

# Python for Scientists

2nd Edition | April & May, 2022

Department of Civil, Chemical and Environmental Engineering (DICCA)  
University of Genoa, Italia

**Andrea Lira Loarca**  
[andrea.lira.loarca@unige.it](mailto:andrea.lira.loarca@unige.it)

# Overview of the course

- Introduction to python, editors, scientific libraries
- Basics of python programming
- Numpy
- Matplotlib
- Scipy
- Pandas
- Overview of advanced libraries - Xarray, Cartopy, Seaborn, Scikit-Learn
- Exercises proposed by students

Team code  
fiOpjq9

# Why Python?

## Scientific's needs

- Get data (simulations, experiments, databases)
- Manipulate and process data
- Visualize results, quickly to understand, but also with high quality figures, for reports or publications



“

I think every kid in the world should learn to code. Whether your passion is in science or the arts, it's a way to express yourself.

Tim Cook CEO OF APPLE

DATA SCIENTISTS

## Why 'Data Scientist' Will Continue To Be 'the Sexiest Job Of the 21st Century'

As technology evolves, the exact skills and nature of jobs in data science will probably vary



By 2025, 463 exabytes of data will be created each day, according to some estimates. (For perspective, one exabyte of storage could hold 50,000 years of DVD-quality video.) It's now easier than ever to translate physical and digital actions into data, and businesses of all types have raced to amass as much data as possible in order to gain a competitive edge.

Published: 04 February 2015

### Programming: Pick up Python

Jeffrey M. Perkel

Nature 518, 125–126 (2015) | Cite this article

278 Accesses | 51 Citations | 689 Altmetric | Metrics

A powerful programming language with huge community support.

Data alone doesn't spur innovation — rather, it's data-driven storytelling that helps uncover hidden trends, powers personalization, and streamlines processes.



Credit: Illustration by the Project Twins

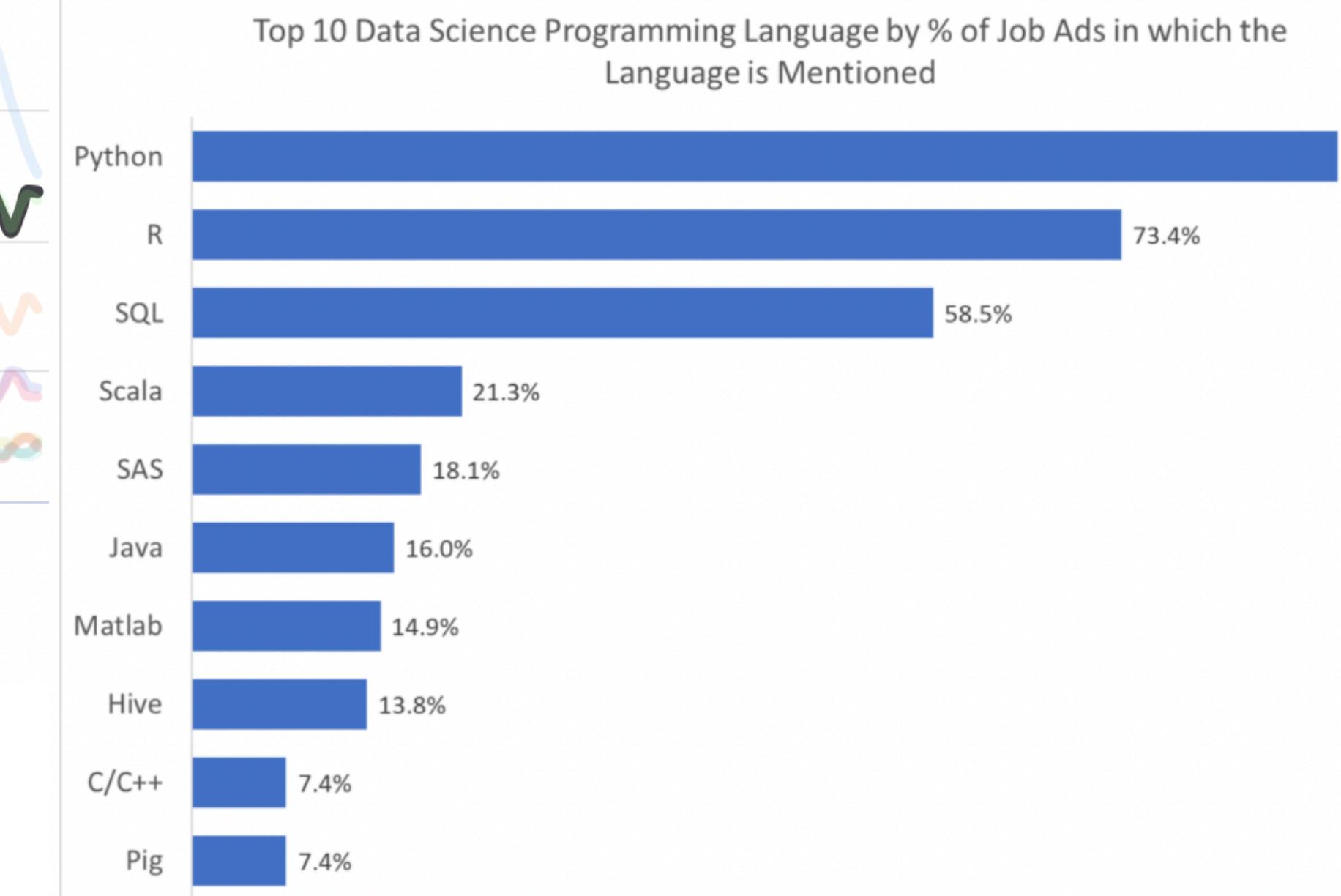
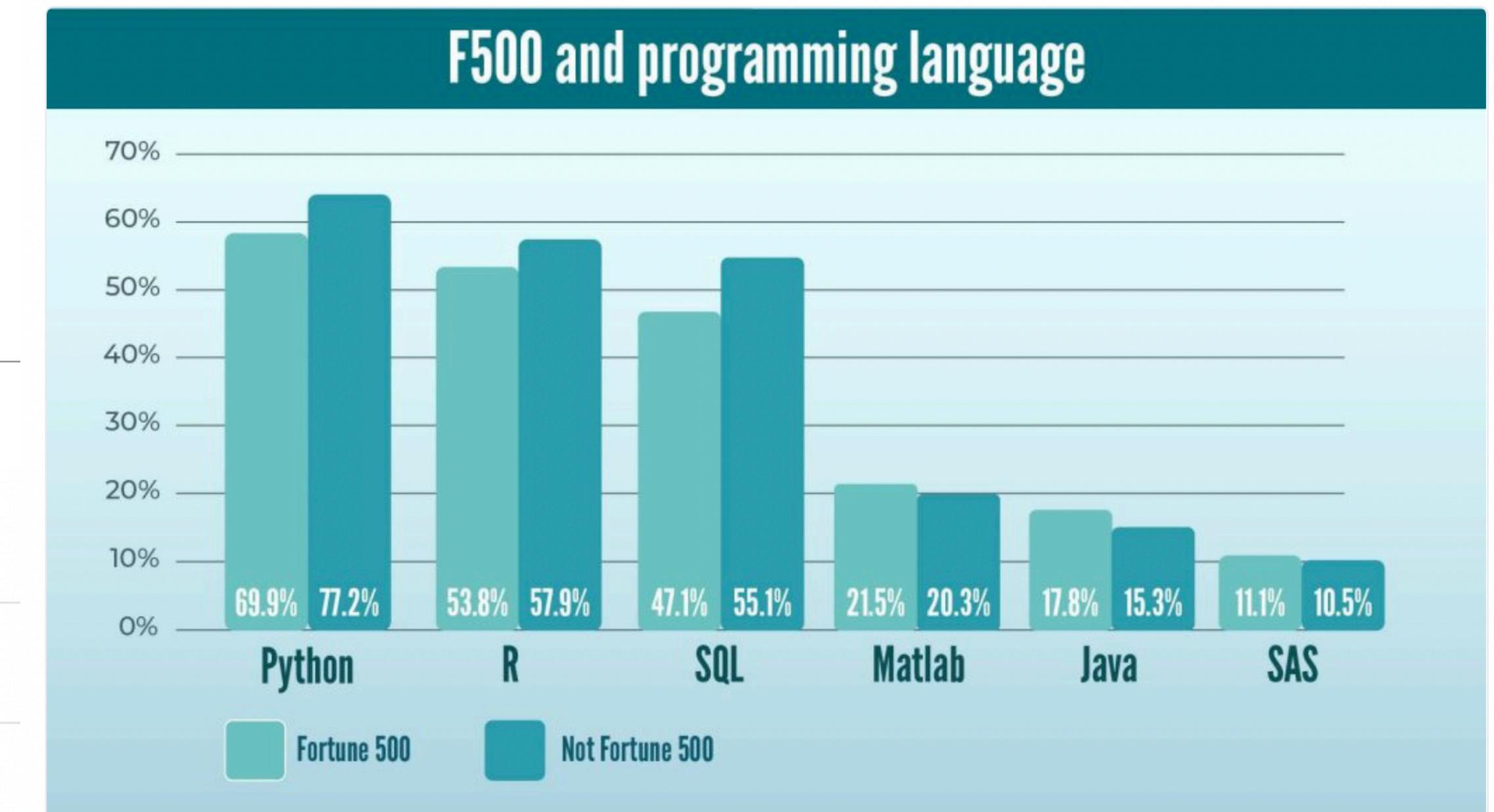
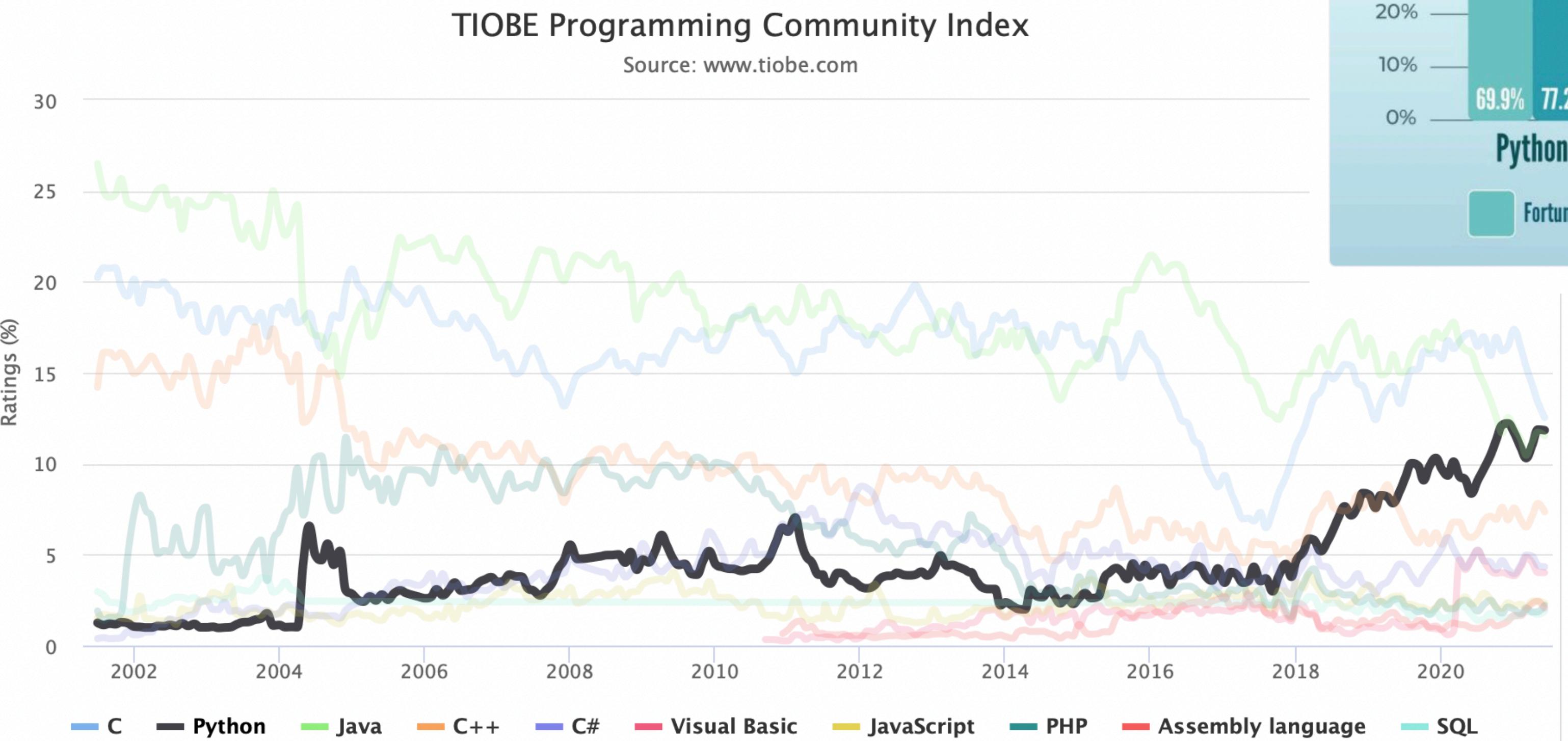
# Why Python?

---

## Python's strengths

- Extensive library: classic numerical methods, plotting or data processing tools. We don't want to re-program the plotting of a curve, a Fourier transform or a fitting algorithm. Don't reinvent the wheel!-
- Easy to learn: most scientists are not payed as programmers, neither have they been trained so. They need to be able to draw a curve, smooth a signal, do a Fourier transform in a few minutes.
- Easy communication: to keep code alive within a lab or a company it should be as readable as a book by collaborators, students, or maybe customers.
- Efficient code: Python numerical modules are computationally efficient.\*\*\*
- Universal Python is a language used for many different problems: Learning Python avoids learning a new software for each new problem.
- Strong community: google new libraries when approaching a problem.

# Python vs. Other Languages



# Python vs. Other Languages

## Python

### Pros:

- Very rich scientific computing libraries
- Well thought out language, allowing to write very readable and well structured code: we “code what we think”.
- Many libraries beyond scientific computing (web server, serial port access, etc.)
- Free and open-source software, widely spread, with a vibrant community.
- A variety of powerful environments to work in, such as IPython, Spyder, Jupyter notebooks, Pycharm, Visual Studio Code

### Cons:

- Not all the algorithms that can be found in more specialized software or toolboxes.

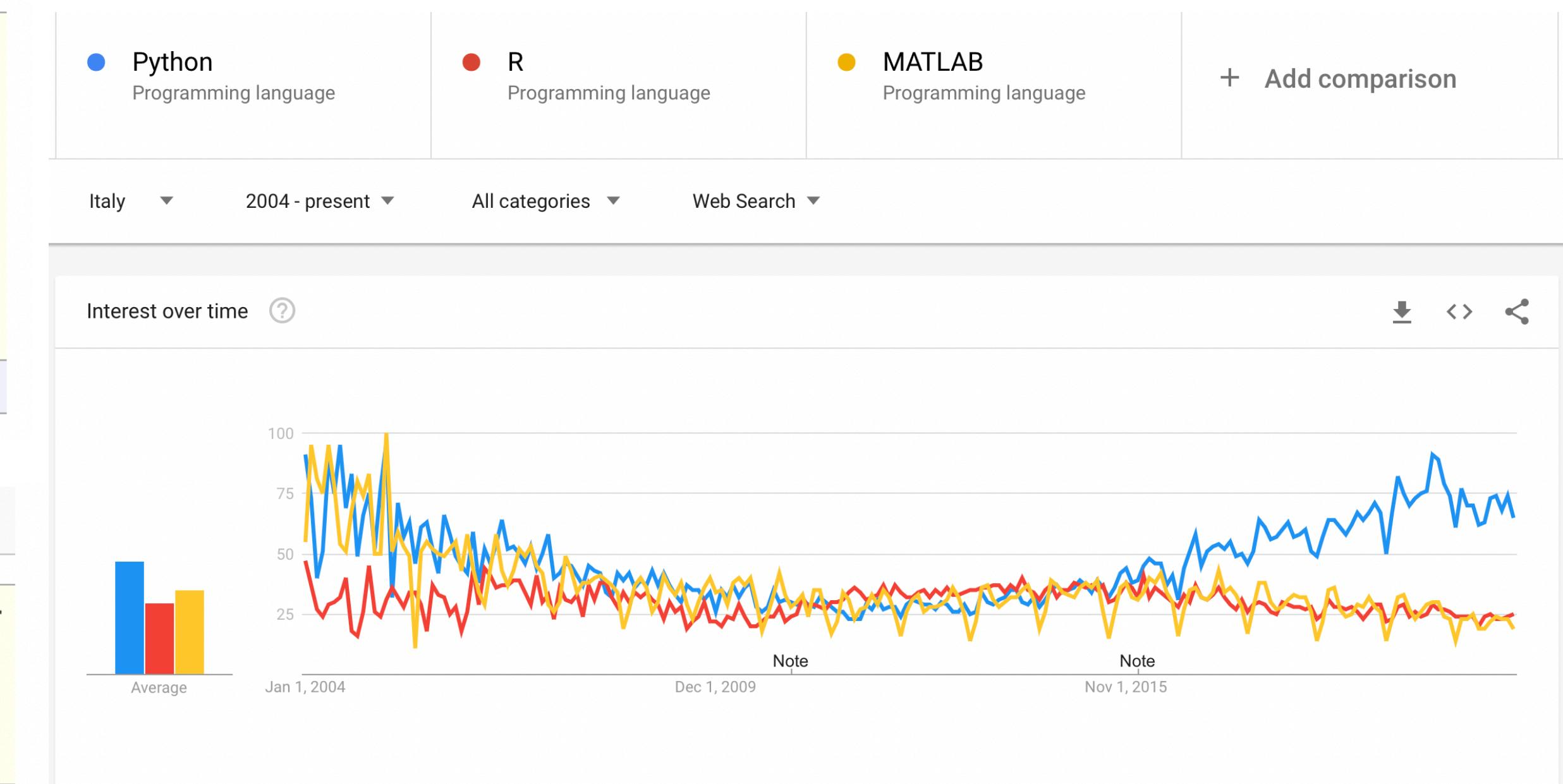
## Matlab scripting language

### Pros:

- Very rich collection of libraries with numerous algorithms, for many different domains. Fast execution because these libraries are often written in a compiled language.
- Pleasant development environment: comprehensive and help, integrated editor, etc.
- Commercial support is available.

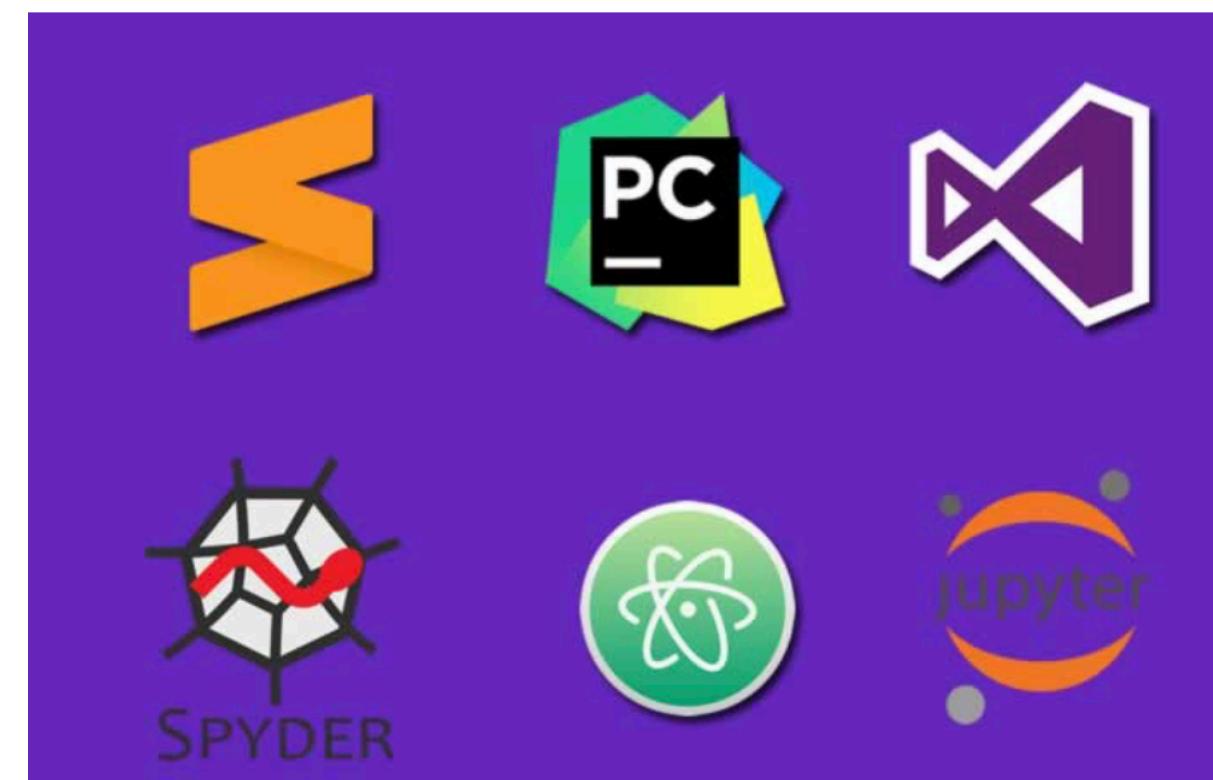
### Cons:

- Base language is quite poor and can become restrictive for advanced users.
- Not free.



# How to run Python?

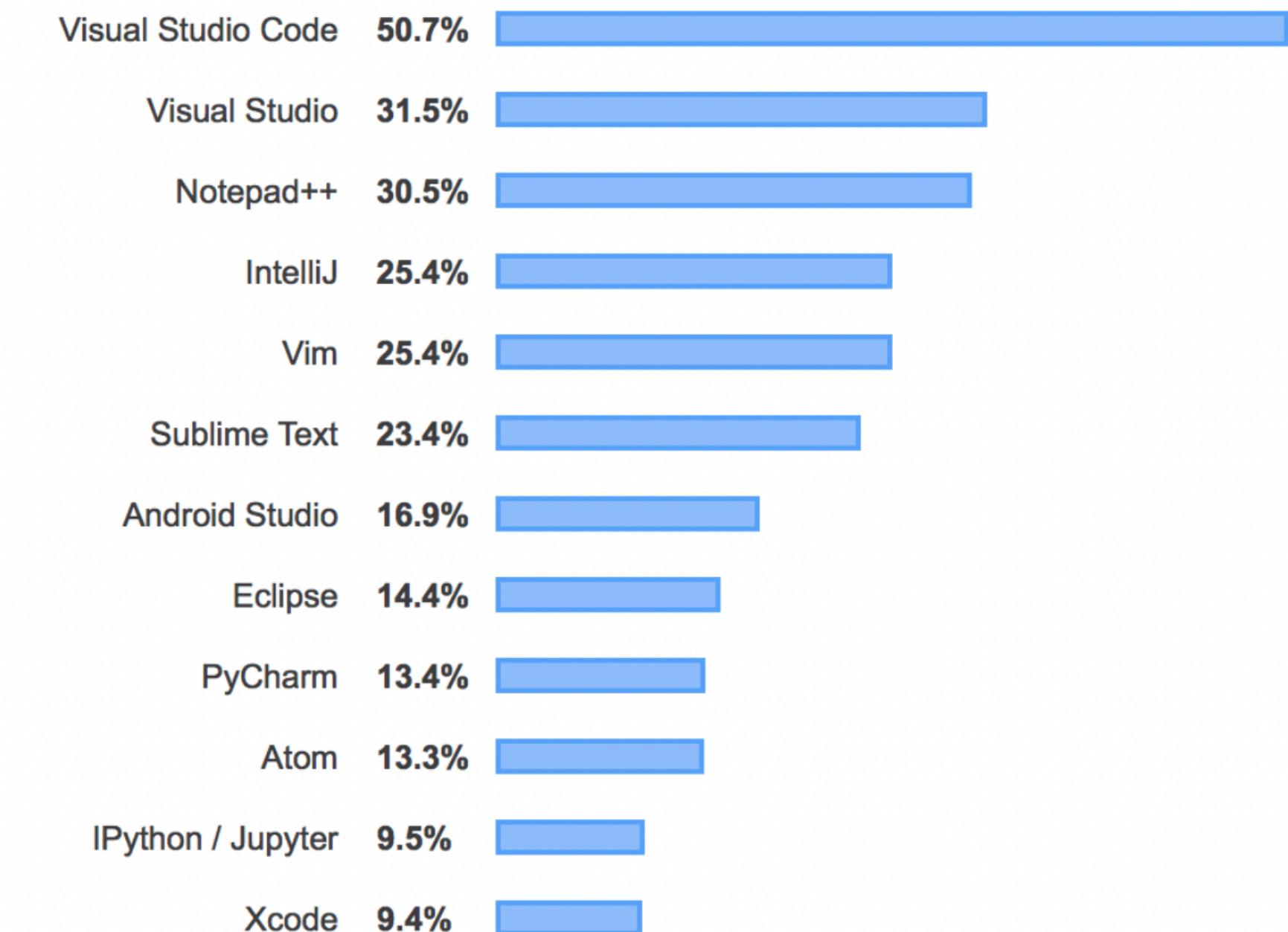
1. Distribution - set of libraries, packages



2. Editor or application

- Notepad++
- Vim
- Sublime Text
- VS Code
- Spyder
- Atom
- PyCharm
- Jupyter

...



# Scientific-based Python distribution - Anaconda

[https://www.anaconda.com/  
products/individual](https://www.anaconda.com/products/individual)

The screenshot displays the Anaconda Distribution website's homepage. At the top, the logo "ANACONDA DISTRIBUTION" and the tagline "Most Trusted Distribution for Data Science" are visible. Below this, there are three main sections: "ANACONDA NAVIGATOR" (Desktop Portal to Data Science), "ANACONDA PROJECT" (Portable Data Science Encapsulation), and "DATA SCIENCE LIBRARIES". The "DATA SCIENCE LIBRARIES" section is highlighted in blue and lists various tools categorized by type: Data Science IDEs (jupyter, spyder, jupyterlab, R Studio), Analytics & Scientific Computing (NumPy, SciPy, Numba, pandas, DASK), Visualization (Bokeh, HoloViews, DataShader, matplotlib), and Machine Learning (TensorFlow, scikit-learn, H2O.ai, theano). A note "...and many more!" is present at the bottom right of this section. The bottom part of the page features the "CONDA" logo and the tagline "Data Science Package & Environment Manager".

# Scientific-based Python distribution - Anaconda

## The Fundamentals



Jupyter is an open-source project created to support interactive data science and scientific computing across programming languages. Jupyter offers a web-based environment for working with notebooks containing code, data, and text. Jupyter notebooks are the standard workspace for most Python data scientists.



The SciPy library consists of a specific set of fundamental scientific and numerical tools for Python that data scientists use to build their own tools and programs. It provides many user-friendly and efficient numerical routines, such as routines for numerical integration, interpolation, optimization, linear algebra, and statistics.



A library for tabular data structures, data analysis, and data modeling tools, including built-in plotting using Matplotlib. pandas aims to be the fundamental high-level building block for doing practical, real world data analysis in Python



A core package for scientific computing with Python. Numpy enables array formation and basic operations with arrays. Numpy is used for indexing and sorting but can also be used for linear algebra and other operations. Many other data-science libraries for Python are built on NumPy internally, including Pandas and SciPy.



Matplotlib is the most well-established Python data visualization tool, focusing primarily on two-dimensional plots (line charts, bar charts, scatter plots, histograms, and many others). It works with many GUI interfaces and file formats, but has relatively limited interactive support in web browsers.



Plotly's Python graphing library makes interactive, publication-quality graphs. It is a popular and powerful browser-based visualization library that lets you create interactive, JavaScript-based plots from Python.

## Data Visualization



Bokeh is an interactive visualization library for modern web browsers. It provides elegant, concise construction of versatile graphics, and affords high-performance interactivity over large or streaming datasets. Bokeh can help anyone who would like to quickly and easily make interactive plots, dashboards, and data applications.



HoloViz is an Anaconda project to simplify and improve Python-based visualization by adding high-performance server-side rendering (Datashader), simple plug-in replacement for static visualizations with interactive Bokeh-based plots (hvPlot), and declarative high-level interfaces for building large and complex systems (HoloViews and Param).

# Scientific-based Python distribution - Anaconda

## Scalable Computing



Numba is a high-performance Python compiler. It makes Python faster and optimizes the performance of Numpy arrays, reaching the speed of FORTRAN and C without a an additional compilation step.



Dask is a Python package used to scale NumPy workflows with parallel processing to enable multi-dimensional data analysis, enabling users to store and process data larger than their computer's RAM. Dask can scale out to clusters, or scale down to a single computer. Dask mimics the pandas and NumPy API, making it more intuitive for Python data scientists.

## RAPIDS

The RAPIDS data science framework is a collection of libraries for running end-to-end data science pipelines completely on the GPU. The interaction is designed to have a familiar look and feel to working in Python, but utilizes optimized NVIDIA® CUDA® primitives and high-bandwidth GPU memory under the hood.



A fault-tolerant cluster computing framework and interface for programming clusters launched by UC Berkeley. Developed for Java/Hadoop ecosystem but with support for Python. PySpark is the Python API for Spark.

## Powered by Dask

These software projects are well-integrated with Dask, or use Dask to power components of their infrastructure.



pandas  
Tabular data analysis



NumPy  
Array and numerical computing



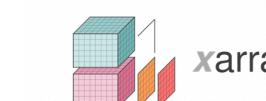
scikit-learn  
Machine learning in Python



scikit-image  
A collection of algorithms for image processing in Python



XGBoost  
Gradient boosted trees for machine learning  
XGBoost can use Dask to bootstrap itself for distributed training



xarray  
Brings the labeled data power of pandas to the physical sciences, by providing N-dimensional variants of the core pandas data structures



RAPIDS  
GPU Accelerated libraries for data science



Iris  
A Python library for analysing and visualising Earth science data



Pangeo  
A community effort for big data geoscience in the cloud



napari  
Multi-dimensional image viewer for Python



Datashader  
Visualization packages for large data



TPOT  
A Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming



Prefect  
A workflow management system, designed for modern infrastructure



Snorkel  
Programmatically build training data for machine learning

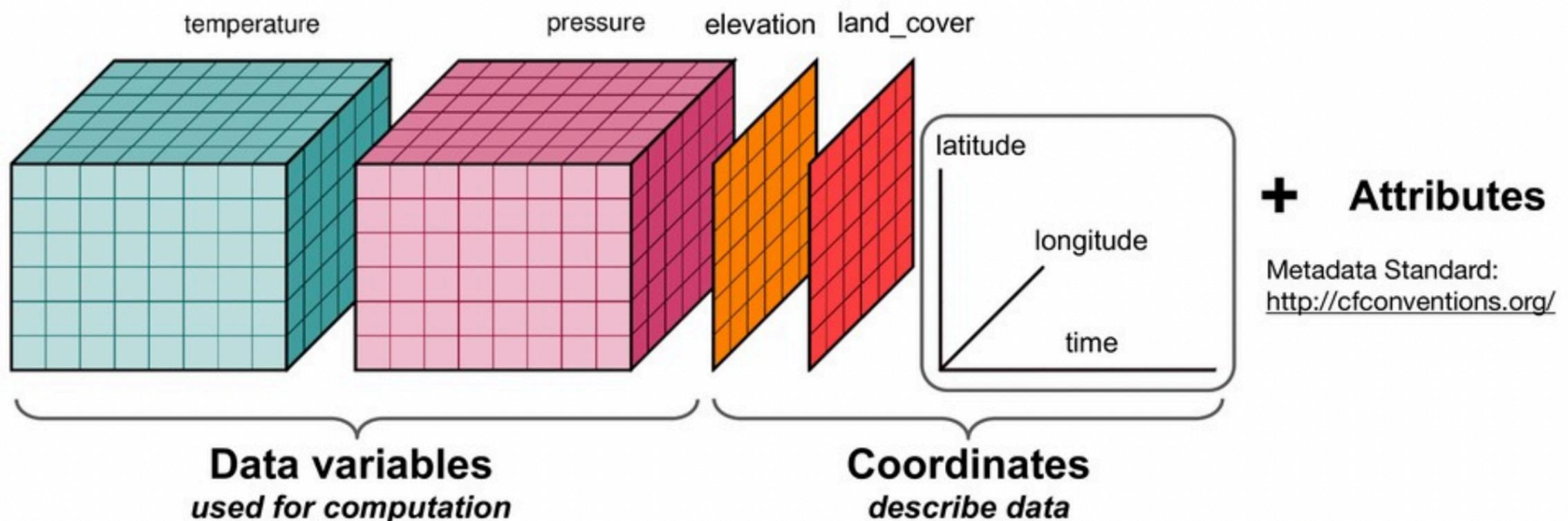
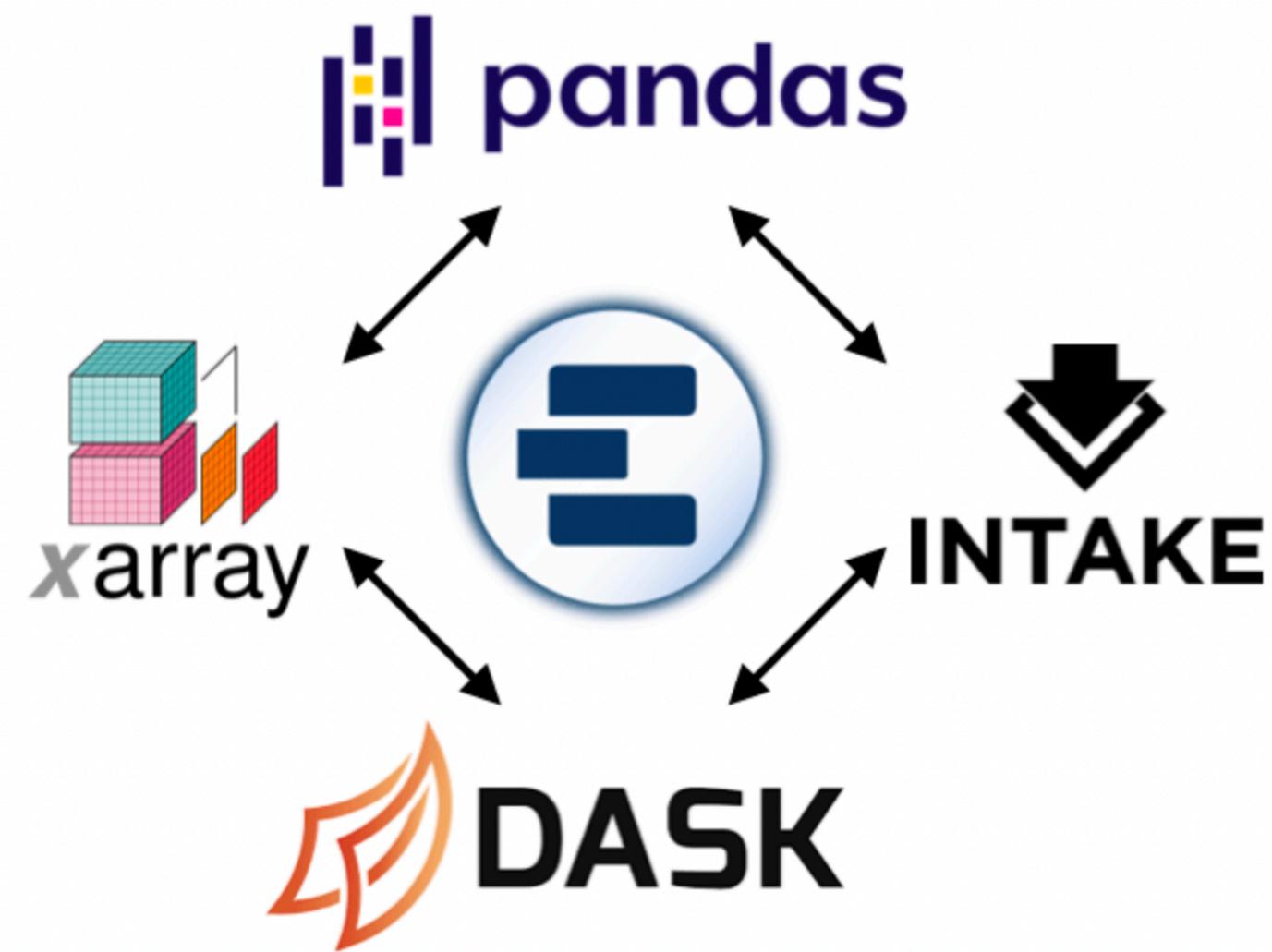


INTAKE  
A lightweight package for finding, investigating, loading and disseminating data



MDAnalysis  
A Python toolkit to analyze molecular dynamics trajectories generated by a wide range of popular simulation packages

# Scientific-based Python distribution - Anaconda



## Machine Learning

**K Keras**

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

**TensorFlow**

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

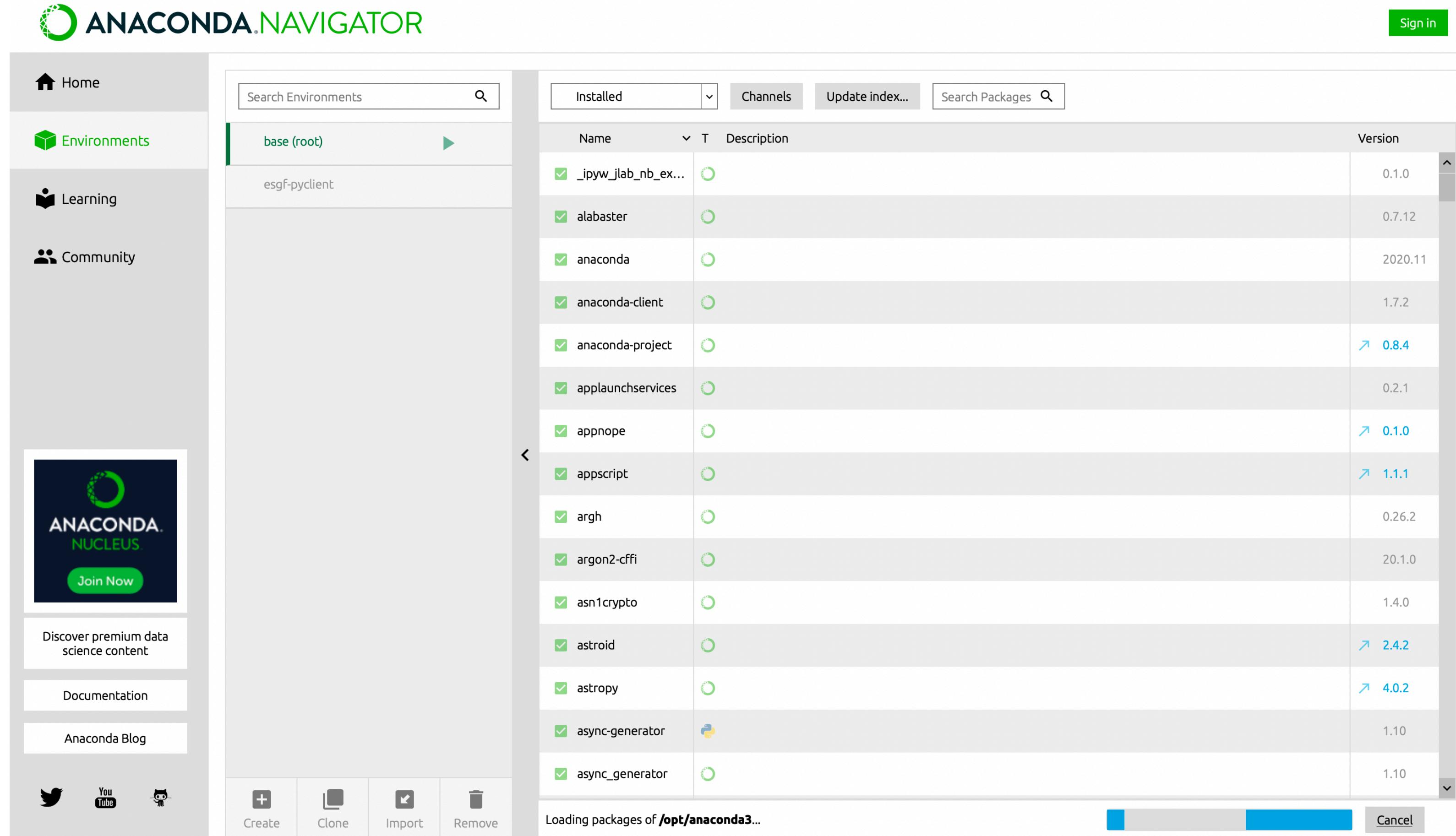
**PYTORCH**

An open-source deep learning framework using GPUs and CPUs that consists of fundamental tools and libraries for Python AI and machine learning development.

**scikit learn**

A powerful and versatile machine learning library for machine learning basics like classification, regression, and clustering. It includes both supervised and unsupervised ML algorithms with important functions like cross-validation and feature extraction. Scikit-learn is the most frequently downloaded machine learning library.

# Scientific-based Python distribution - Anaconda



OR

from the terminal:  
conda install ...  
conda install -c conda-forge ...

# Scientific-based Python distribution - Anaconda

The screenshot shows the Anaconda Navigator interface. On the left is a sidebar with icons for Home, Environments, Learning, and Community. A central grid displays applications: Datalore, IBM Watson Studio Cloud, JupyterLab, Jupyter Notebook, PyCharm Professional, Qt Console, Spyder, and Glueviz. Each application card includes a 'Launch' button. A 'Sign in' button is at the top right. A 'Join Now' button is visible on the PyCharm Professional card.

- Notepad++
- Vim
- Sublime Text
- **Spyder**
- Atom
- VS Code
- Pycharm
- Jupyter
- ...

| Editor             | Learning curve | Users                   | Benefits              |
|--------------------|----------------|-------------------------|-----------------------|
| Spyder             | pretty short   | Matlab and R background | mature, many features |
| Jupyter            | smooth         | teachers                | interactive           |
| Visual Studio Code | moderate       | scientists / developers | code quality          |
| PyCharm            | steep          | developers              | professional code     |

The screenshot shows the Spyder IDE interface. On the left is the code editor with a file named `main.py` containing Python code for reading netCDF files. The code uses pandas and numpy libraries to extract variables and save them as a MATLAB (.mat) file. The middle section is a help browser for the `pandas.DataFrame` class, providing its definition, type, and a detailed description of its two-dimensional tabular nature. The right section is the IPython console, showing the Python environment and command history.

```

16 def print_variables(dataset):
17     for d in dataset.variables:
18         desc = dataset.variables[d].name + ' : ' + dataset.variables[d].long_name
19         desc += ' [' + dataset.variables[d].units + ']' if 'units' in dataset.variables[d].ncattrs() else ''
20
21     print desc
22
23
24 def get_value(dataset, variable, cp):
25     # extract index for the station
26     m = cp[0]
27     n = cp[1]
28     stations = pd.DataFrame(dataset.variables['MNSTAT'][0], columns=['M', 'N'])
29     station_index = stations[(stations.M == m) & (stations.N == n)].index[0]
30
31     # extract julian date
32     start_time = datetime.strptime(dataset.variables['time'].units, 'seconds since %Y-%m-%d %H:%M:%S')
33     seconds_offset = np.array(dataset.variables['time'][:, :, 0], dtype='double')
34
35     time_vector = np.array([date2num(start_time + timedelta(seconds=s)) + 366 for s in seconds_offset])
36     time_vector = np.array([date2num(start_time + timedelta(seconds=s)) + 366 for s in seconds_offset], dtype='double')
37
38     # extract values
39     values = np.array([], dtype='double')
40     if variable == 'ZWL':
41         values = np.array(dataset.variables[variable][:, station_index], dtype='double')
42     elif variable == 'ZCURU':
43         values = np.array(dataset.variables[variable][:, 0, station_index], dtype='double')
44
45     data = {'data_nc': OrderedDict([
46         ('X', np.array(dataset.variables['XSTAT'][:, :, station_index], dtype='double')),
47         ('Y', np.array(dataset.variables['YSTAT'][:, :, station_index], dtype='double')),
48         ('XUnits', 'm'),
49         ('YUnits', 'm'),
50         ('Val', values),
51         ('Time', time_vector),
52         ('Name', dataset.variables[variable].long_name),
53         ('Units', dataset.variables[variable].units)])
54     }
55
56     return data
57
58
59 def save_mat(data, filename):
60     savemat(filename, data, oned_as='column')
61

```

# Integrated Development Environment - IDEs

Spyder

- A clear interface
- -> Matlab
- Variables and figures visualization

The screenshot shows the Visual Studio Code (VS Code) interface. The Explorer sidebar on the left lists files and folders, including `test_summary.py`, `test_analysis.py`, `test_read.py`, and `summary.py`. The Editor pane displays the `test_summary.py` file with Python code. The Terminal pane at the bottom shows a clean slate with the message "Hasta el momento, no se encontraron problemas en el área de trabajo." (Up to now, no problems were found in the workspace).

```
#!/usr/bin/env python
# coding: utf-8
from __future__ import (absolute_import, division,
                       print_function, unicode_literals)
try:
    # noinspection PyUnresolvedReferences, PyCompatibility
    from builtins import * # noqa
except ImportError:
    pass
import os
import matplotlib.pyplot as plt
from climate import summary, tests
from climate.util import plot
from climate.stats import empirical_distributions
from input import saih, tidal_model_driver
```

# Integrated Development Environment - IDEs

VS Code

```
#!/usr/bin/env python
# coding: utf-8
from __future__ import (absolute_import, division,
print_function, unicode_literals)

try:
    # noinspection PyUnresolvedReferences, PyCompatibility
    from builtins import *
except ImportError:
    pass

import os

from meteoceandataframe.meteoceandataframe import MetOceanDF
from preprocessing import missing_values
from report import latex, tests

def test_create_latex_document_granada_beach():
    location = 'granada_beach'
    drivers = ['wave', 'wind', 'astronomical_tide', 'sea_level_pressure']

    data = []
    # Data
    for driver in drivers:
        modf = os.path.join(tests.current_path, '...', '...', 'inputadapter', 'tests', 'output', 'modf',
                           '{}_{}.modf'.format(location, driver))
        data.append(MetOceanDF.read_file(modf))

    # Config report file
    template = os.path.join(tests.current_path, '...', 'templates', 'latex', '{}.conf'.format(location))

    latex.create_document(data, template, output_title=location)

def test_create_latex_document_cancun():
    location = 'cancun'
    drivers = ['wave', 'wind', 'astronomical_tide', 'sea_level_pressure']

    data = []
    # Data
```

Run console or debugger to view available data

Documentation: protocol x

No documentation found.

- Projects
- External python interpreters (ssh)
- The ability to refactor (rename) a variable throughout your code, without using a global find and replace.
- Fully integrated Git repository for version control and team collaboration.

# Integrated Development Environment - IDEs

Pycharm

The Jupyter Notebook is a web-based interactive computing platform that allows users to author data- and code-driven narratives that combine live code, equations, narrative text, visualizations, interactive dashboards and other media.

## Jupyter Notebook

The Jupyter Notebook is a web-based interactive computing platform that allows users to author data- and code-driven narratives that combine live code, equations, narrative text, visualizations, interactive dashboards and other media.

# Integrated Development Environment - IDEs

## Jupyter

```
In [57]:  
from sympy import diff, sin, exp  
  
diff(sin(x)*exp(x), x)  
  
Out[57]:  $e^x \sin(x) + e^x \cos(x)$ 
```

Compute  $\int(e^x \sin(x) + e^x \cos(x)) dx$

```
In [58]:  
from sympy import integrate, cos  
  
integrate(exp(x) * sin(x) + exp(x) * cos(x), x)  
  
Out[58]:  $e^x \sin(x)$ 
```

Compute  $\int_{-\infty}^{\infty} \sin(x^2) dx$

```
In [59]:  
from sympy import oo  
  
integrate(sin(x**2), (x, -oo, oo))  
  
Out[59]: 
$$\frac{\sqrt{2}\sqrt{\pi}}{2}$$

```

# Cloud-based software

---

- Binder
- Kaggle Kernels
- Google Colaboratory (Colab)
- Microsoft Azure Notebooks -> Github Codespaces (beta)\*
- CoCalc
- Datalore
  - <https://www.dataschool.io/cloud-services-for-jupyter-notebook/>
- Datalore
  - <https://docs.google.com/spreadsheets/d/12thaaxg1ldr3iwst8qyasndso8sjdpd6m9mbcgthfn0/edit#gid=1505836451>

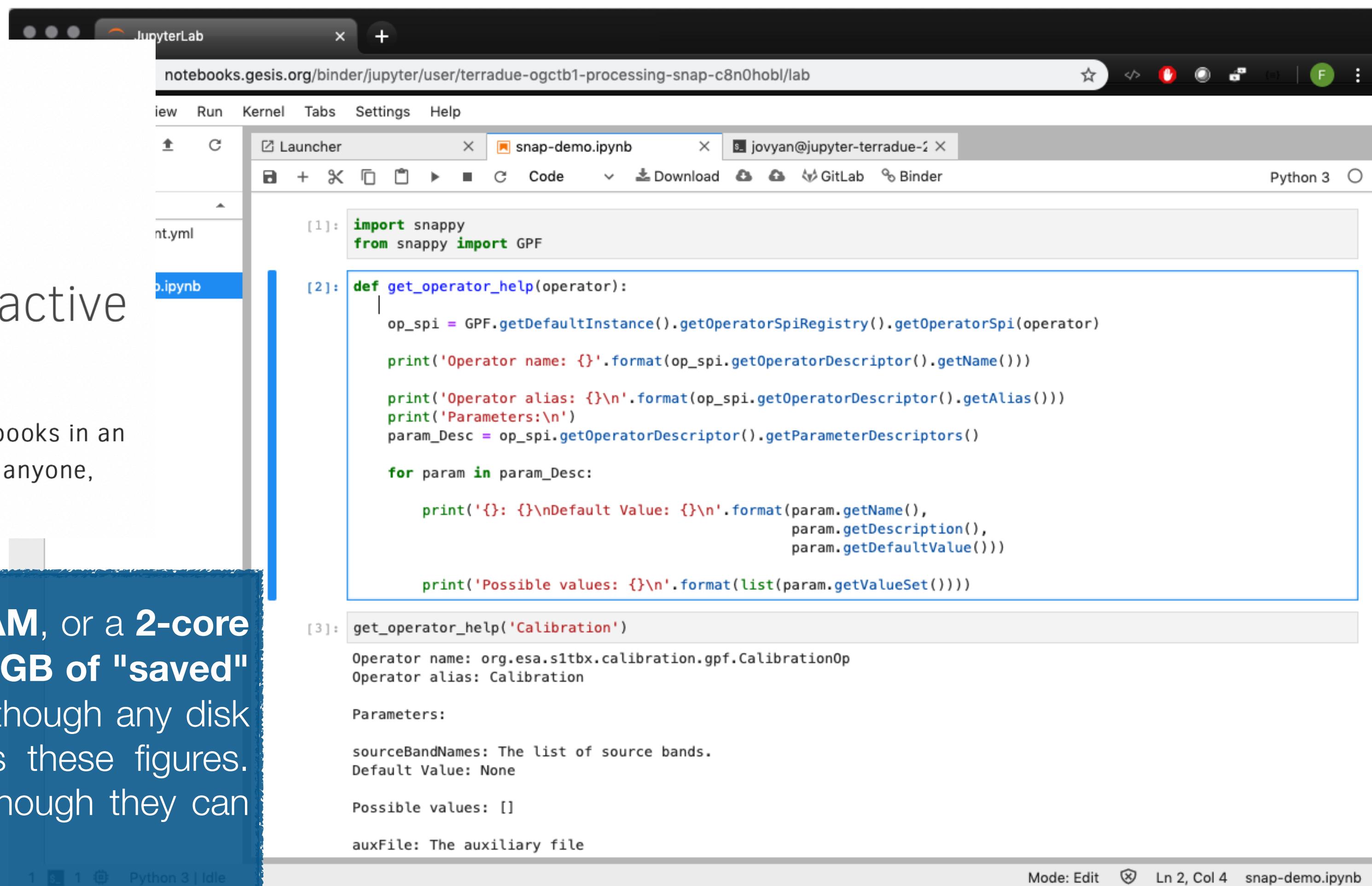
# Cloud-based software | Binder



Turn a Git repo into a collection of interactive notebooks

Have a repository full of Jupyter notebooks? With Binder, open those notebooks in an executable environment, making your code immediately reproducible by anyone, anywhere.

You can access either a **4-core CPU with 17 GB of RAM**, or a **2-core CPU with 14 GB of RAM plus a GPU**. You will have **5 GB of "saved" disk space and 17 GB of "temporary" disk space**, though any disk space used by your dataset does not count towards these figures. Sessions will shut down after 60 minutes of inactivity, though they can run for up to 9 hours.



The screenshot shows a JupyterLab interface with a tab bar at the top labeled 'notebooks.gesis.org/binder/jupyter/user/terradue-ogctb1-processing-snap-c8n0hob1/lab'. The main area displays a Python notebook titled 'snap-demo.ipynb'. The code in the notebook is as follows:

```
[1]: import snappy
from snappy import GPF

[2]: def get_operator_help(operator):
    op_spi = GPF.getDefaultInstance().getOperatorSpiRegistry().getOperatorSpi(operator)

    print('Operator name: {}'.format(op_spi.getOperatorDescriptor().getName()))

    print('Operator alias: {}'.format(op_spi.getOperatorDescriptor().getAlias()))
    print('Parameters:\n')
    param_Desc = op_spi.getOperatorDescriptor().getParameterDescriptors()

    for param in param_Desc:

        print('{}: {} \nDefault Value: {} \n'.format(param.getName(),
                                                     param.getDescription(),
                                                     param.getDefaultValue()))

    print('Possible values: {}'.format(list(param.getValueSet())))

[3]: get_operator_help('Calibration')
Operator name: org.esa.s1tbx.calibration.gpf.CalibrationOp
Operator alias: Calibration

Parameters:

sourceBandNames: The list of source bands.
Default Value: None

Possible values: []

auxFile: The auxiliary file
```

At the bottom right of the interface, it says 'Mode: Edit' and 'Ln 2, Col 4 snap-demo.ipynb'.

# Cloud-based software | Kaggle

K tutorial  
File Edit Insert Run Help Draft Saved

+ ADD DATASET ✓ COMMIT ↵

## PyCon 2018: Using pandas for Better (and Worse) Data Science

GitHub: <https://github.com/justmarkham/pycon-2018-tutorial>

```
In[1]:  
import matplotlib.pyplot as plt  
import pandas as pd  
pd.__version__  
  
Out[1]:  
'0.23.4'
```

**Dataset: Stanford Open Policing Project ([video](#))**

```
# ri stands for Rhode Island  
ri = pd.read_csv('../input/police.csv')
```

You can access either a **4-core CPU with 17 GB of RAM**, or a **2-core CPU with 14 GB of RAM plus a GPU**. You will have **5 GB of "saved" disk space and 17 GB of "temporary" disk space**, though any disk space used by your dataset does not count towards these figures. Sessions will shut down after 60 minutes of inactivity, though they can run for up to 9 hours.

## Datasets

Explore, analyze, and share quality data. [Learn more about data types, creating, and collaborating.](#)

+ New Dataset

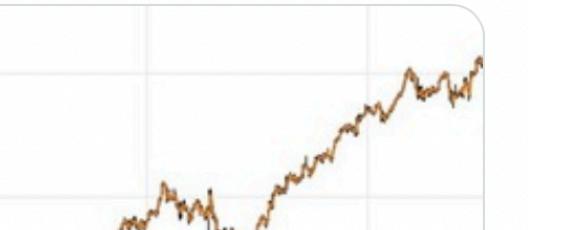
Search datasets

Filters

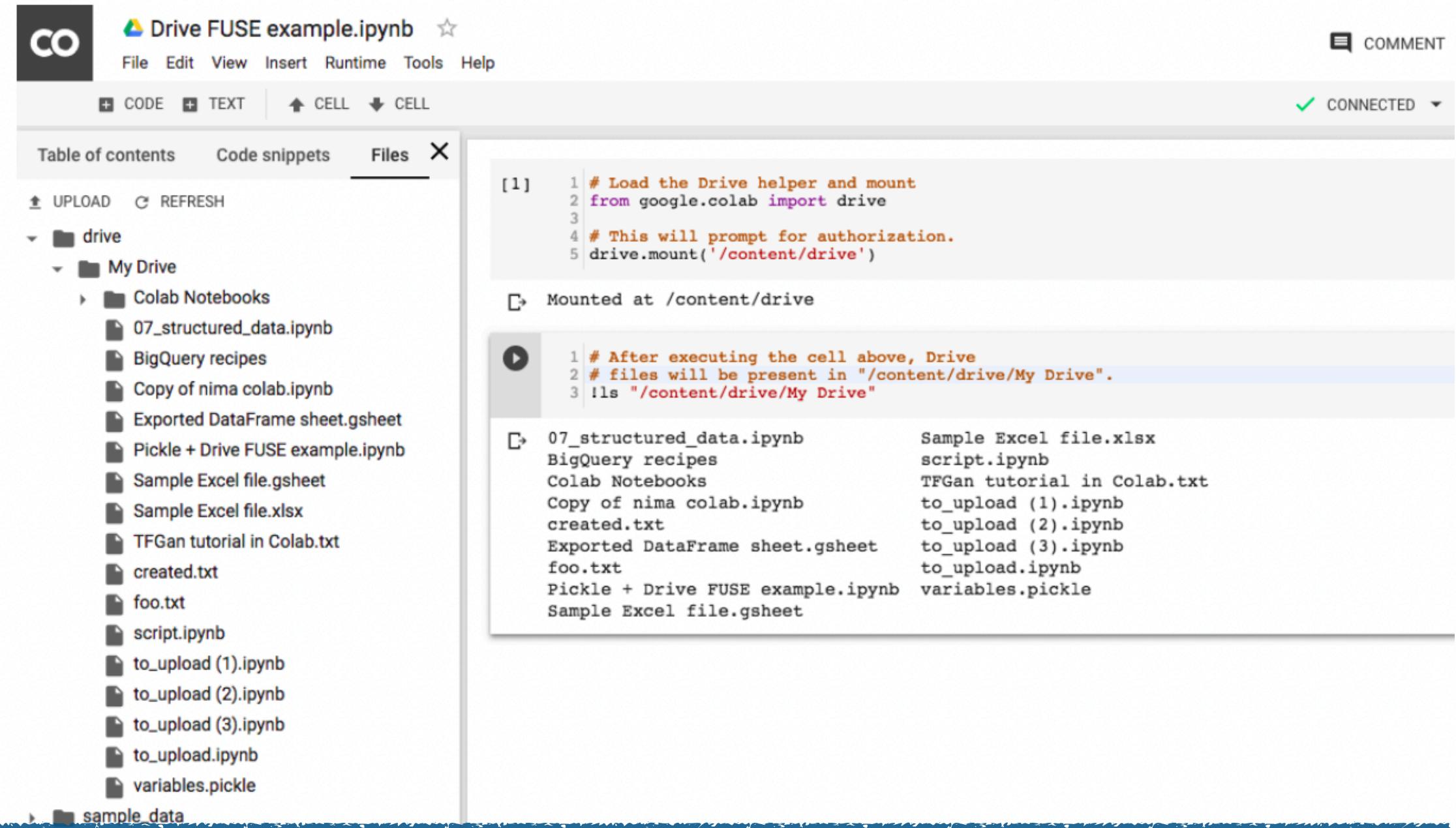
Datasets Tasks Computer Science Education Classification Computer Vision NLP Data Visualization

### Trending Datasets

See All

|   |  |   |   |
|---|--|---|---|
| <br>National Stock Exchange Sensex<br>Dataset - Sensex<br>Suman Dey · Updated 3 hours ago<br>Usability 9.7 · 2 MB<br>1 Task · 1 File (CSV) | <br>List of .gov.uk Domain Names<br>Peter Nooteboom · Updated 4 hours...<br>Usability 9.4 · 53 KB<br>1 File (CSV) | <br>Kerala_Loksabha_election<br>Prabin Raj · Updated 6 hours ago<br>Usability 10.0 · 9 KB<br>1 Task · 1 File (CSV) | <br>Bank NIFTY stock trades<br>Abhirukth Chakravarthy · Updated 6...<br>Usability 10.0 · 6 MB<br>1 Task · 2 Files (CSV, other) |
|---|--|---|---|

# Cloud-based software | Google Colab



```
# Load the Drive helper and mount
from google.colab import drive
# This will prompt for authorization.
drive.mount('/content/drive')

# After executing the cell above, Drive
# files will be present in "/content/drive/My Drive".
!ls "/content/drive/My Drive"

07_structured_data.ipynb      Sample Excel file.xlsx
BigQuery recipes               script.ipynb
Colab Notebooks                TFGan tutorial in Colab.txt
Copy of nima colab.ipynb       to_upload (1).ipynb
created.txt                     to_upload (2).ipynb
Exported DataFrame sheet.gsheet to_upload (3).ipynb
Pickle + Drive FUSE example.ipynb variables.pickle
Sample Excel file.gsheet
Sample Excel file.xlsx
TFGan tutorial in Colab.txt
foo.txt
Pickle + Drive FUSE example.ipynb
Sample Excel file.gsheet
```

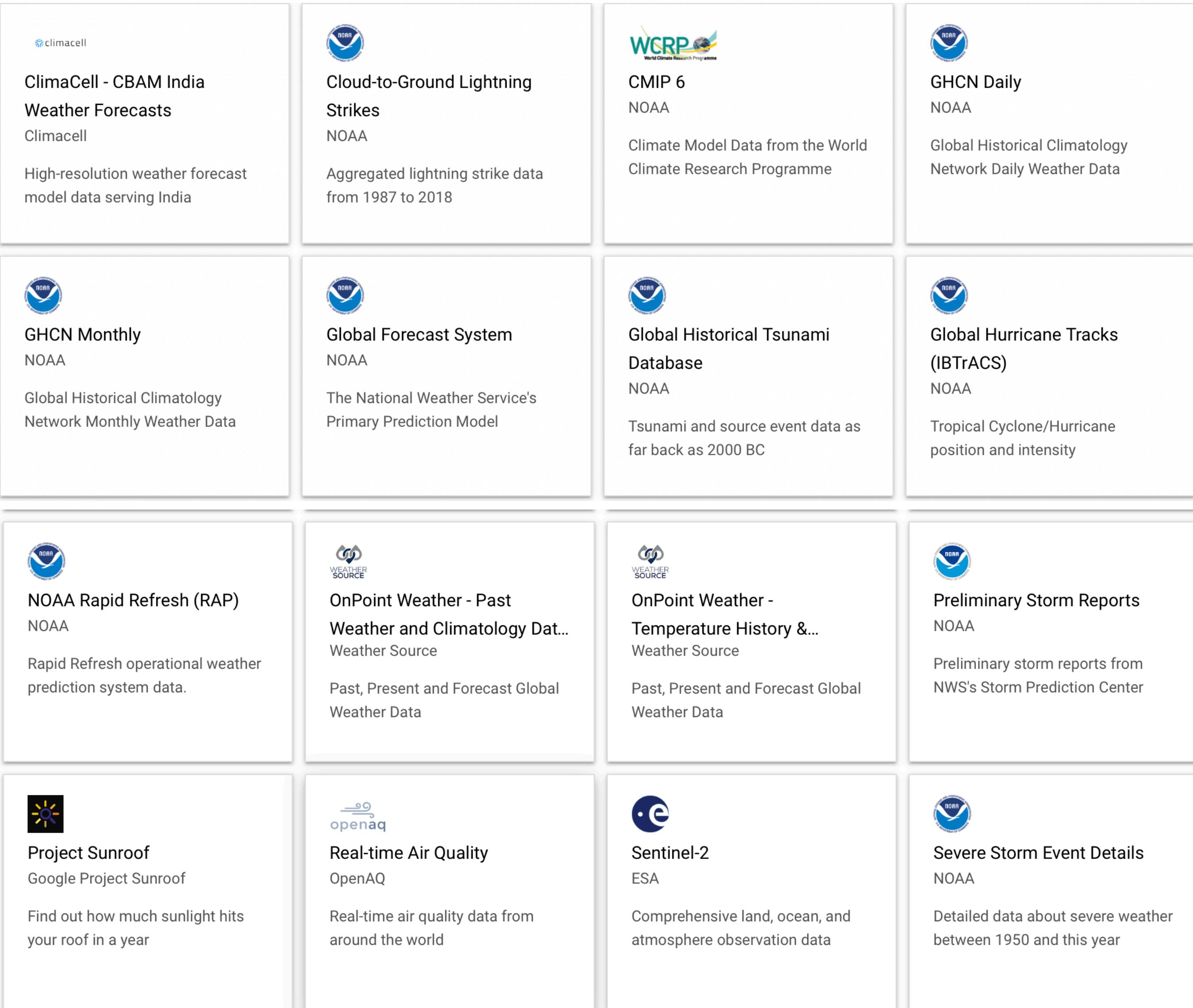
Colab does give you access to a GPU or a TPU. Up to 15 GB of disk space for storing your datasets. Sessions will shut down after 60 minutes of inactivity, though they can run for up to 12 hours.

*Presently they provide **2-core VMs with 12 Gb of RAM**, and either **50 Gb (non-GPU kernel)** or **350 Gb (GPU kernel)** of disk space.*



Earth Engine

Earth Engine's public data archive includes more than forty years of historical imagery and scientific datasets, updated daily and available for online analysis.



|  |   |   |  |
|--|---|---|--|
|  <b>ClimaCell - CBAM India</b><br>Weather Forecasts<br>Climacell<br><br>High-resolution weather forecast model data serving India |  <b>Cloud-to-Ground Lightning Strikes</b><br>NOAA<br><br>Aggregated lightning strike data from 1987 to 2018                              |  <b>CMIP 6</b><br>NOAA<br><br>Climate Model Data from the World Climate Research Programme                                       |  <b>GHCN Daily</b><br>NOAA<br><br>Global Historical Climatology Network Daily Weather Data                        |
|  <b>GHCN Monthly</b><br>NOAA<br><br>Global Historical Climatology Network Monthly Weather Data                                  |  <b>Global Forecast System</b><br>NOAA<br><br>The National Weather Service's Primary Prediction Model                                  |  <b>Global Historical Tsunami Database</b><br>NOAA<br><br>Tsunami and source event data as far back as 2000 BC                 |  <b>Global Hurricane Tracks (IBTrACS)</b><br>NOAA<br><br>Tropical Cyclone/Hurricane position and intensity      |
|  <b>NOAA Rapid Refresh (RAP)</b><br>NOAA<br><br>Rapid Refresh operational weather prediction system data.                       |  <b>OnPoint Weather - Past Weather and Climatology Data...</b><br>Weather Source<br><br>Past, Present and Forecast Global Weather Data |  <b>OnPoint Weather - Temperature History &amp;...</b><br>Weather Source<br><br>Past, Present and Forecast Global Weather Data |  <b>Preliminary Storm Reports</b><br>NOAA<br><br>Preliminary storm reports from NWS's Storm Prediction Center   |
|  <b>Project Sunroof</b><br>Google Project Sunroof<br><br>Find out how much sunlight hits your roof in a year                    |  <b>Real-time Air Quality</b><br>OpenAQ<br><br>Real-time air quality data from around the world  |  <b>Sentinel-2</b><br>ESA<br><br>Comprehensive land, ocean, and atmosphere observation data                                    |  <b>Severe Storm Event Details</b><br>NOAA<br><br>Detailed data about severe weather between 1950 and this year |

# Cloud-based software | Datalore

You will have access to a **2-core CPU with 4 GB of RAM**, and **10 GB of disk space**. Sessions will shut down after 60 minutes of inactivity, though there is no specific limit on the length of individual sessions. You can use the service for up to 120 hours per month.

The screenshot shows the Datalore web interface. At the top, there's a dark header bar with a logo, 'File', 'Tools', 'Kernel', 'View', 'Help', a 'tutorial' link, a 'forum' button, and a user profile icon. Below the header is a title card for 'PyCon 2018: Using pandas for Better (and Worse) Data Science' with a GitHub link. The main area is a code editor with a notebook interface. A code cell imports matplotlib and pandas, prints the pandas version ('0.23.4'), and loads a dataset from 'police.csv'. A preview of the dataset is shown as a table:

|   | stop_date  | stop_time | county_name | driver_gender | driver_age_raw | driver_age | driver_race | violation_raw |
|---|------------|-----------|-------------|---------------|----------------|------------|-------------|---------------|
| 0 | 2005-01-02 | 01:55     | nan         | M             | 1985.0         | 20.0       | White       | Speeding      |
| 1 | 2005-01-18 | 08:15     | nan         | M             | 1965.0         | 40.0       | White       | Speeding      |

At the bottom of the interface, there are status indicators: 'Live computation' (green), 'Calculated: 85', 'In Process: 0', 'Errors: 0', 'Computation status: running on agent', 'Python status: PyAgent is re...', 'CPU: 3%', and 'FreeMem: 2838MB'.

# Cloud-based software | Google Colab

The screenshot shows the Google Colab interface. At the top, there's a toolbar with File, Edit, View, Insert, Runtime, Tools, and Help. Below it is a navigation bar with CODE, TEXT, CELL, and CELL. A 'COMMENT' button is on the right. A 'CONNECTED' status indicator with a green checkmark and a dropdown menu is also present. The main area shows a code cell with the following Python code:

```
[1]: # Load the Drive helper and mount
      from google.colab import drive
      # This will prompt for authorization.
      drive.mount('/content/drive')

[2]: # After executing the cell above, Drive
      # files will be present in "/content/drive/My Drive".
      !ls "/content/drive/My Drive"
```

Below the code cell, a file browser sidebar shows the contents of 'My Drive'. It includes 'Colab Notebooks', '07\_structured\_data.ipynb', 'BigQuery recipes', 'Copy of nima colab.ipynb', 'Exported DataFrame sheet.gsheet', 'Pickle + Drive FUSE example.ipynb', 'Sample Excel file.gsheet', 'Sample Excel file.xlsx', 'TFGan tutorial in Colab.txt', 'created.txt', 'foo.txt', 'script.ipynb', 'to\_upload (1).ipynb', 'to\_upload (2).ipynb', 'to\_upload (3).ipynb', 'to\_upload.ipynb', and 'variables.pickle'. There are also 'sample\_data' and '07\_structured\_data.ipynb' entries.

## Earth Engine

Earth Engine's public data archive includes more than forty years of historical imagery and scientific datasets, updated daily and available for online analysis.

The grid displays 16 cards, each representing a different cloud-based service or dataset:

- Climacell**: ClimaCell - CBAM India Weather Forecasts. NOAA. High-resolution weather forecast model data serving India.
- WCRP**: Cloud-to-Ground Lightning Strikes. NOAA. Aggregated lightning strike data from 1987 to 2018.
- CMIP 6**: Climate Model Data from the World Climate Research Programme. NOAA. Climate Research Programme.
- GHCN Daily**: Global Historical Climatology Network Daily Weather Data. NOAA.
- GHCN Monthly**: Global Historical Climatology Network Monthly Weather Data. NOAA.
- Global Forecast System**: The National Weather Service's Primary Prediction Model. NOAA.
- Global Historical Tsunami Database**: Tsunami and source event data as far back as 2000 BC. NOAA.
- Global Hurricane Tracks**: Tropical Cyclone/Hurricane position and intensity. NOAA.
- NOAA Rapid Refresh (RAP)**: Rapid Refresh operational weather prediction system data. NOAA.
- OnPoint Weather - Past Weather and Climatology Data**: Past, Present and Forecast Global Weather Data. Weather Source.
- OnPoint Weather - Temperature History &...**: Past, Present and Forecast Global Weather Data. Weather Source.
- Preliminary Storm Reports**: Preliminary storm reports from NWS's Storm Prediction Center. NOAA.
- Project Sunroof**: Find out how much sunlight hits your roof in a year. Google Project Sunroof.
- Real-time Air Quality**: Real-time air quality data from around the world. OpenAQ.
- Sentinel-2**: Comprehensive land, ocean, and atmosphere observation data. ESA.
- Severe Storm Event Details**: Detailed data about severe weather between 1950 and this year. NOAA.

# Course docs

---

- [https://github.com/andreall/python\\_course-dicca.git](https://github.com/andreall/python_course-dicca.git)
- [https://mybinder.org/v2/gh/andreall/python\\_course-dicca.git/HEAD](https://mybinder.org/v2/gh/andreall/python_course-dicca.git/HEAD)

## Additional resources

- <https://projectpythia.org>
- <https://dabeaz-course.github.io/practical-python/Notes/Contents.html>