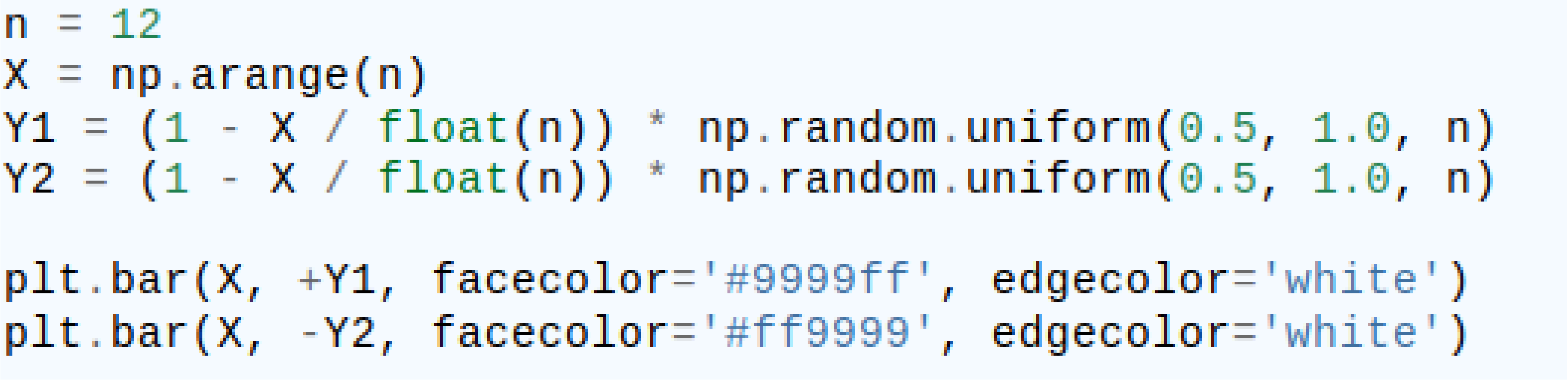
```
n = 1024
X = np.random.normal(0,1,n)
Y = np.random.normal(0,1,n)

plt.scatter(X,Y)
```



Barplot

- bar() function

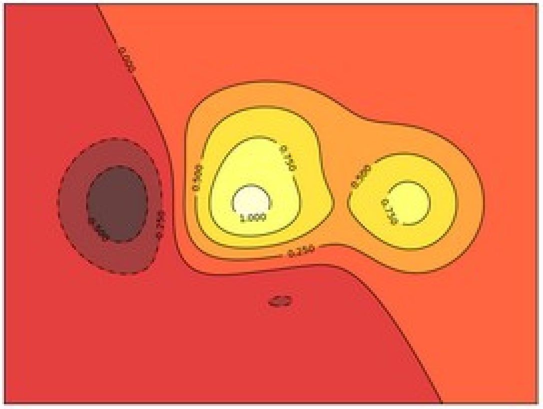




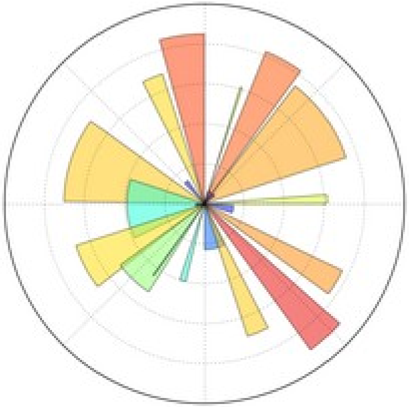
```
n = 12
X = np.arange(n)
Y1 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
Y2 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)

plt.bar(X, +Y1, facecolor='#9999ff', edgecolor='white')
plt.bar(X, -Y2, facecolor='#ff9999', edgecolor='white')
```

And many more

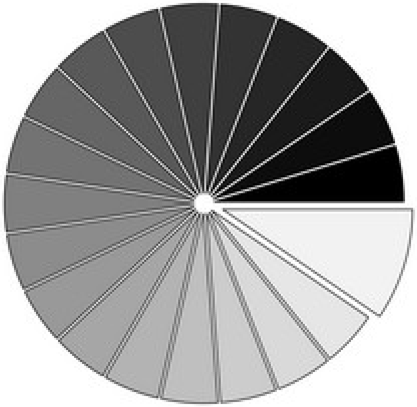
Contour



Polar axis



Piechart



Pandas

- Pandas is Python package for data analysis.
- adds data structures and tools designed to work with table-like data (similar to Series and Data Frames in R)
- provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc.
- allows handling missing data

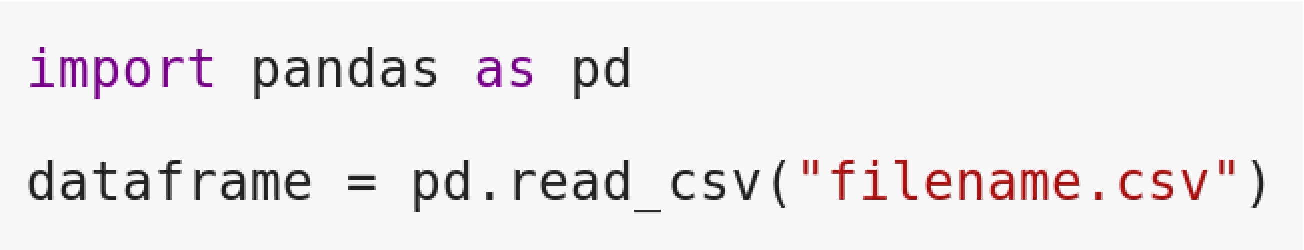
Pandas: Essential Concepts

- A Series is a named Python list (dict with list as value).
  { ‘grades’ : [50,90,100,45] }

- A DataFrame is a dictionary of Series (dict of series):
  {   { ‘names’ : [‘bob’,’ken’,’art’,’joe’]}
    { ‘grades’ : [50,90,100,45] }
  }

Reading CSV

- Importing the library using import keyword
- Reading csv using read_csv() function

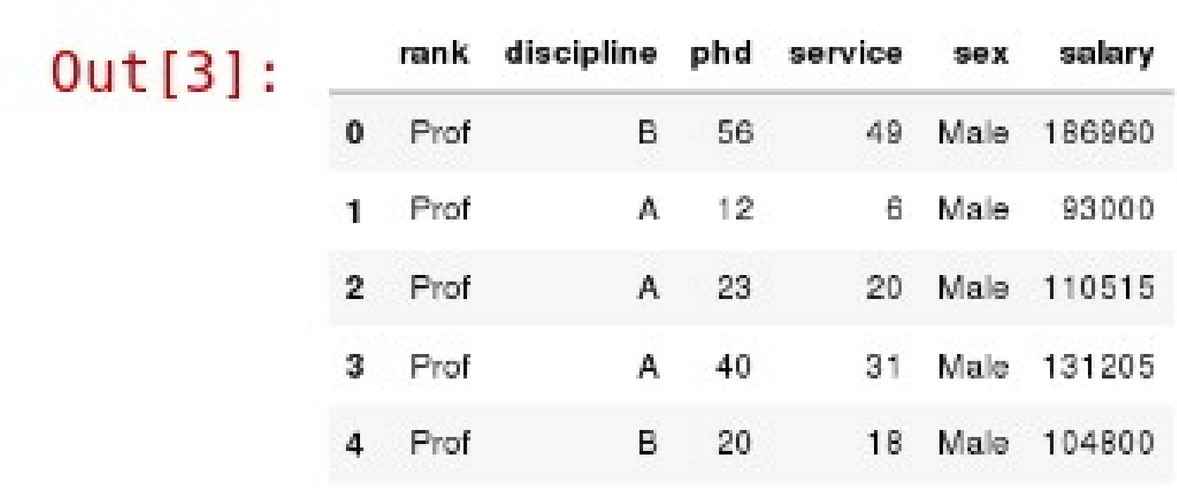


```python
import pandas as pd

dataframe = pd.read_csv("filename.csv")
```

Exploring data

- First n items
- Or last n items

```
dataframe.head(n)
dataframe.tail(n)
```



Out[3]:

	rank	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
1	Prof	A	12	6	Male	93000
2	Prof	A	23	20	Male	110515
3	Prof	A	40	31	Male	131205
4	Prof	B	20	18	Male	104800

Data Frames attributes

df.attribute	description
dtypes	list the types of the columns
columns	list the column names
axes	list the row labels and column names
ndim	number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy representation of the data

Data frame methods

df.method()	**description**
head([n]), tail([n])	first/last n rows
describe()	generate descriptive statistics (for numeric columns only)
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values

Selecting a column in a Data Frame

- *Method 1:* Subset the data frame using column name:

```
df['sex']
```

- *Method 2*: Use the column name as an attribute:

```
df.sex
```

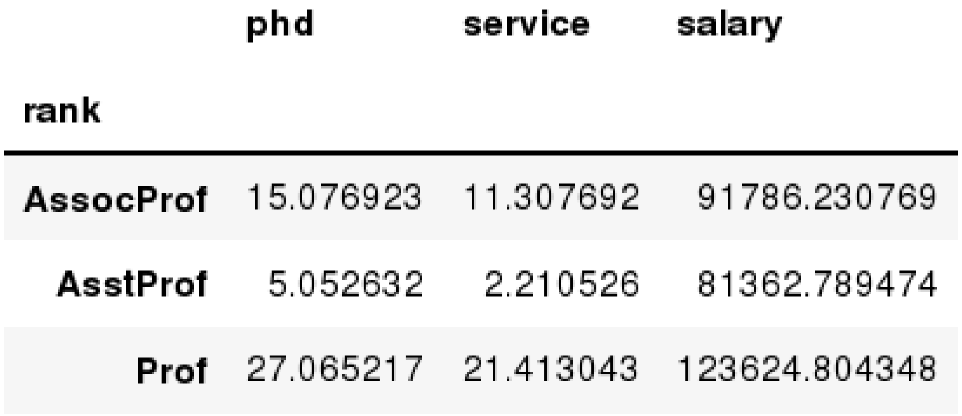
Data frame groupby()

- Using "group by" method we can:
- Split the data into groups based on some criteria
- Calculate statistics (or apply a function) to each group

Example

```
In [ ]:  #Group data using rank
         df_rank = df.groupby(['rank'])
```

```
In [ ]:  #Calculate mean value for each numeric column per each group
         df_rank.mean()
```



	phd	service	salary
rank			
AssocProf	15.076923	11.307692	91786.230769
AsstProf	5.052632	2.210526	81362.789474
Prof	27.065217	21.413043	123624.804348

Data Frame: filtering

- To subset the data we can apply Boolean indexing. This indexing is commonly known as a filter. For example if we want to subset the rows in which the salary value is greater than $120K:

```
#Calculate mean salary for each professor rank:
df_sub = df[ df['salary'] > 120000 ]
```

Dataframe slicing

- There are a number of ways to subset the Data Frame:
  - one or more columns
  - one or more rows
  - a subset of rows and columns

Slicing

- When selecting one column, it is possible to use single set of brackets, but the resulting object will be a Series (not a DataFrame):

```
#Select column salary:
df['salary']
```

- When we need to select more than one column and/or make the output to be a DataFrame, we should use double brackets:

```
#Select column salary:
df[['rank','salary']]
```

Selecting rows

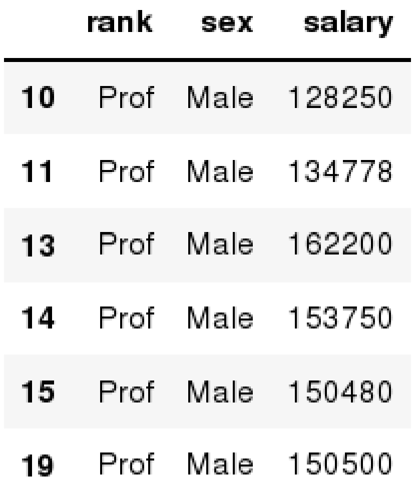
- If we need to select a range of rows, we can specify the range using ":"

```
#Select rows by their position:
df[10:20]
```

Method loc()

- If we need to select a range of rows, using their labels we can use method loc():

```
#Select rows by their labels:
df_sub.loc[10:20,['rank','sex','salary']]
```

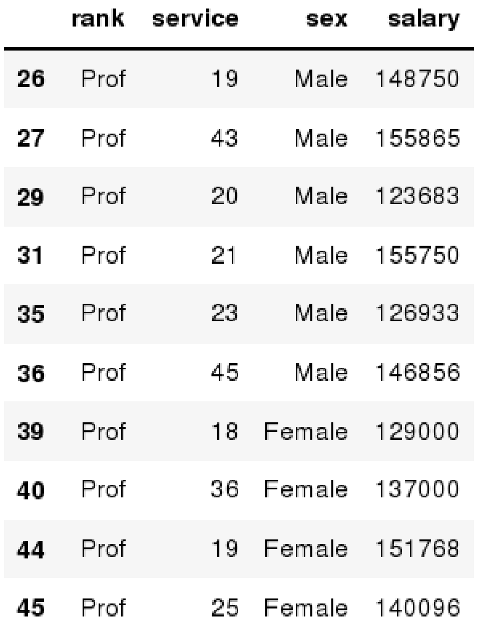


	rank	sex	salary
10	Prof	Male	128250
11	Prof	Male	134778
13	Prof	Male	162200
14	Prof	Male	153750
15	Prof	Male	150480
19	Prof	Male	150500

Method iloc()

- If we need to select a range of rows and/or columns, using their positions we can use method iloc():

```
#Select rows by their labels:
df_sub.iloc[10:20,[0, 3, 4, 5]]
```



	rank	service	sex	salary
26	Prof	19	Male	148750
27	Prof	43	Male	155865
29	Prof	20	Male	123683
31	Prof	21	Male	155750
35	Prof	23	Male	126933
36	Prof	45	Male	146856
39	Prof	18	Female	129000
40	Prof	36	Female	137000
44	Prof	19	Female	151768
45	Prof	25	Female	140096

References

- Python for Data Analysis by Katia Oleinik
- Data Analysis with Pandas – IST256 url: ist256.syr.edu/content/12/Data-Analysis.pptx
- http://scipy-lectures.org/