

Title of Work.	No. of Sentences.	Average Number of Words per Sentence				
		First 100.	Second 100.	Third 100.	Fourth 100.	Fifth 100.
'Der Buergergeneral'.....	500	6.7	5.1	4.5	3.9	4.7
'Gesetz v. Berlichingen' *....	2500	8.7	9.2	8.7	7.8	8.1
'Letters to Frau v. Stein'.....	500	13.5	12.5	11.3	11.2	12.4
'Faust': First Part.....	500	14.6	15.2	13.6	13.0	12.0
'Faust': Second Part.....	500	16.3	17.2	15.5	12.3	16.5
'Liebe Finecke Fuchs'.....	500	18.5	16.5	16.9	14.8	16.6
'Leiden d. j. Werthers'....	500	20.7	20.9	19.1	22.7	18.2
'Miliaenische Reise: Rom'....	500	23.1	23.7	22.7	21.5	22.7
'Wallenstein'.....	500	21.9	22.3	22.7	25.1	24.5

Introduction to Data Analysis with Python

Jordi Vitrià, PhD



Introduction to Data Analysis with Python

Syllabus

Data Science Steps & Python Data Science Stack

Data Science Plumbing & Agile Data Science

Why Python?

What is IPython/Jupyter?

Data Science Steps

What do I want?
Does it have sense?

What are my data
sources? How reliable
are they?

How do I develop an
understanding of the
content of my data?

What are the key
relationships in my
data?

How do I develop an
understanding of the
content of my data?

What are the likely
future outcomes?

Are my expectations
fulfilled?

Question

Acquire

Describe

Discover

Analyze

Predict

Evaluate

Acquire: How do I get my data?

- Web Scraping.
- Data Base queries.
- Access to bulk data stores.

Processing: How I do clean and separate my data?

- Identification: filter data.
- Outliers.
- Imputation: missing value processing.
- Reduction: dimensionality reduction.
- Normalization: duplicates, ranges, format, coordinates, units, etc.
- Feature extraction.

Aggregation: How do I collect and summarize my data?

- Basic Statistics: mean, std, box plots, scatter plots, counts, etc.
- Distribution fitting.
- Feature aggregation.

Enrichment: How do I add more information to my data?

- Feature engineering.
- Search for additional data sources.

Discover: What are the key relationships in my data?

- Clustering (How do I segment the data to find natural groupings?)
- Visualization (Are there unexpected relationships?)

Analyze: How do I model my data?

- Variable selection (How do I determine important variables?)
- Probabilistic modeling (How are my variables related?)

Predict: What are the likely future outcomes?

- Regression (How do I predict the future?)
- Classification (How do I predict a category?)
- Recommendation (How do I predict relevant conditions?)

Evaluate: Are the outcomes generic and robust?

- Statistical Testing.
- Model performance.

Python
Data Stack

Scrapy
PyDB
PyMongo
Numpy
Pandas
SciPy
SciKit Learn

Matplotlib
SymPy
IPython
Cython
Bokeh
Blaze
PySpark
StatsModels
Shogun
PyMC
seaborn
PyTables
cvskit
sqlite3
plotly
NetworkX
nltk
gensim
twython

Data Science Plumbing

Agile Development & Model Deployment

Logs from servers, sensors, transactions, etc. Events must be serialized in a common format.

Events are collected from different sources and aggregated to bulk storage.

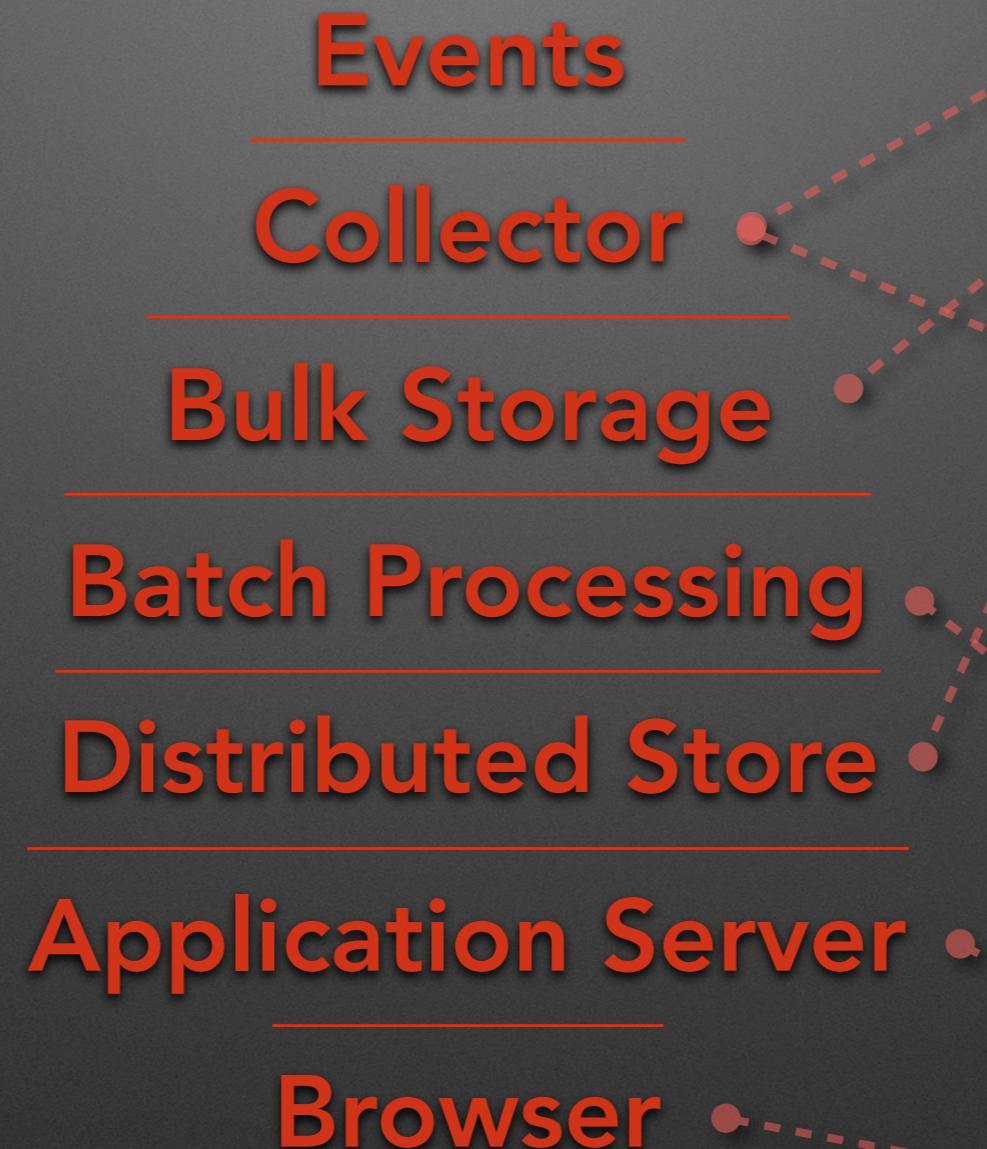
This is a filesystem capable of parallel access by concurrent processes.

Data processing with scripting languages with out-of-core processing capabilities.

These are multinode stores to publish data for consumption by web apps.

Web application server.

Presentation of the data as an interactive experience + story telling.



Why Python?

- Python is a modern, open source, **object-oriented programming** language, created by a Dutch programmer, Guido van Rossum, in 1991.
- Officially, it is an **interpreted scripting language**. There are several implementations, being the most used the one implemented in C.
- It offers the power and flexibility of lower level (i.e. compiled) languages, without the steep learning curve, and without most of the associated debugging pitfalls.
- The language is very **clean and readable**, and it is available for almost every modern computing platform.
- It is widely used: Google, DropBox, Deutsche Borse, NASA, etc.

Why Python?

In Data Science, **programming is mostly about discovering solutions by writing code and using a language to express them.**

Python offers a number of advantages to data scientists:

- Powerful and easy to use.
- Anything that can be coded in C, FORTRAN, or Java can be done in Python, almost always in fewer lines of code, and with fewer debugging headaches.
- Its standard library is extremely rich, including modules for string manipulation, regular expressions, file compression, mathematics, scientific programming, profiling and debugging.
- Unnecessary language constructs, such as END statements and brackets are absent, making the code efficient and easy to read.
- Python is object-oriented, which allows data structures to be abstracted in a natural way.

Why Python?

- It is interactive
 - Python may be run interactively on the command line. Commands may be entered serially followed by the Return key.
 - This is useful for exploratory programming.
- It is extensible
 - Python is often referred to as a “glue” language, meaning that it is a useful in a mixed-language environment. C or FORTRAN code can be compiled directly into Python programs.
 - Sometimes, it can be slow relative to compiled languages. In many cases this performance deficit is due to a short loop of code that runs thousands or millions of times. Such bottlenecks may be easily removed by coding a function in C or Cython that can be compiled into a Python module.

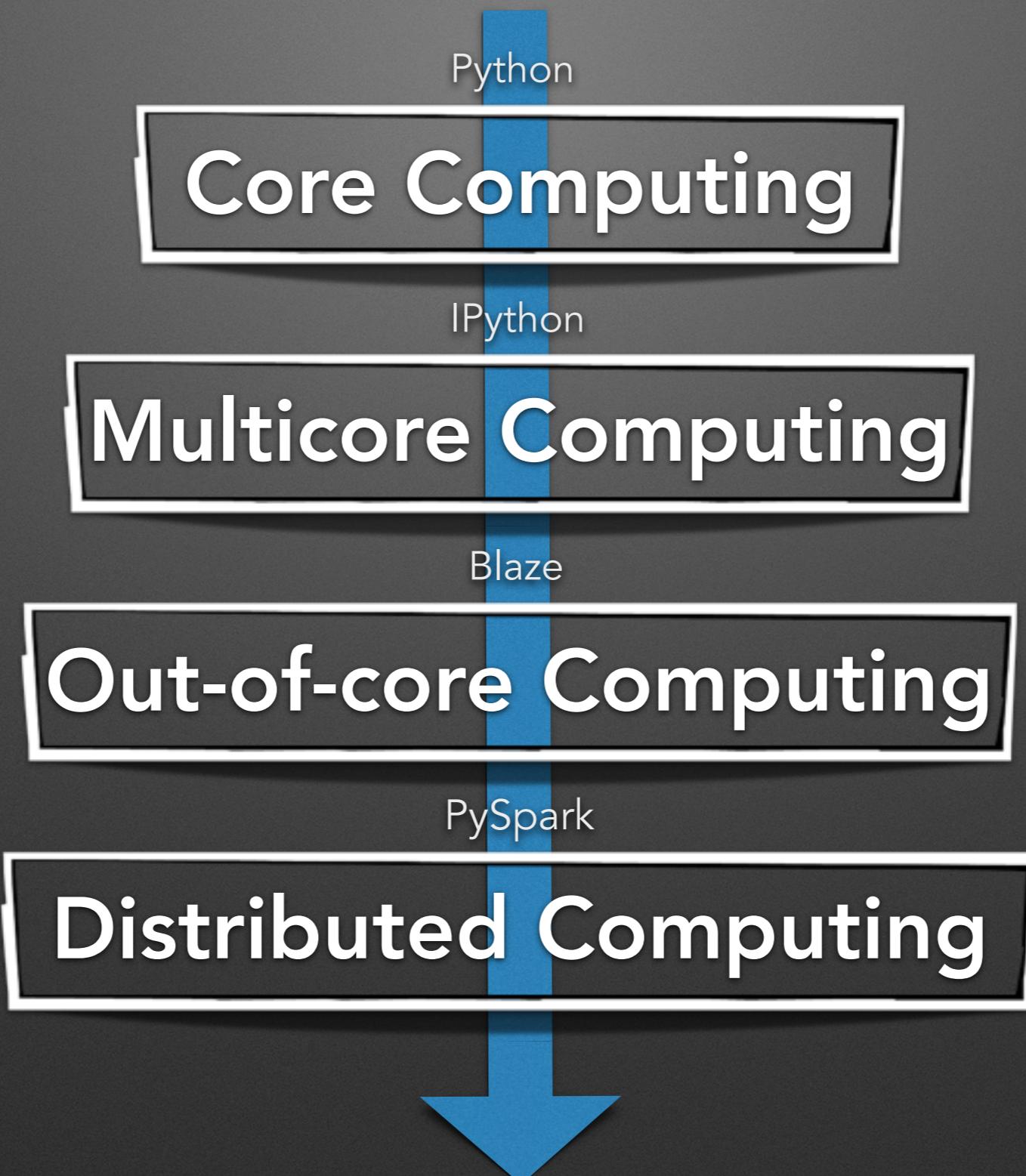
Why Python?

- There is a vast body of Python third party modules:
 - **NumPy**: Numerical Python (NumPy) is a set of extensions that provides the ability to specify and manipulate array data structures.
 - **SciPy**: An open source library of scientific tools for Python, SciPy supplements the NumPy module. SciPy includes modules for graphics and plotting, optimization, integration, special functions, signal and image processing, genetic algorithms, ODE solvers, and others.
 - **Matplotlib**: Matplotlib is a python 2D plotting library which produces publication-quality figures.
 - **pandas**: A module that provides high-performance, easy-to-use data structures and data analysis tools. In particular, the DataFrame class is useful for spreadsheet-like representation and manipulation of data.
 - **IPython**: An enhanced Python shell, designed to increase the efficiency and usability of coding, testing and debugging Python.

Why Python?

- Strong points:
 - Viable alternative to SAS/Matlab/R/...
 - Uptake in the financial sector.
 - Upcoming generation of programmers with Python as a first language.
 - Python communities in HPC/Scientific Computing.
- Weak points
 - No (public) great success story.
 - Weak support.

Why Python?



More on Python

www.codecademy.com

The screenshot shows the Codecademy Python course page. At the top, there's a navigation bar with the Codecademy logo, a login button, and a sign-up button. Below the navigation, the word "Python" is prominently displayed. A sub-headline says "Learn to program in Python, a powerful language used by sites like YouTube and Dropbox." A large red "START" button is centered below the headline. At the bottom of the main section, there are three metrics: "2.5m+ enrolled students", "13 Hours estimated course time", and "Beginner required technical level".

<http://anandology.com/python-practice-book/>

The screenshot shows the Python Practice Book landing page. It features a sidebar with a search bar and a list of topics: 1. Getting Started, 2. Working with Data, 3. Modules, 4. Object Oriented Programming, 5. Iterators & Generators, and 6. Functional Programming. The main content area is titled "Python Practice Book" and includes a welcome message, an "About this Book" section, and a note about upcoming trainings.

www.python.org

The screenshot shows the Python Tutorial page from the Python documentation. The title is "The Python Tutorial". The page starts with a brief introduction: "Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms." It then discusses the availability of the interpreter and standard library, the extendability of Python, and its use as an extension language. The page also mentions the standard library and the tutorial's focus on introducing basic concepts and features.

What is Jupyter?

In this class, we will use **Jupyter notebooks** for teaching and programming assignments.

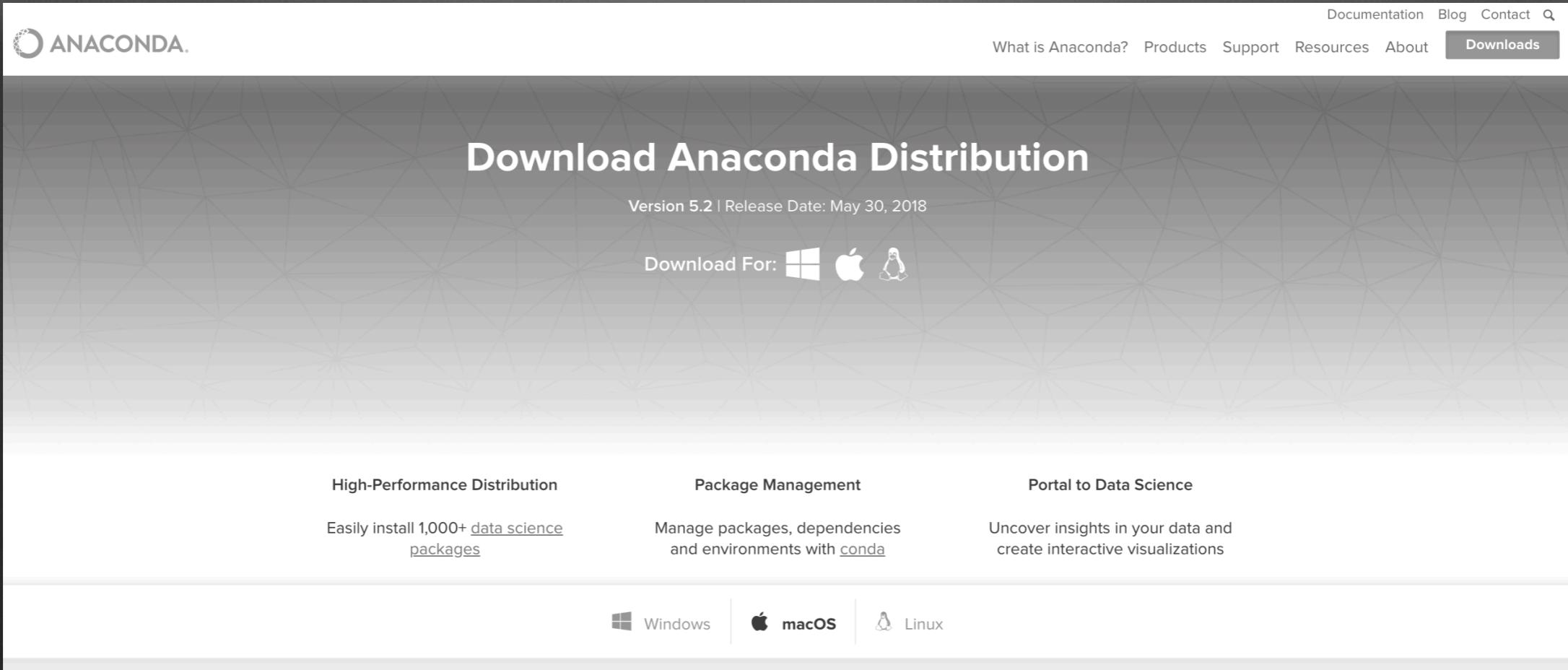
A Jupyter notebook lets you write and execute Python code in your web browser.

Jupyter notebooks make it very easy to tinker with code and execute it in bits and pieces; for this reason Jupyter notebooks are widely used in scientific computing.

Anaconda

For new users who want to get up and running with minimal effort, we suggest you to download and install Anaconda (<http://www.anaconda.com>) which provide a setup based on Python 3.7

Anaconda is a free collection of powerful packages for Python that enables large-scale data management, analysis, and visualization for Business Intelligence, Scientific Analysis, Engineering, Machine Learning, and more.



The image shows the Anaconda Distribution landing page. At the top right, there are links for Documentation, Blog, Contact, and a search icon. Below that is a navigation bar with links for What is Anaconda?, Products, Support, Resources, About, and Downloads, where 'Downloads' is highlighted. The main title 'Download Anaconda Distribution' is centered above the subtitle 'Version 5.2 | Release Date: May 30, 2018'. Below the subtitle are download links for Windows, macOS, and Linux. The page features three main sections: 'High-Performance Distribution', 'Package Management', and 'Portal to Data Science', each with a brief description and a corresponding icon.

Documentation Blog Contact 

What is Anaconda? Products Support Resources About **Downloads**

Download Anaconda Distribution

Version 5.2 | Release Date: May 30, 2018

Download For:   

High-Performance Distribution

Easily install 1,000+ [data science packages](#)

Package Management

Manage packages, dependencies and environments with [conda](#)

Portal to Data Science

Uncover insights in your data and create interactive visualizations

 Windows |  macOS |  Linux

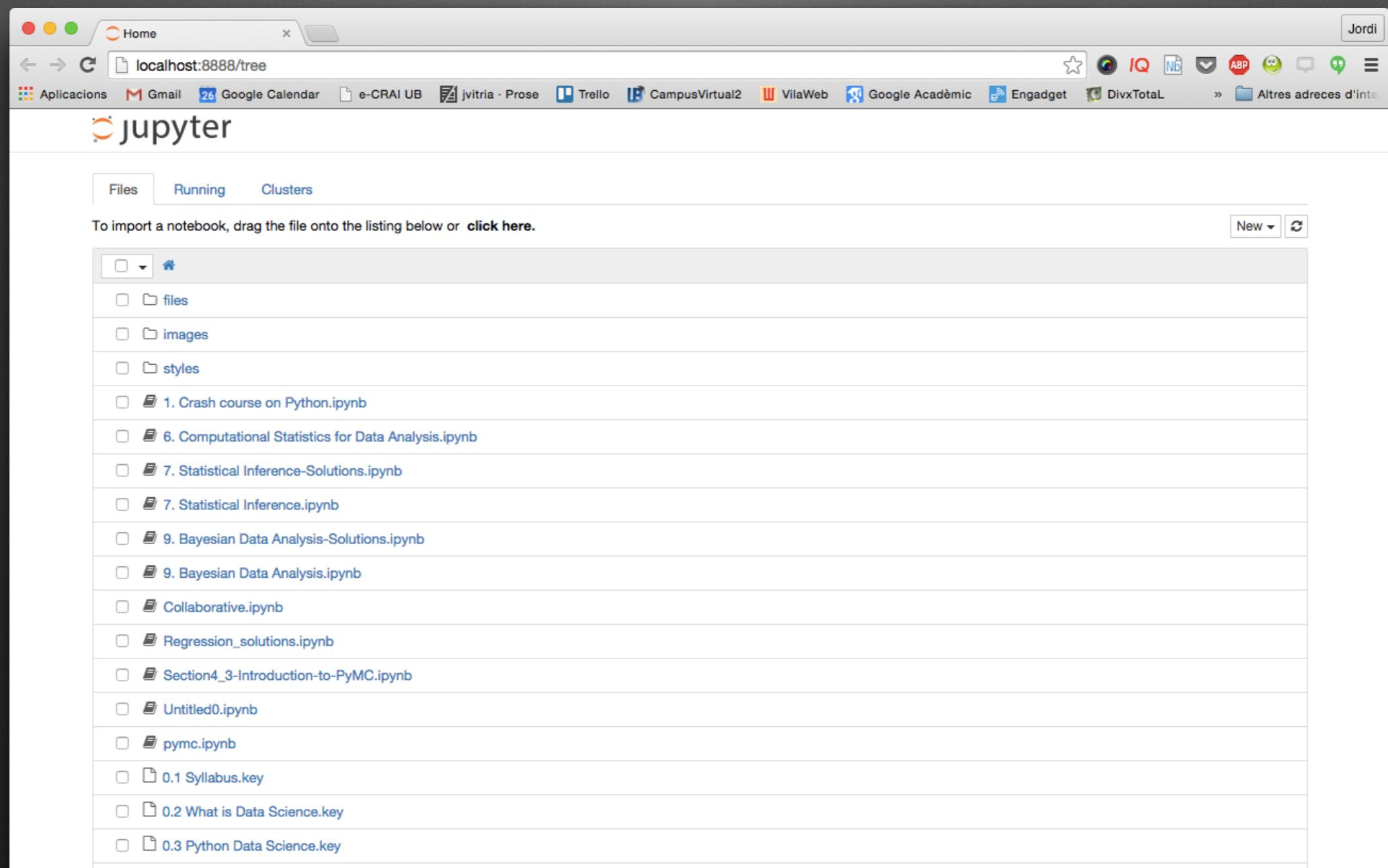
<https://www.anaconda.com/distribution/>

You need to install Anaconda (Python 3.7)

The screenshot shows the Google Colab interface. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the navigation is a toolbar with buttons for '+ CODE', '+ TEXT', 'CELL UP', 'CELL DOWN', and 'COPY TO DRIVE'. On the right side of the toolbar are 'SHARE', 'CONNECT', 'EDITING', and a user profile icon. The left sidebar has tabs for 'Table of contents', 'Code snippets', and 'Files'. The 'Cells' tab is selected, showing sub-sections: 'Code cells', 'Text cells', 'Adding and moving cells', 'Working with python', 'System aliases', 'Magics', 'Tab-completion and exploring code', 'Rich, interactive outputs', 'Integration with Drive', and 'Commenting on a cell'. The main content area starts with a section titled 'Cells' which describes a notebook as a list of cells containing explanatory text or executable code and its output. It then branches into 'Code cells' and 'Text cells'. The 'Code cells' section provides instructions for running cells using various keyboard shortcuts. A modal window titled 'Overall, how satisfied are you with Colaboratory?' is open, listing five satisfaction levels with corresponding smiley face icons: 'Very satisfied', 'Somewhat satisfied', 'Neither satisfied nor dissatisfied', 'Somewhat dissatisfied', and 'Very dissatisfied'. The bottom right corner of the modal shows the word 'Google'.

In some cases Google Colab is an interesting alternative.

Notebooks



Notebooks

The screenshot shows a Jupyter Notebook interface running in a web browser. The title bar indicates the notebook is titled "1. Crash course on Python". The browser's address bar shows the URL "localhost:8888/notebooks/1.%20Crash%20course%20on%20Python.ipynb". The notebook content starts with a section titled "Python". It explains that once IPython is installed, you are ready to perform operations. It introduces the concept of an interpreter and the difference between scripts and cells. It shows examples of simple arithmetic operations (In [3]: 3 + 4 + 9, Out[3]: 16) and range functions (In [4]: range(10), Out[4]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]). It then discusses the four parts of an operator: name, arguments, value, and side effects. Finally, it mentions that Python is a general-purpose language and requires imports for specific commands like numpy.

Once you have IPython installed, you are ready to perform all sorts of operations.

The software program that you use to invoke operators is called an **interpreter**. You enter your commands as a 'dialog' between you and the interpreter. Commands can be entered as part of a script (a text file with a list of commands to perform) or directly at the **cell**.

In order to ask to the interpreter what to do, you must **invoke** an operator:

```
In [3]: 3 + 4 + 9
Out[3]: 16
```

```
In [4]: range(10)
Out[4]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

It's helpful to think of the computation carried out by an operator as involving four parts:

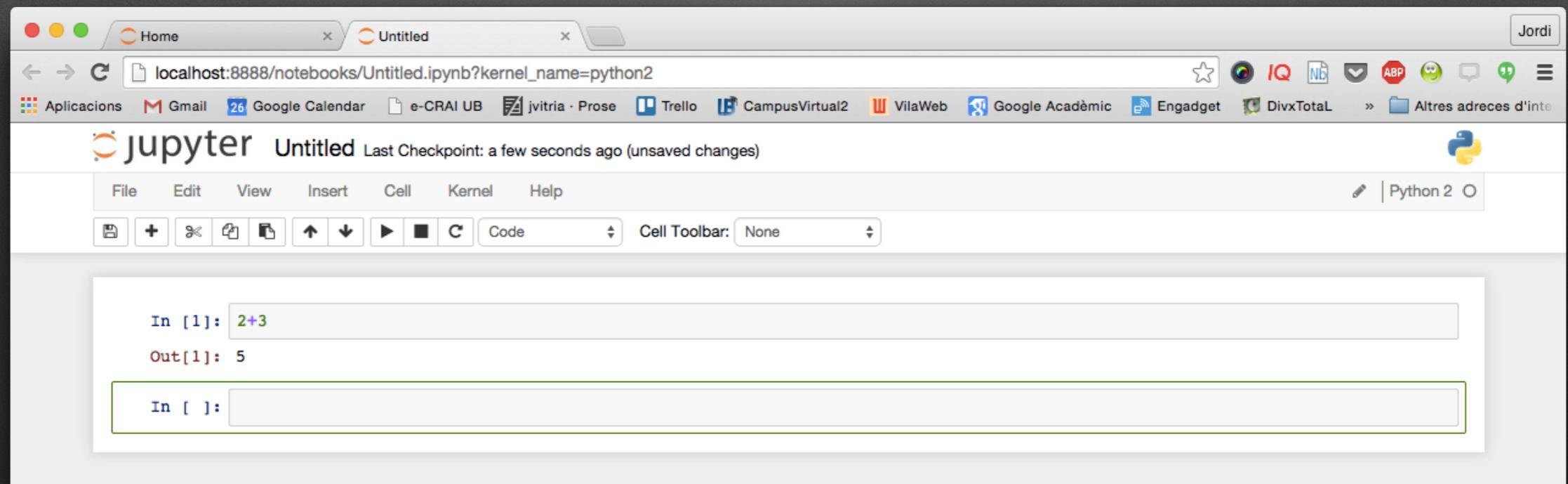
- The name of the operator
- The input arguments
- The output value
- Side effects

A typical operation takes one or more input arguments and uses the information in these to produce an output value. Along the way, the computer might take some action: display a graph, store a file, make a sound, etc. These actions are called side effects.

Python is a general-purpose programming language, so when we want to use more specific commands (such as statistical operators or string processing operators) we usually need to import them before we can use them. For Scientific Python, one of the most important libraries that we need is numpy (Numerical Python), which can be loaded like this:

Notebooks

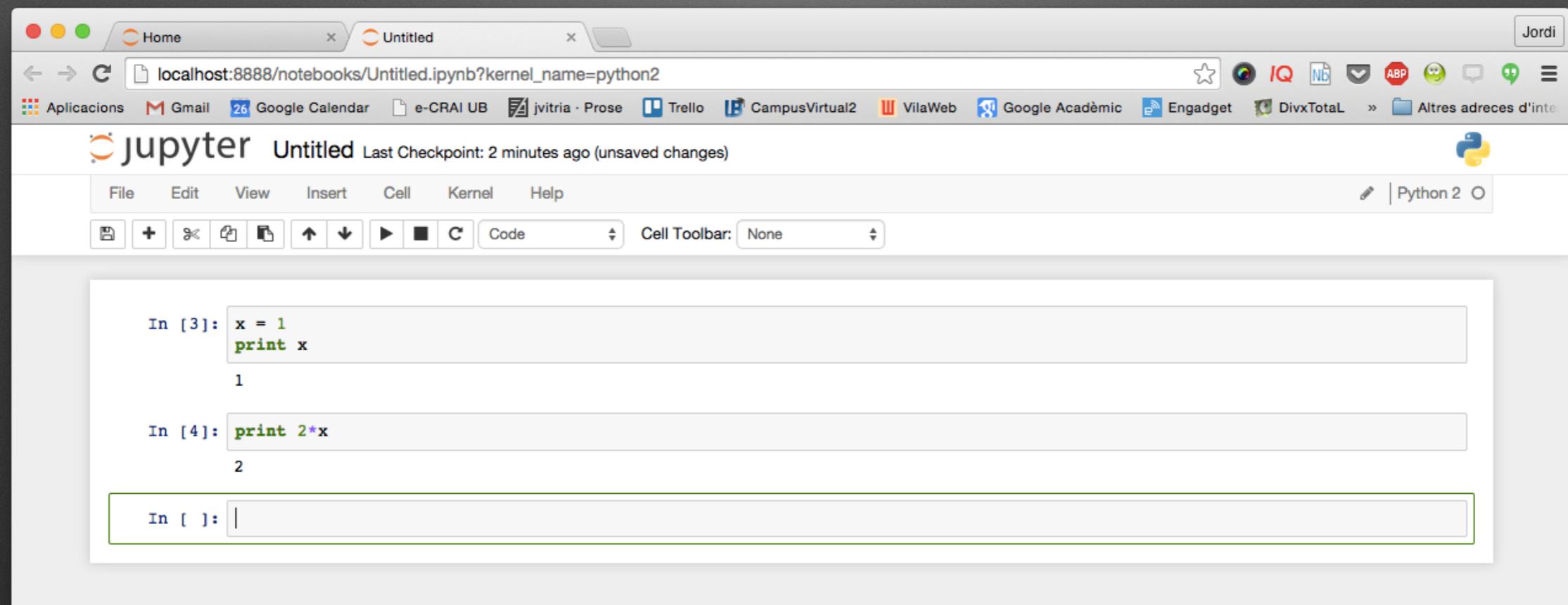
A notebook is made up of a number of cells. Each cell can contain Python code.



You can execute a cell by clicking on it and pressing Shift-Enter. When you do so, the code in the cell will run, and the output of the cell will be displayed beneath the cell.

Notebooks

Global variables are shared between cells. Executing the second cell thus gives the following result:



The screenshot shows a Jupyter Notebook running in a web browser window titled "Untitled". The browser's address bar indicates the URL is "localhost:8888/notebooks/Untitled.ipynb?kernel_name=python2". The notebook interface includes a toolbar with various icons and a menu bar with options like File, Edit, View, Insert, Cell, Kernel, and Help. The main area displays two code cells. The first cell, labeled "In [3]:", contains the Python code "x = 1" followed by "print x", which outputs the value "1". The second cell, labeled "In [4]:", contains the code "print 2*x", which outputs the value "2". A third cell, labeled "In []:", is visible at the bottom, indicating where the next code input will be placed. The entire interface is set against a dark background.

```
In [3]: x = 1
        print x
1

In [4]: print 2*x
2
```

By convention, notebooks are expected to be run from top to bottom. Failing to execute some cells or executing cells out of order can result in errors.

Notebooks

After you have modified a notebook for one of the assignments by modifying or executing some of its cells, remember to save your changes!

