

Important Python Packages

Prof. Gheith Abandah

Reference: Vanderplas, Jacob T. *A Whirlwind Tour of Python*. O'Reilly Media, 2016.

Outline

- Modules and Packages
- Jupyter Notebooks
- NumPy: Numerical Python
- Pandas: Labeled Column-Oriented Data
- Matplotlib: MATLAB-style Scientific Visualization
- SciPy: Scientific Python
- Other Data Science Packages
- Exercises

Modules and Packages

- For loading **built-in** and **third-party** modules, Python provides the **import** statement.
- Options:
 - Explicit import
 - Import with alias
 - Specific content import

```
import math
math.cos(math.pi)
-1.0
```

```
import numpy as np
np.cos(np.pi)
-1.0
```

```
from math import cos, pi
cos(pi)
-1.0
```

Built-in Modules

Module	Description
os and sys	Tools for interfacing with the operating system, including navigating file directory structures and executing shell commands
math and cmath	Mathematical functions and operations on real and complex numbers
itertools	Tools for constructing and interacting with iterators and generators
functools	Tools that assist with functional programming
random	Tools for generating pseudorandom numbers
pickle	Tools for saving objects to and loading objects from disk
json and csv	Tools for reading JSON-formatted and CSV-formatted files
urllib	Tools for doing HTTP and other web requests

Reference: <https://docs.python.org/3/library/>

Importing from Third-Party Modules

- Python has excellent ecosystem of free third-party modules.
- They can be imported just as the built-in modules, but must first be installed on your system.
- The standard registry for such modules is the **Python Package Index (PyPI)**:
<http://pypi.python.org/>
- Python comes with **pip** (pip installs packages).
**\$ pip install jupyter numpy scipy
pandas matplotlib scikit-learn**

Outline

- Modules and Packages
- Jupyter Notebooks
- NumPy: Numerical Python
- Pandas: Labeled Column-Oriented Data
- Matplotlib: MATLAB-style Scientific Visualization
- SciPy: Scientific Python
- Other Data Science Packages
- Exercises

Jupyter Notebook

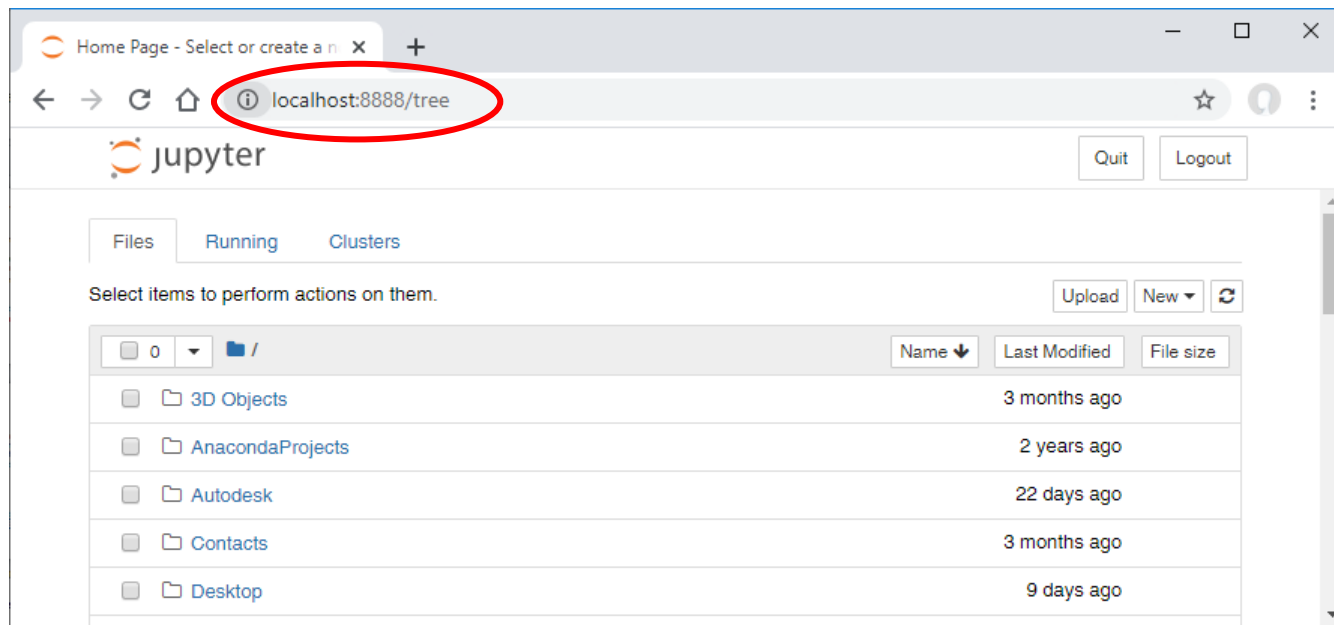
- The Jupyter Notebook is an open-source **web application** that allows you to create and share documents that contain live code, equations, visualizations and narrative text.
- Website: <http://jupyter.org/>
- Tutorial on Jupyter's website: <https://jupyter-notebook.readthedocs.io/en/stable/notebook.html>

Starting Jupyter Notebook

- To start Jupyter notebooks server from your OS command prompt enter:

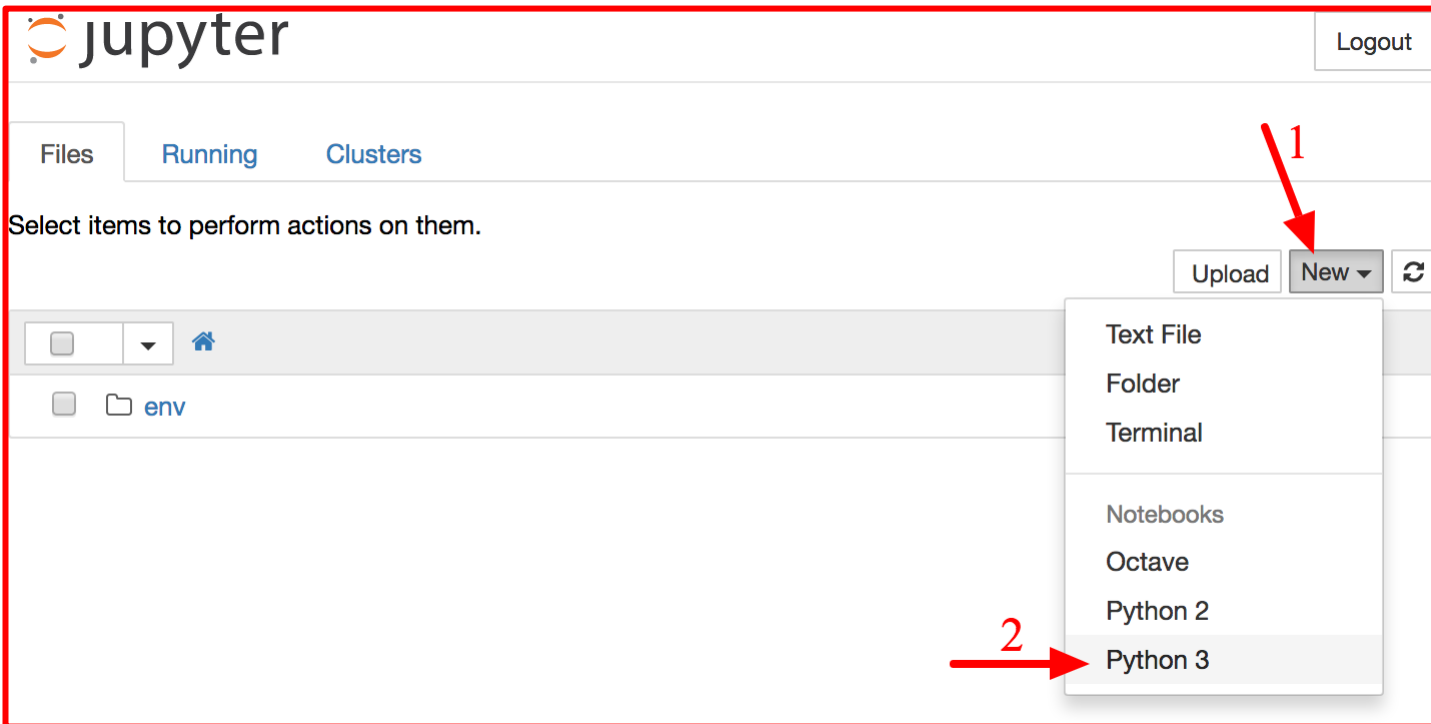
C:\>jupyter notebook

- Then browse to <http://localhost:8888/>.



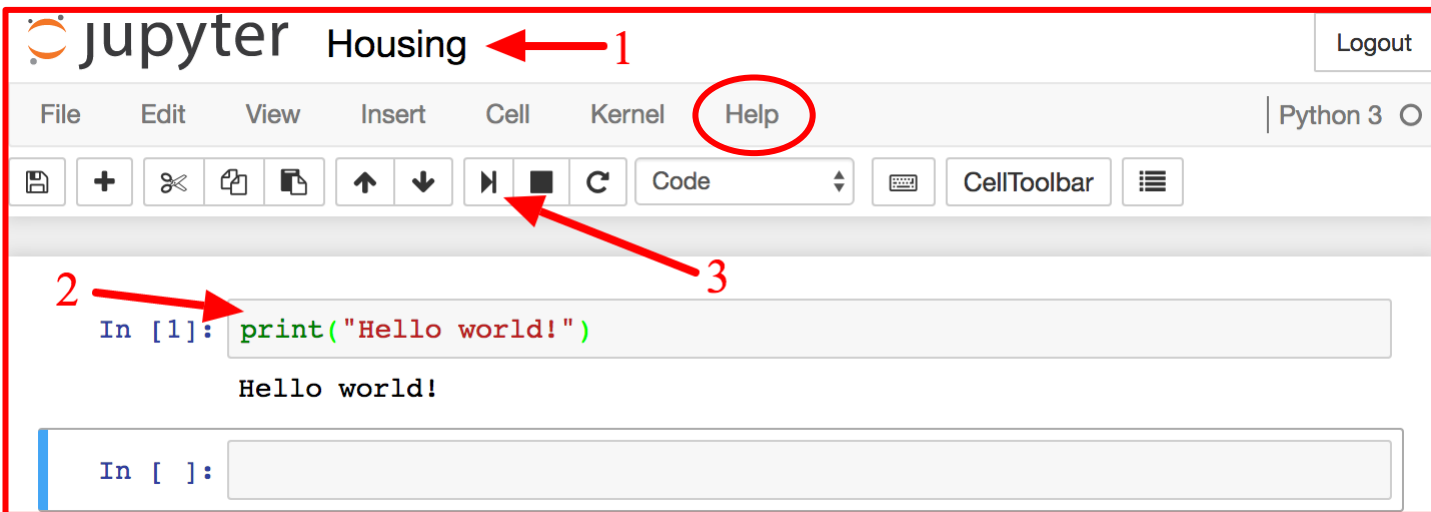
Creating Python Notebook

- Click on the **New** button and select the appropriate Python version.



Working with Notebooks

- A notebook contains a list of cells. Each cell can contain executable code or formatted text.
- To change the notebook name, click on the title bar.
- Type `print("Hello world!")` in the first cell, and click on the play button. The result is displayed below the cell.
- Check the User Interface Tour from Jupyter's Help menu.



Outline

- Modules and Packages
- Jupyter Notebooks
- NumPy: Numerical Python
- Pandas: Labeled Column-Oriented Data
- Matplotlib: MATLAB-style Scientific Visualization
- SciPy: Scientific Python
- Other Data Science Packages
- Exercises

NumPy: Numerical Python

- NumPy is the fundamental package for **fast scientific computing** with Python. It contains:
 - A powerful N-dimensional array object
 - Sophisticated (broadcasting) functions
 - Useful linear algebra, Fourier transform, and random number capabilities
- Website: <http://www.numpy.org/>
- Also, check the tutorial on Learn Python: [https://www.learnpython.org/en/Numpy Arrays](https://www.learnpython.org/en/Numpy_Arrays)

ndarray Creation

- Vectors can be created using **arange()**.
- Vectors and arrays can be created using **zeros()** and **ones()**.

```
>>> import numpy as np
>>> x = np.arange(1, 5)
>>> x
array([1, 2, 3, 4])
```

```
>>> y = np.zeros(5)
>>> y
array([0., 0., 0., 0., 0.])
>>> z = np.ones((3, 2))
>>> z
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```

ndarray Operations

- NumPy allows efficient elementwise operations.

```
>>> x + x
array([2, 4, 6, 8])
>>> x ** 2
array([ 1,  4,  9, 16], dtype=int32)
```

- Also includes:

- Reshape
- Transpose
- Inverse
- etc.

```
>>> M = x.reshape((2, 2))
>>> M
array([[1, 2],
       [3, 4]])
>>> M.T
array([[1, 3],
       [2, 4]])
>>> np.linalg.inv(M)
array([[ -2. ,  1. ],
       [ 1.5, -0.5]])
```

Outline

- Modules and Packages
- Jupyter Notebooks
- NumPy: Numerical Python
- Pandas: Labeled Column-Oriented Data
- Matplotlib: MATLAB-style Scientific Visualization
- SciPy: Scientific Python
- Other Data Science Packages
- Exercises

Pandas: Labeled Column-Oriented Data

- Pandas is high-performance, easy-to-use **data structures** and **data analysis tools**. It contains:
 - A set of labeled array data structures (Series and DataFrame)
 - Index objects enabling both simple axis indexing and multi-level / hierarchical axis indexing
 - Input/Output tools: loading tabular data from flat files (CSV, delimited, Excel 2003), and saving and loading pandas objects
- Website: <https://pandas.pydata.org/>
- Also, check the tutorial on Learn Python: [https://www.learnpython.org/en/Pandas Basics](https://www.learnpython.org/en/Pandas_Basics)

Creating Data Frame from Dictionary

```
dict = {"country": ["Brazil", "Russia", "India", "China"],  
        "capital": ["Brasilia", "Moscow", "New Dehli", "Beijing"],  
        "area": [8.516, 17.10, 3.286, 9.597],  
        "population": [200.4, 143.5, 1252, 1357] }
```

```
import pandas as pd  
brics = pd.DataFrame(dict)  
print(brics)
```

	country	capital	area	population
0	Brazil	Brasilia	8.516	200.4
1	Russia	Moscow	17.100	143.5
2	India	New Dehli	3.286	1252.0
3	China	Beijing	9.597	1357.0

Creating Data Frame from File

```
# Import pandas as pd
```

```
import pandas as pd
```

```
# Import the cars.csv data: cars
```

```
cars = pd.read_csv('cars.csv')
```

```
# Print out cars
```

```
print(cars)
```

```
   Unnamed: 0  cars_per_cap  country  drives_right
0           US           809  United States         True
1           AUS           731   Australia         False
2           JAP           588     Japan         False
3           IN            18     India         False
4           RU           200     Russia         True
5           MOR            70   Morocco         True
6           EG            45     Egypt         True
```

Data Frame Operations

- Data Frames have many powerful operations such as **sum()**, **info()** and **describe()**.

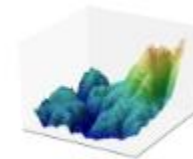
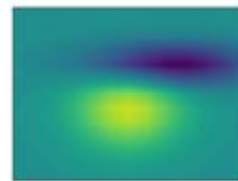
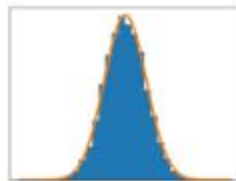
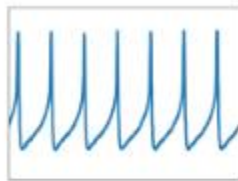
```
>>> brics['population'].sum()
2952.9
>>> brics.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4 entries, 0 to 3
Data columns (total 4 columns):
#   Column      Non-Null Count  Dtype
---  -
0   country     4 non-null      object
1   capital     4 non-null      object
2   area        4 non-null      float64
3   population  4 non-null      float64
dtypes: float64(2), object(2)
memory usage: 256.0+ bytes
>>> brics.describe()
           area  population
count  4.000000    4.000000
mean    9.624750   738.225000
std     5.694711   655.693223
min     3.286000   143.500000
25%    7.208500   186.175000
50%    9.056500   726.200000
75%   11.472750  1278.250000
max   17.100000  1357.000000
```

Outline

- Modules and Packages
- Jupyter Notebooks
- NumPy: Numerical Python
- Pandas: Labeled Column-Oriented Data
- Matplotlib: MATLAB-style Scientific Visualization
- SciPy: Scientific Python
- Other Data Science Packages
- Exercises

Matplotlib: MATLAB-style Scientific Visualization

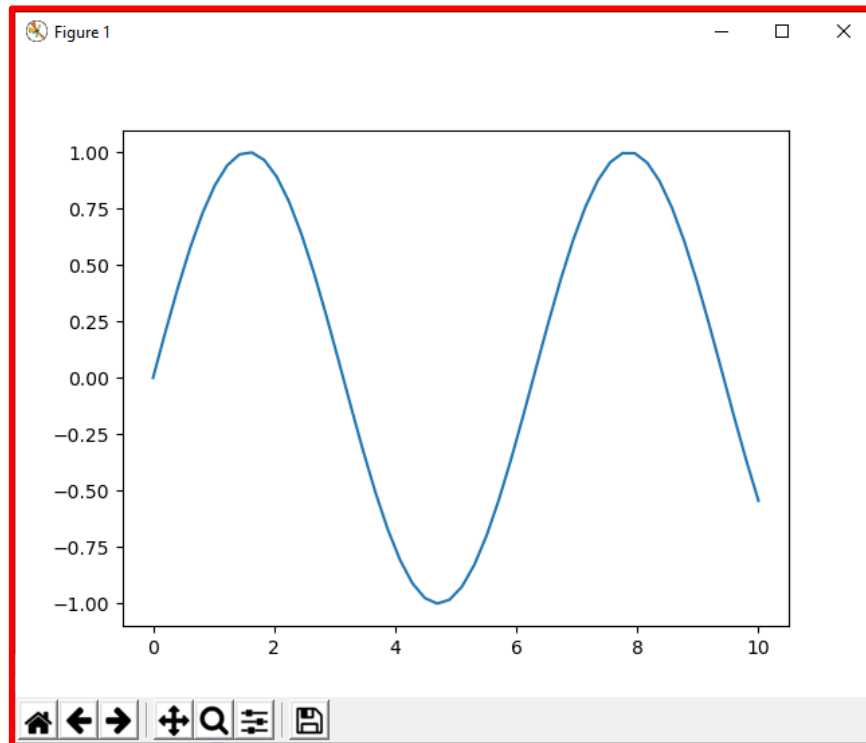
- Matplotlib is a Python **plotting library** which produces publication quality figures in a variety of hardcopy formats.



- Website: <https://matplotlib.org/>
- Also, check the tutorial package website: <https://matplotlib.org/tutorials/introductory/pyplot.html>

Matplotlib Example

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 10) # range of values from 0 to 10
y = np.sin(x) # sine of these values
plt.plot(x, y); # plot as a line
plt.show()
```



Outline

- Modules and Packages
- Jupyter Notebooks
- NumPy: Numerical Python
- Pandas: Labeled Column-Oriented Data
- Matplotlib: MATLAB-style Scientific Visualization
- **SciPy: Scientific Python**
- **Other Data Science Packages**
- Exercises

SciPy: Scientific Python

- Collection of scientific functionality that is built on NumPy.
 - **scipy.fftpack** Fast Fourier transforms
 - **scipy.integrate** Numerical integration
 - **scipy.interpolate** Numerical interpolation
 - **scipy.linalg** Linear algebra routines
 - **scipy.optimize** Numerical optimization of functions
 - **scipy.sparse** Sparse matrix storage and linear algebra
 - **scipy.stats** Statistical analysis routines
- Website: <https://www.scipy.org/>

Outline

- Modules and Packages
- Jupyter Notebooks
- NumPy: Numerical Python
- Pandas: Labeled Column-Oriented Data
- Matplotlib: MATLAB-style Scientific Visualization
- SciPy: Scientific Python
- **Other Data Science Packages**
- **Exercises**

Other Data Science Packages

- [Scikit-Learn](#) for machine learning
- [Scikit-Image](#) for image analysis
- [StatsModels](#) for statistical modeling
- [AstroPy](#) for astronomy and astrophysics
- [NiPy](#) for neuro-imaging
- and many, many more.

Outline

- Modules and Packages
- Jupyter Notebooks
- NumPy: Numerical Python
- Pandas: Labeled Column-Oriented Data
- Matplotlib: MATLAB-style Scientific Visualization
- SciPy: Scientific Python
- Other Data Science Packages
- Exercises

Exercises from

<http://www.practicepython.org/>

- Ex1: Create a program that asks the user to enter their name and their age. Print out a message addressed to them that tells them the year that they will turn 100 years old.
- Ex3: Take a list, say for example this one: `a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]` and write a program that prints out all the elements of the list that are less than 5.
- Ex5: Take two lists, say for example these two: `a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]` and `b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]` and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.
- Ex28: Implement a function that takes as input three variables, and returns the largest of the three. Do this without using the Python `max()` function!

Exercises from

<https://www.w3resource.com/python-exercises/>

- Class 9: Write a Python class which has two methods `get_String` and `print_String`. `get_String` accept a string from the user and `print_String` print the string in upper case.
- Class 10: Write a Python class named `Rectangle` constructed by a length and width and a method which will compute the area of a rectangle.

Exercises from

<https://www.w3resource.com/python-exercises/>

- **NumPy 3:** Create a 3x3 matrix with values ranging from 2 to 10.
- **NumPy 73:** Write a Python program to create an array of (3, 4) shape, multiply every element value by 3 and display the new array.
- **Pandas DataFrame 4 and 5:** Write a Python program to get the first 3 rows and the 'name' and 'score' columns from the following DataFrame.

Sample Python dictionary data and list labels:

```
exam_data = {'name': ['Anastasia', 'Dima', 'Katherine',  
                    'James', 'Emily', 'Michael', 'Matthew', 'Laura', 'Kevin',  
                    'Jonas'],  
            'score': [12.5, 9, 16.5, np.nan, 9, 20, 14.5, np.nan, 8,  
                    19],  
            'attempts': [1, 3, 2, 3, 2, 3, 1, 1, 2, 1],  
            'qualify': ['yes', 'no', 'yes', 'no', 'no', 'yes', 'yes',  
                    'no', 'no', 'yes']}  
labels = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
```

Summary

- Modules and Packages
- Jupyter Notebooks
- NumPy: Numerical Python
- Pandas: Labeled Column-Oriented Data
- Matplotlib: MATLAB-style Scientific Visualization
- SciPy: Scientific Python
- Other Data Science Packages
- Exercises