

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
'Die Lust': First Part.....	500	14.6	15.2	13.6	13.0	12.0
'Die Lust': Second Part.....	500	16.3	17.2	15.5	12.3	16.5
'Die Leidenschaft des Knecke Fuchs', .....	500	18.5	16.5	16.9	14.8	16.6
'Die Leiden d. j. Werthers', ...	500	20.7	20.9	19.1	22.7	18.2
'Die Sylvaenische Reise: Rom', ...	500	23.1	23.7	22.7	21.5	22.7
Total	5000	21.9	22.6	22.7	25.1	24.5

# Introduction to Data Analysis with Python

Jordi Vitrià, PhD  
Universitat de Barcelona

# Syllabus

Data Science Steps & Python Data Science Stack

Data Science Plumbing & Agile Data Science

Why Python?

What is IPython?

**Support The Guardian**  
Available for everyone, funded by readers

[Contribute →](#)

[Subscribe →](#)

[Sign in](#)

The  
**Guardian**

News | Opinion | Sport | Culture | Lifestyle



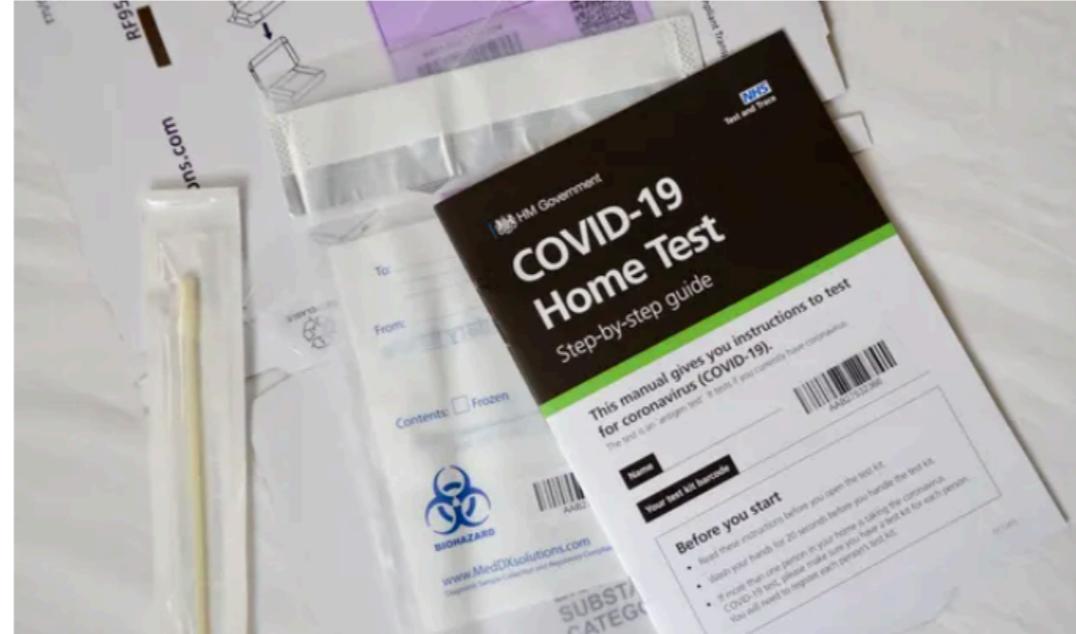
UK ► UK politics Education Media Society Law Scotland Wales Northern Ireland

**Health policy**

## Covid: how Excel may have caused loss of 16,000 test results in England

**Public Health England data error blamed on limitations of Microsoft spreadsheet**

- [Coronavirus - latest updates](#)
- [See all our coronavirus coverage](#)



▲ More than 50,000 potentially infectious people may have been missed by contact tracers after 15,841 positive tests were left off the daily figures. Photograph: Simon Leigh/Alamy

**Health policy**

## Covid: how Excel may have caused loss of 16,000 test results in England

...

But while CSV files can be any size, Microsoft Excel files can only be 1,048,576 rows long – or, in older versions which PHE may have still been using, a mere 65,536. When a CSV file longer than that is opened, the bottom rows get cut off and are no longer displayed. That means that, once the lab had performed more than a million tests, it was only a matter of time before its reports failed to be read by PHE.

...

# 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.
- Data Pipelines.
- APIs

### 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.

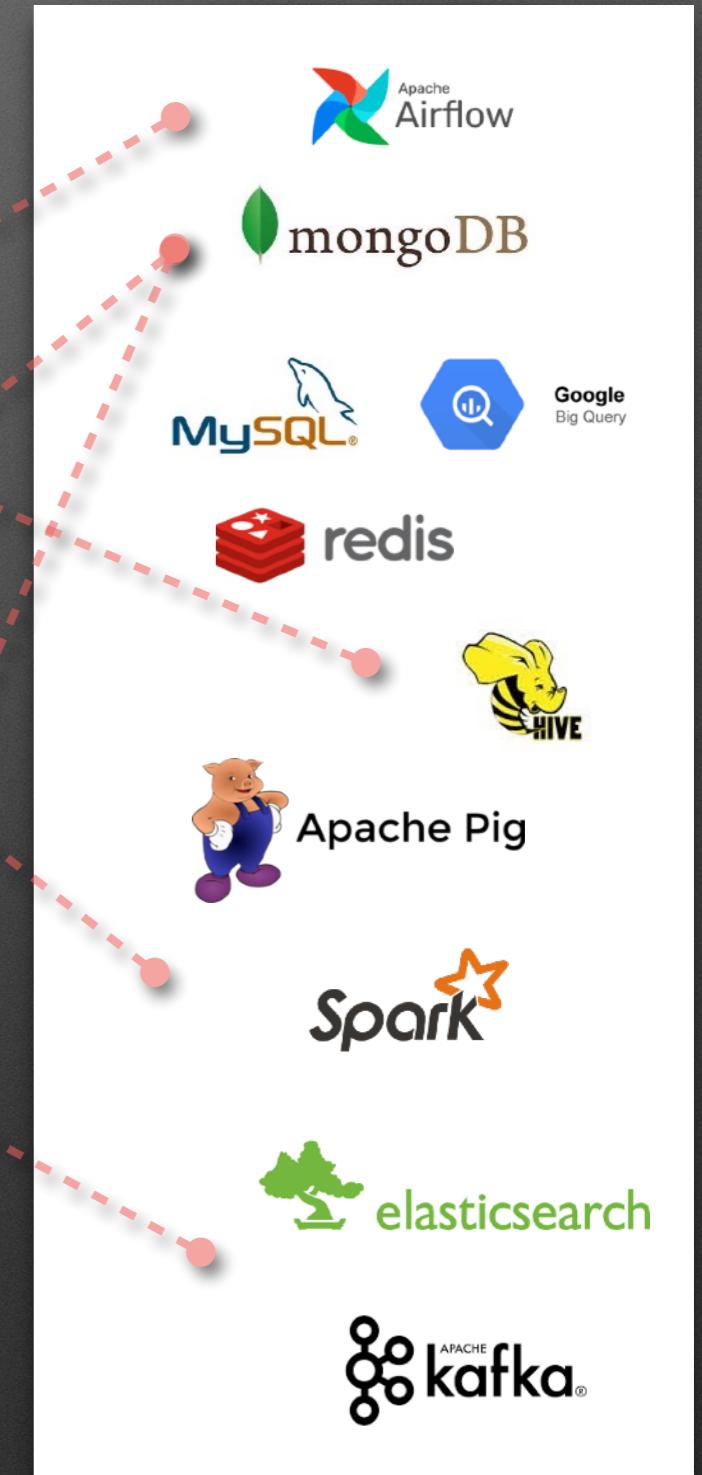
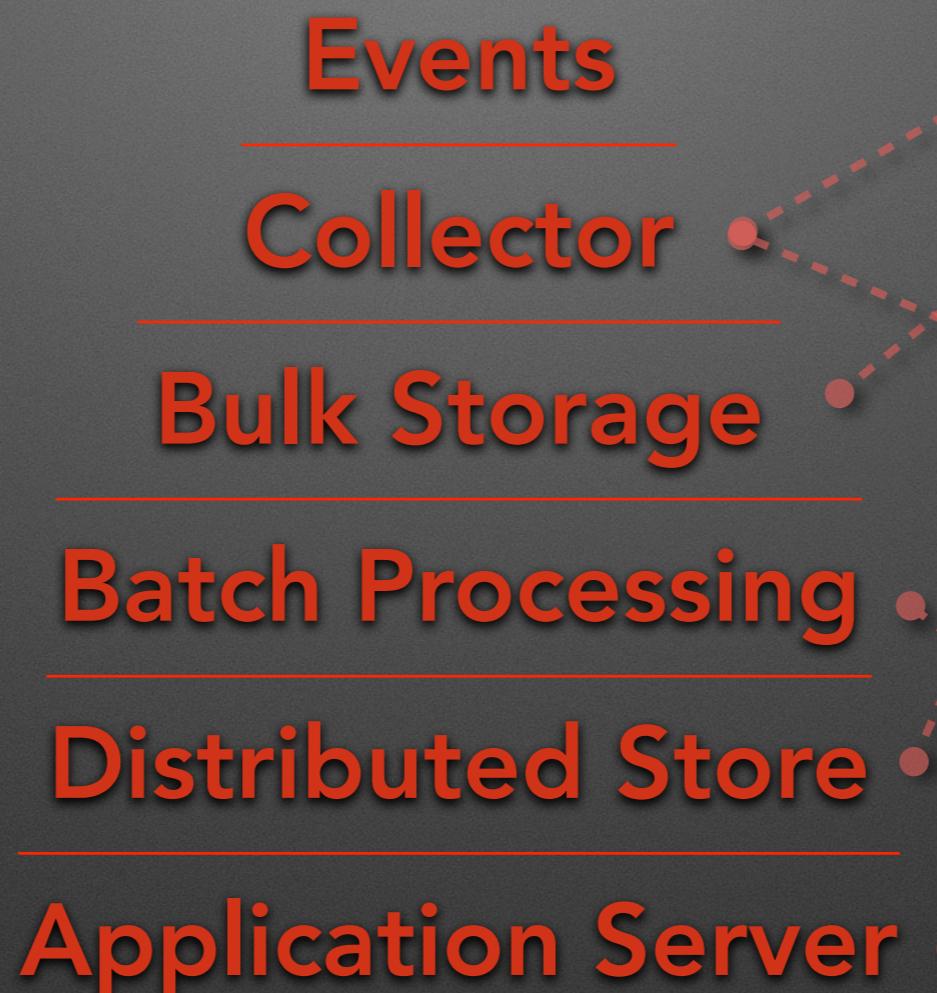
Events are collected from different sources and aggregated to bulk storage.

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.

Application server.



# 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.
  - **Tensorflow, PyTorch**.

# Why Python?

- Strong points:
  - Viable alternative to SAS/Matlab/R/...
  - Uptake in the several sectors.
  - Upcoming generation of programmers with Python as a first language.
  - Strong Python communities in HPC/Scientific Computing/Data Science.
- Weak points
  - Weak support.

# Why Python?

## Python in Big Data Workflows

Cloud

Counting/ETL-ELT

HDF5  
Spark  
Hadoop  
SQL

...

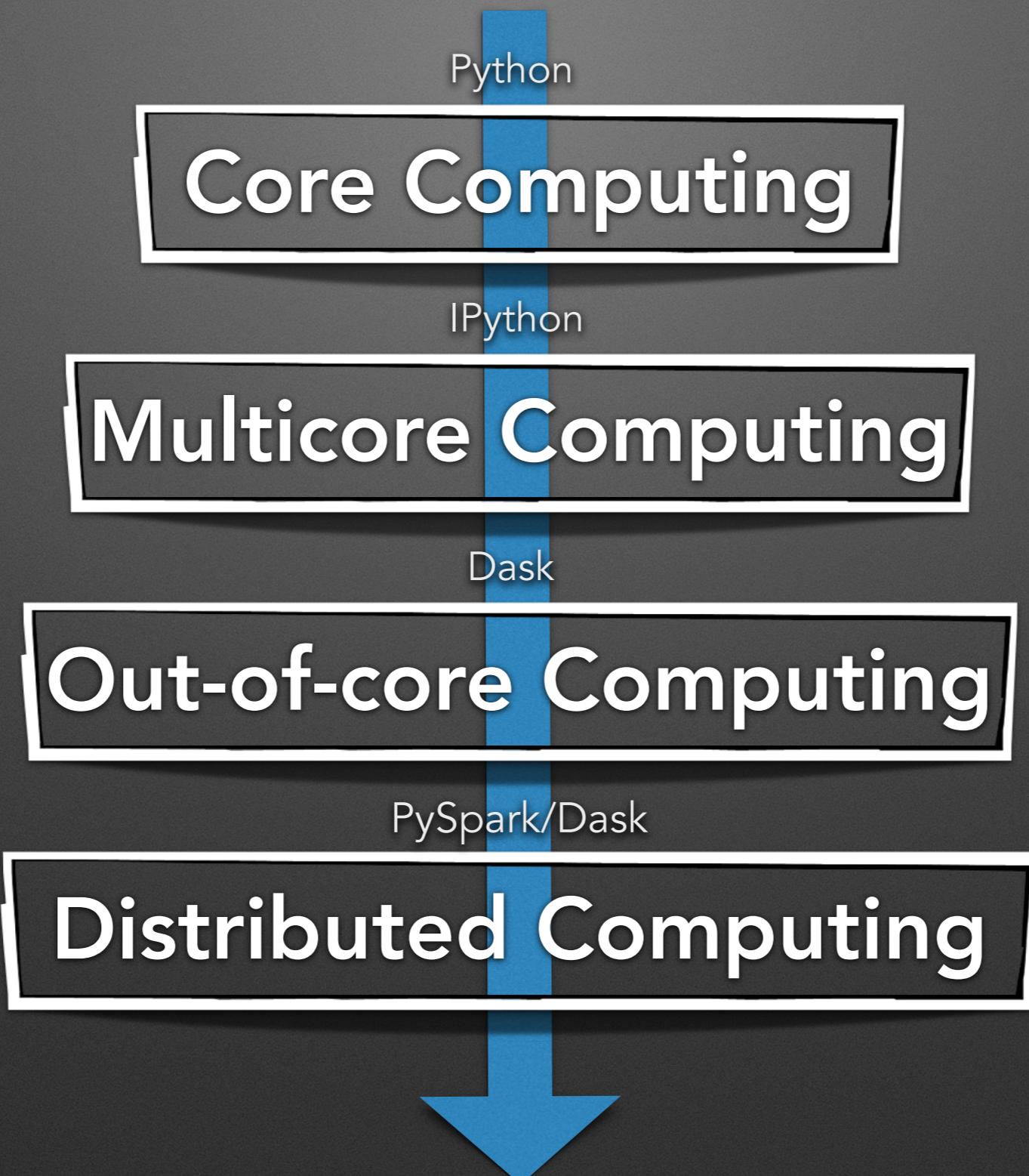
My Machine

Insight/Reporting

pandas  
scikit-learn  
Blaze  
PySpark  
PyMongo

...

# Why Python?



# More on Python

[www.codecademy.com](http://www.codecademy.com)

The screenshot shows the Codecademy homepage with the navigation bar: codecademy Catalog Resources ▾ Community ▾ Pro Pricing ▾ For Business. Below the navigation is a search bar and a 'Log In' button. The main content area features a section titled 'Explore our catalog' with four cards: 'Explore All Javascript' (pink), 'Explore All Python' (blue), 'Explore All Web Development' (yellow), and 'Explore All Data Science' (green). Each card has a small icon and a 'Browse Courses →' link.

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

The screenshot shows the Python Practice Book page. At the top is a sidebar with a blue header containing the book's name and a search bar. The sidebar lists chapters: 1. Getting Started, 2. Working with Data, 3. Modules, 4. Object Oriented Programming, 5. Iterators & Generators, and 6. Functional Programming. The main content area has a header 'Python Practice Book' and a sub-header 'About this Book'. It includes a welcome message, a note about the author, and information about upcoming trainings.

[www.python.org](http://www.python.org)

The screenshot shows the Python.org homepage. The top navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. The main header features the Python logo and navigation tabs: About, Downloads, Documentation, Community, Success Stories, News, and Events. A central content area displays a Python code snippet for generating a Fibonacci series up to n, followed by the resulting sequence of numbers. To the right is a sidebar with a heading 'Functions Defined' and text explaining Python's function definition capabilities. At the bottom, a large call-to-action banner states: 'Python is a programming language that lets you work quickly and integrate systems more effectively.' with a 'Learn More' link.

# What is IPython/Jupyter?

I is for interactive

In this class, we will use **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 and data science.

# Jupyter Notebooks

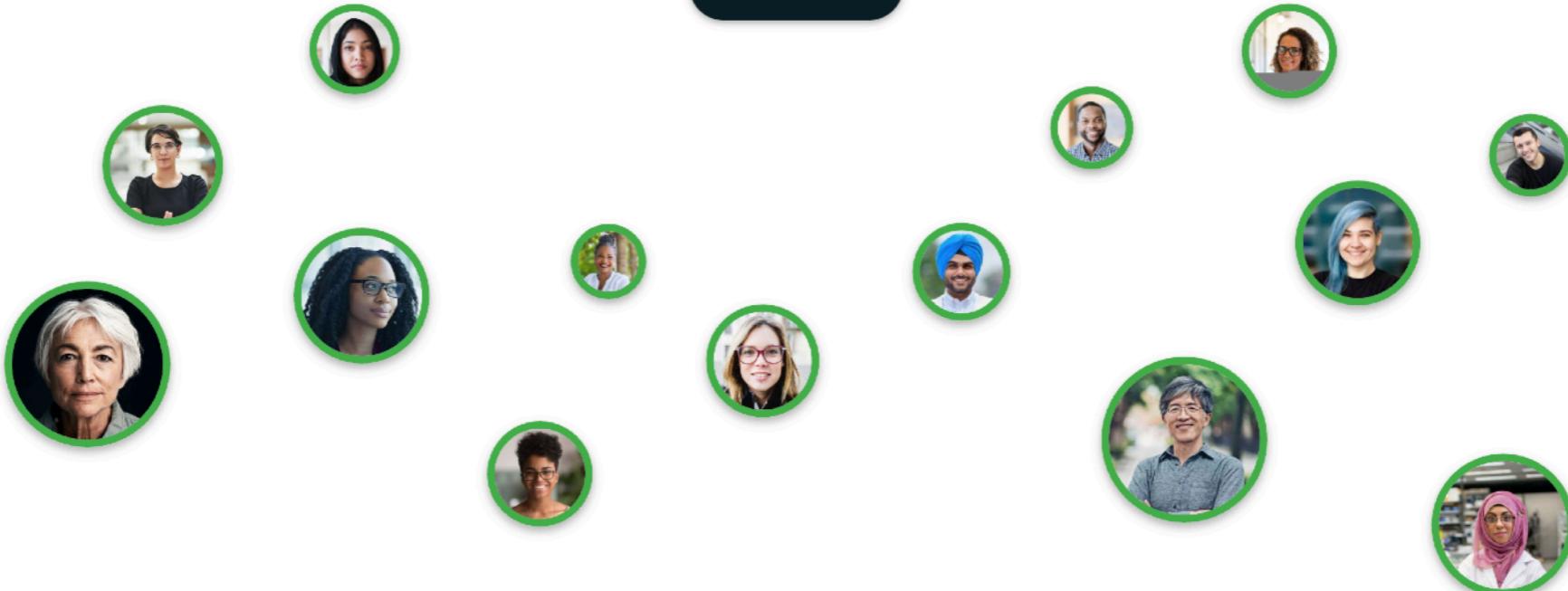
For new users who want to get up and running with minimal effort, we suggest you to download and install Anaconda (<https://www.anaconda.com/products/individual>) which provide a setup based on Python 3.8

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.

[Products ▾](#)[Pricing](#)[Solutions ▾](#)[Resources ▾](#)[Blog](#)[Company ▾](#)[Get Started](#)

# Data science technology for human sensemaking.

A movement that brings together millions of data science practitioners,  
data-driven enterprises, and the open source community.

[Get Started](#)

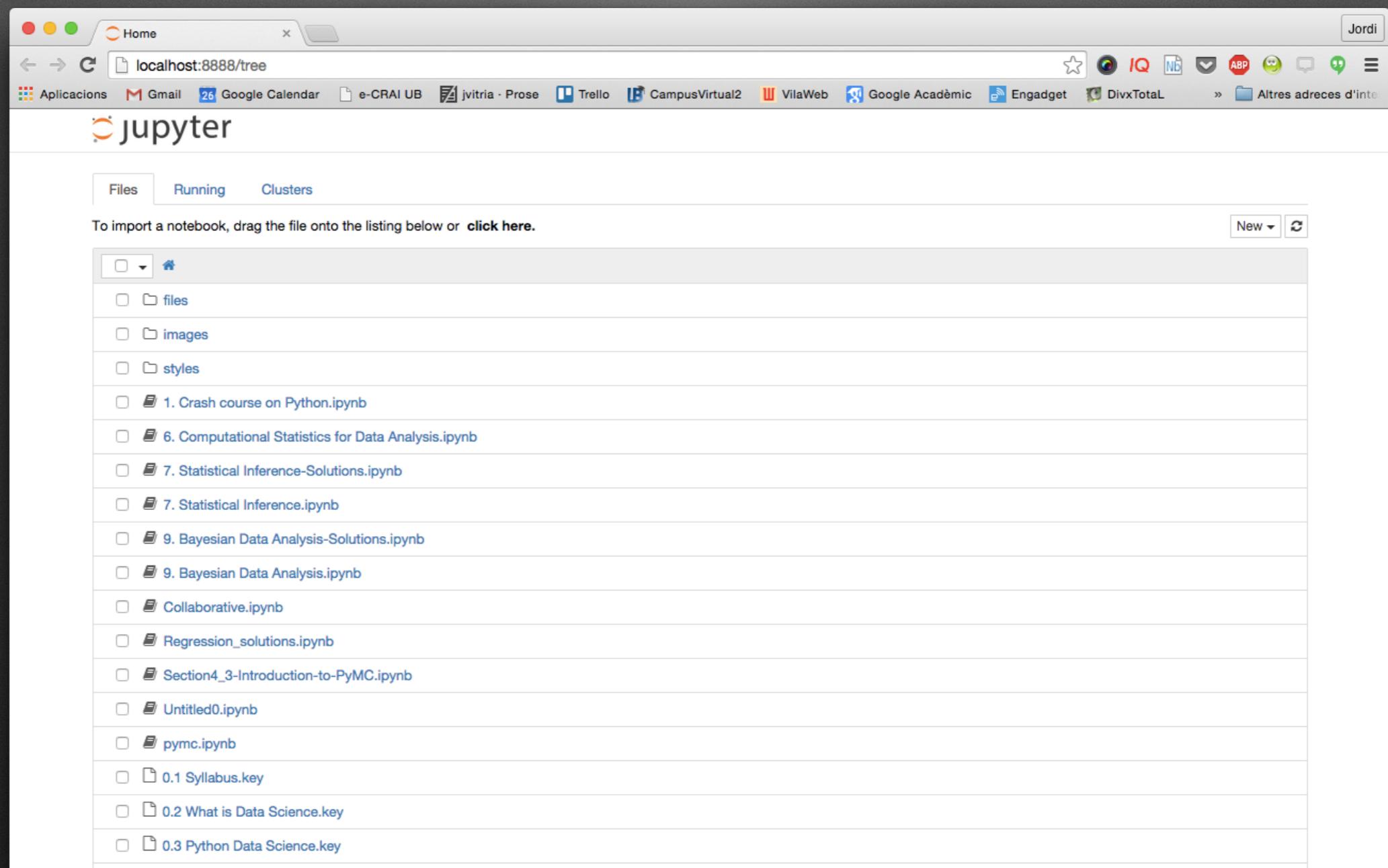
# Jupyter Notebooks

To run a notebook,

- (1) Using the command line, navigate to the directory you want to store notebooks, and
- (2) Start running a notebook server from the command line using the following command:

jupyter notebook

# Jupyter Notebooks



# Jupyter 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 address bar shows the URL "localhost:8888/notebooks/1.%20Crash%20course%20on%20Python.ipynb". The main content area displays a section titled "Python". It includes text about IPython, an interpreter, and operators, followed by two code cells showing the execution of simple Python code. The first cell's input is "3 + 4 + 9" and its output is "16". The second cell's input is "range(10)" and its output is "[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]". Below the code cells, there is a bulleted list of four parts of an operator computation. At the bottom, there is a note about side effects and a reference to the numpy library.

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:

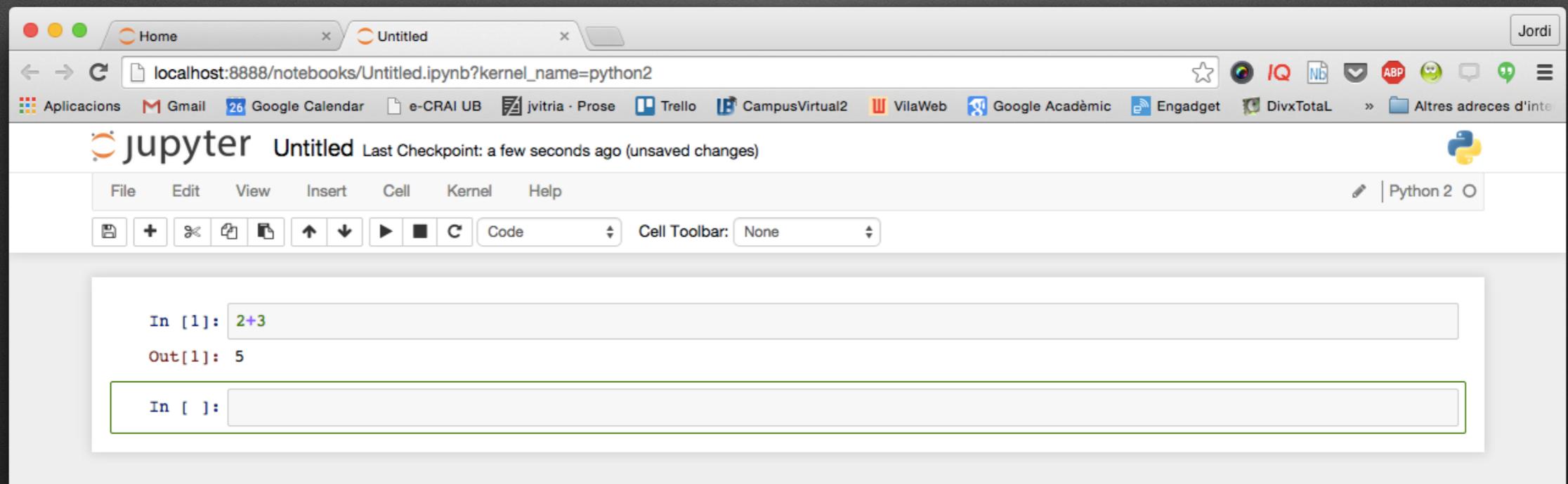
- The name of the operator
- The input arguments
- The output value
- Side effects

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:

# Jupyter Notebooks

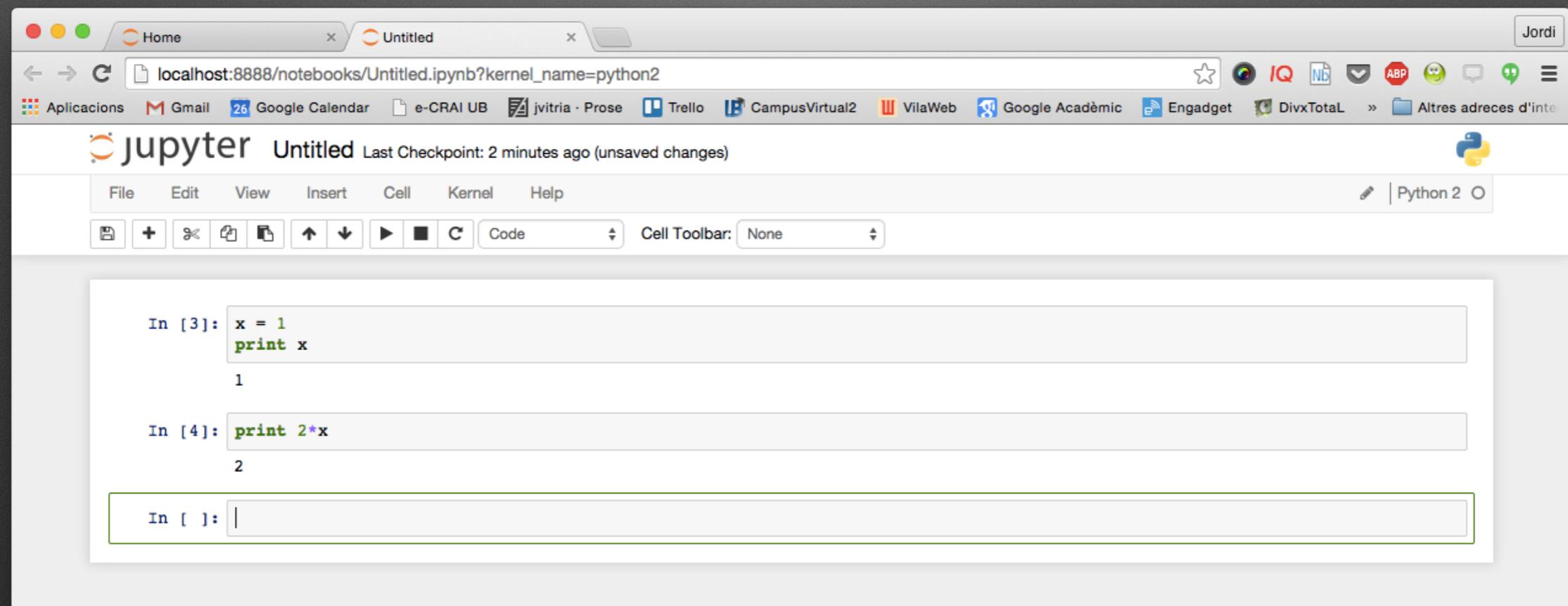
An IPython notebook is made up of a number of cells. Each cell can contain Python code.



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.

# Jupyter Notebooks

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



The screenshot shows a Jupyter Notebook window titled "Untitled". The browser tab is "localhost:8888/notebooks/Untitled.ipynb?kernel\_name=python2". The notebook interface includes a toolbar with File, Edit, View, Insert, Cell, Kernel, Help, and a Python 2 kernel selector. Below the toolbar are cell controls for selecting, running, and deleting cells. The main area contains two cells:

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

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

The first cell has been run, showing the output "1". The second cell has also been run, showing the output "2". A third cell, "In [ ]:", is visible at the bottom, indicating it is ready for input.

By convention, jupyter 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.

# Jupyter Notebooks

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

