

Python for Scientists

3rd Edition | February & March, 2023

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

Andrea Lira Loarca
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.



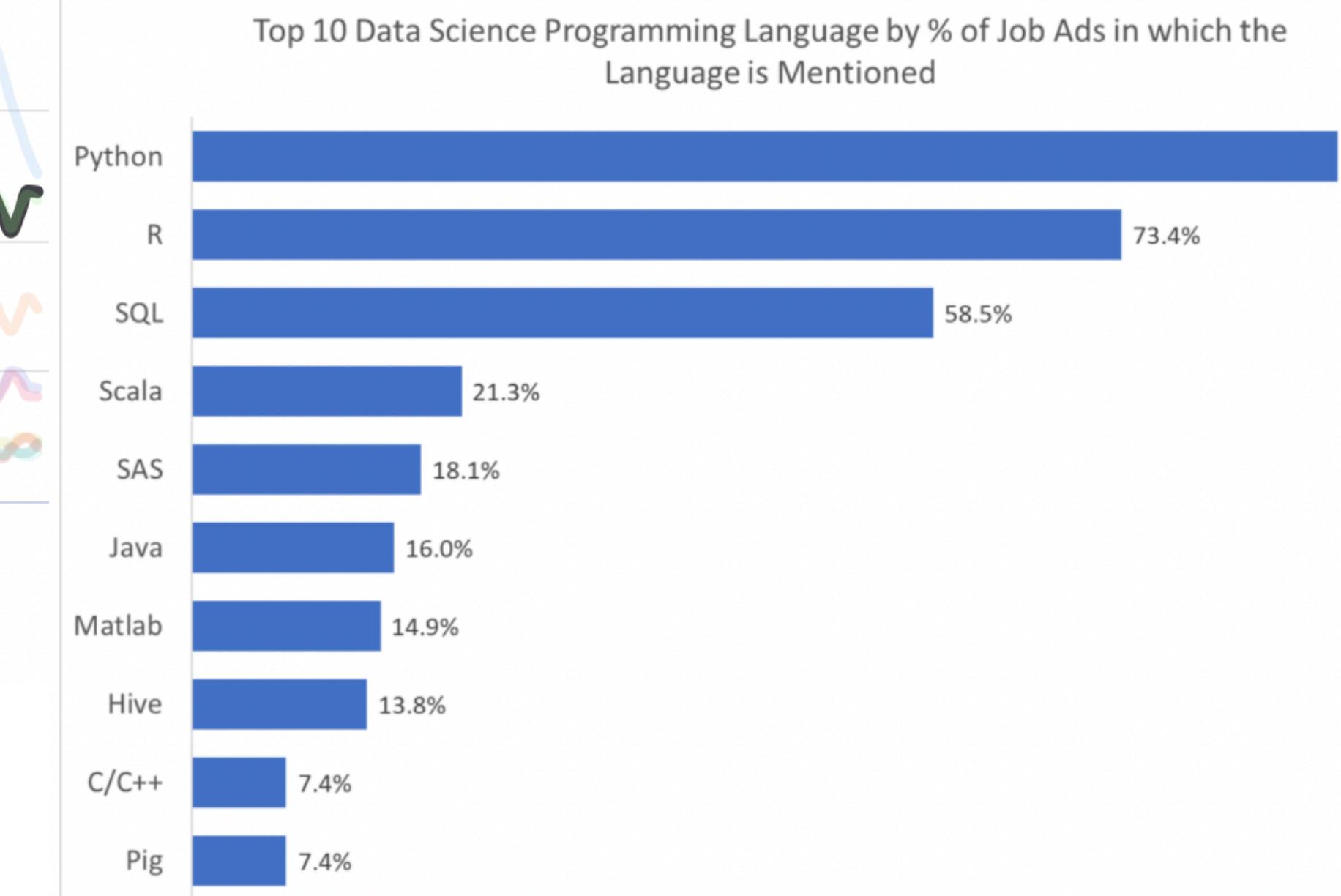
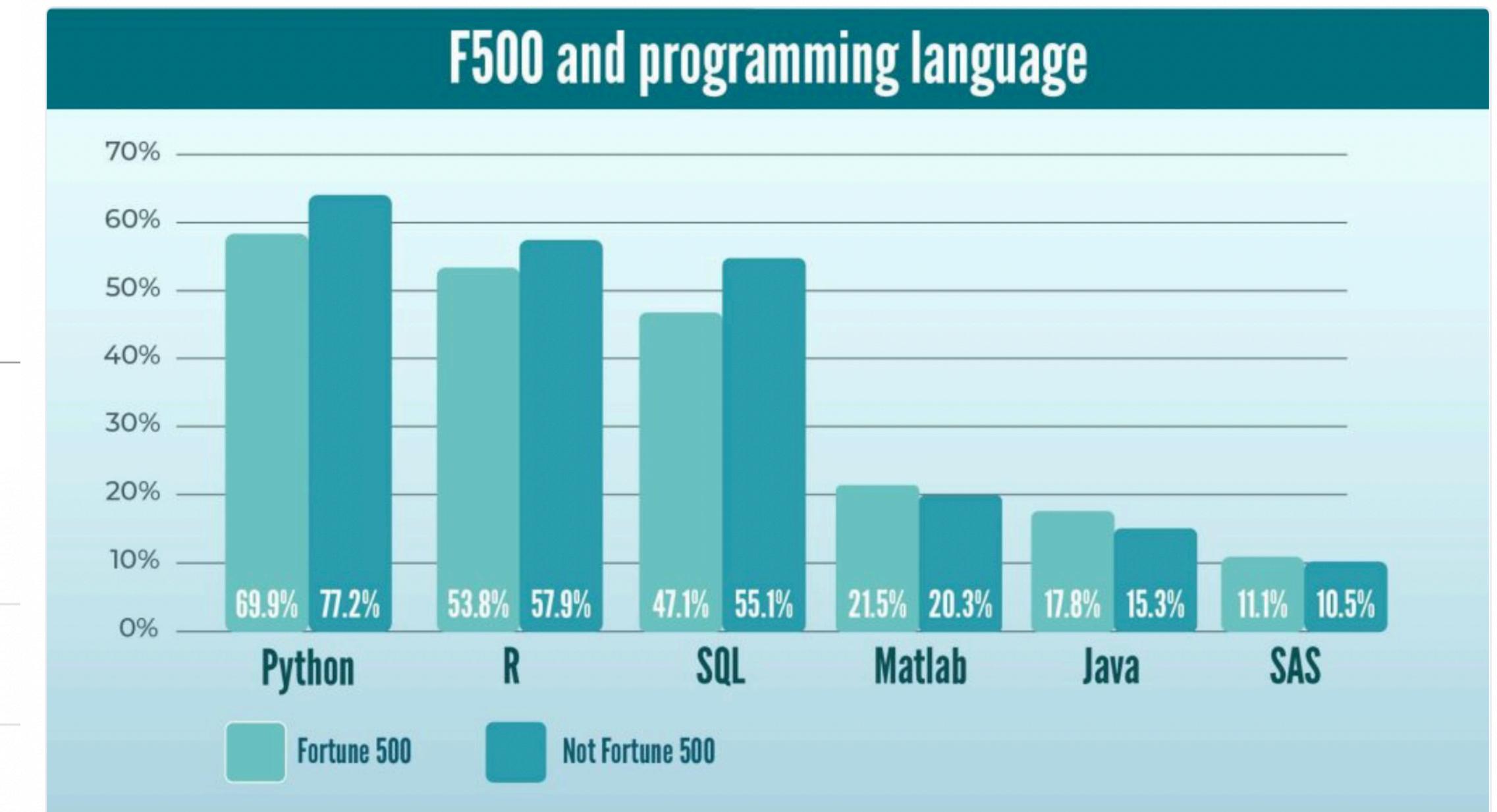
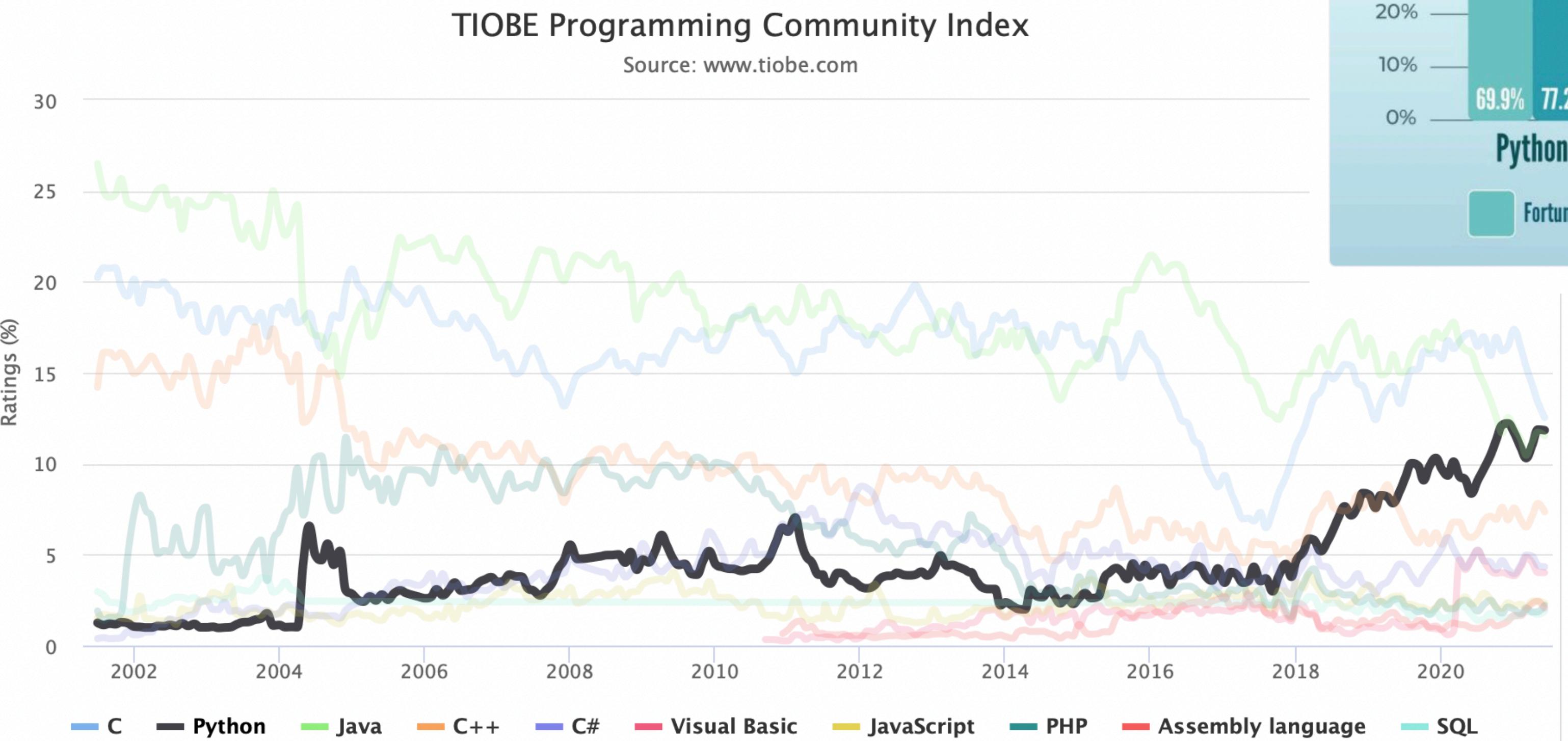
Credit: Illustration by the Project Twins

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



R's strengths: Statistical analysis: data manipulation, visualization.
Machine learning

Python vs. Other Languages

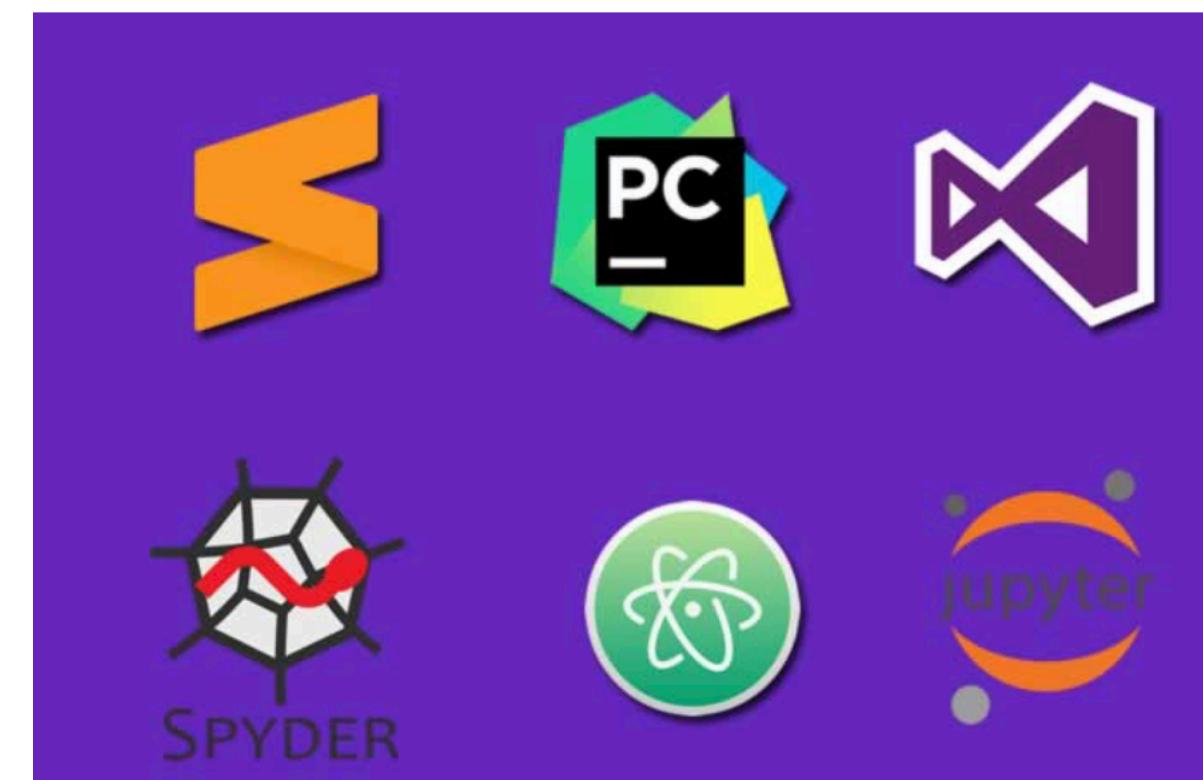
C++'s strengths: High performance speed. Structure-based coding

	 Python	 JavaScript (js)	 C++	 PHP	 Go
Learning Curve	Quite easy	Comparatively harder	Hard	Quite easy	Hard
Cross-Platform	Yes	Yes	Yes	Yes	Yes
Compatibility with operating systems	Windows, macOS, and Linux	Windows, macOS, and Linux	Mac, Windows, Linux, BSD and more	Windows, Linux	Mac, Windows, Linux, BSD and more
Nature	Dynamically typed	Dynamically typed	Statically typed	Dynamically typed	Statically typed
Performance	Comparatively Slower	Fast	Fast	Comparatively Slower	Fast
Type	Interpreted	Interpreted	Complied	Interpreted	Complied
Libraries	Yes	Not many	Not many	Not many	Not many

	 Perl	 Ruby	 C#	 Scala	 R
Learning Curve	Hard	Quite easy	Comparatively harder	Quite easy	Hard
Cross-Platform	Yes	No	Yes	Yes	Yes
Compatibility with other OS	Windows, macOS, and Linux	Linux, Solaris, FreeBSD, and macOS	Windows, macOS, and Linux	Windows, MacOS, Linux, Android and more	Windows, macOS, and Linux
Nature	Dynamically typed	Dynamically typed	Statically typed	Statically typed	Dynamically typed
Performance	Fast	Slow	Fast	Fast	Slow
Type	Interpreted	Complied	Complied	Complied	Interpreted
Libraries	Yes, as modules	Yes	Not many	Not many	Yes

How to run Python?

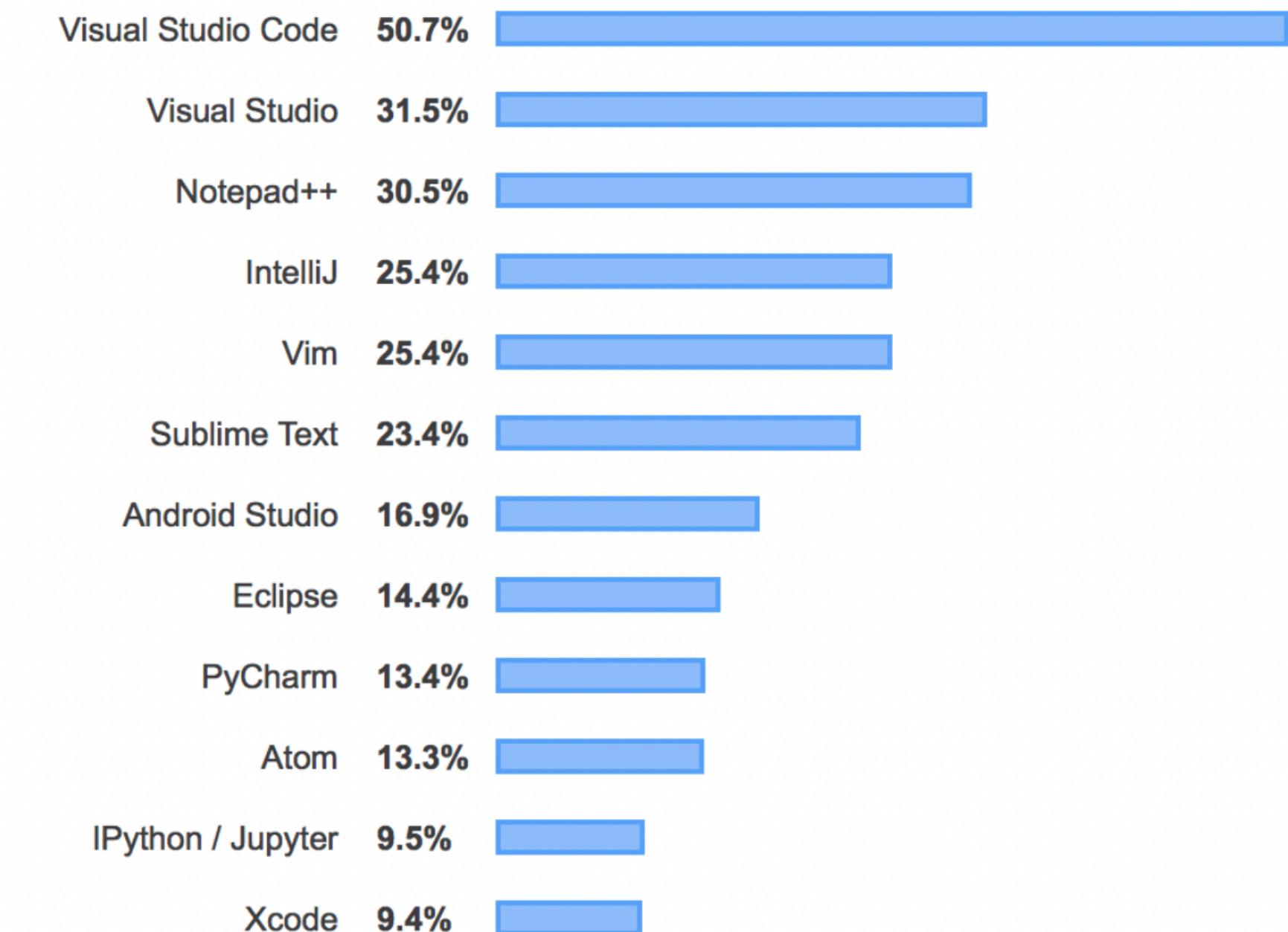
1. Distribution - set of libraries, packages



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 four main sections: "ANACONDA NAVIGATOR" (Desktop Portal to Data Science), "ANACONDA PROJECT" (Portable Data Science Encapsulation), "DATA SCIENCE LIBRARIES" (listing various tools like Jupyter, NumPy, Matplotlib, etc.), and "CONDA" (Data Science Package & Environment Manager). The "DATA SCIENCE LIBRARIES" section includes sub-sections for Data Science IDEs, Analytics & Scientific Computing, Visualization, and Machine Learning.

ANACONDA DISTRIBUTION
Most Trusted Distribution for Data Science

ANACONDA NAVIGATOR
Desktop Portal to Data Science

ANACONDA PROJECT
Portable Data Science Encapsulation

DATA SCIENCE LIBRARIES

Data Science IDEs	Analytics & Scientific Computing	Visualization	Machine Learning
jupyter spyder jupyterlab R Studio	NumPy SciPy Numba pandas DASK	Bokeh HoloViews DataShader matplotlib	TensorFlow learn H2O.ai theano

...and many more!

CONDA
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 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

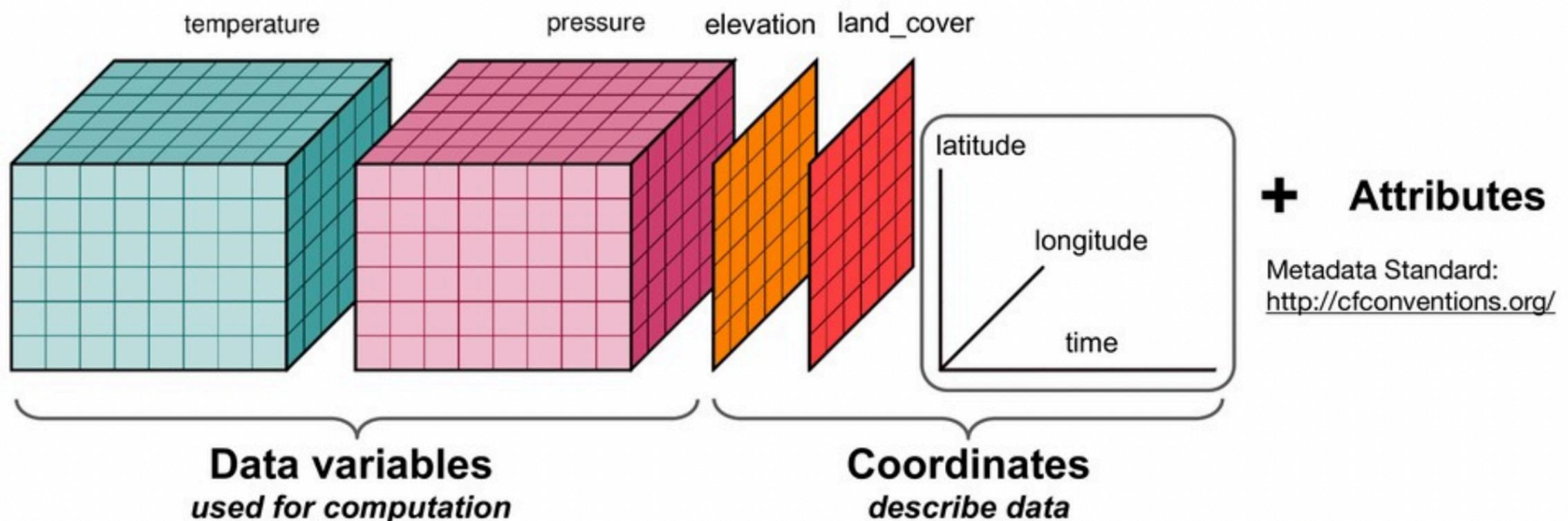
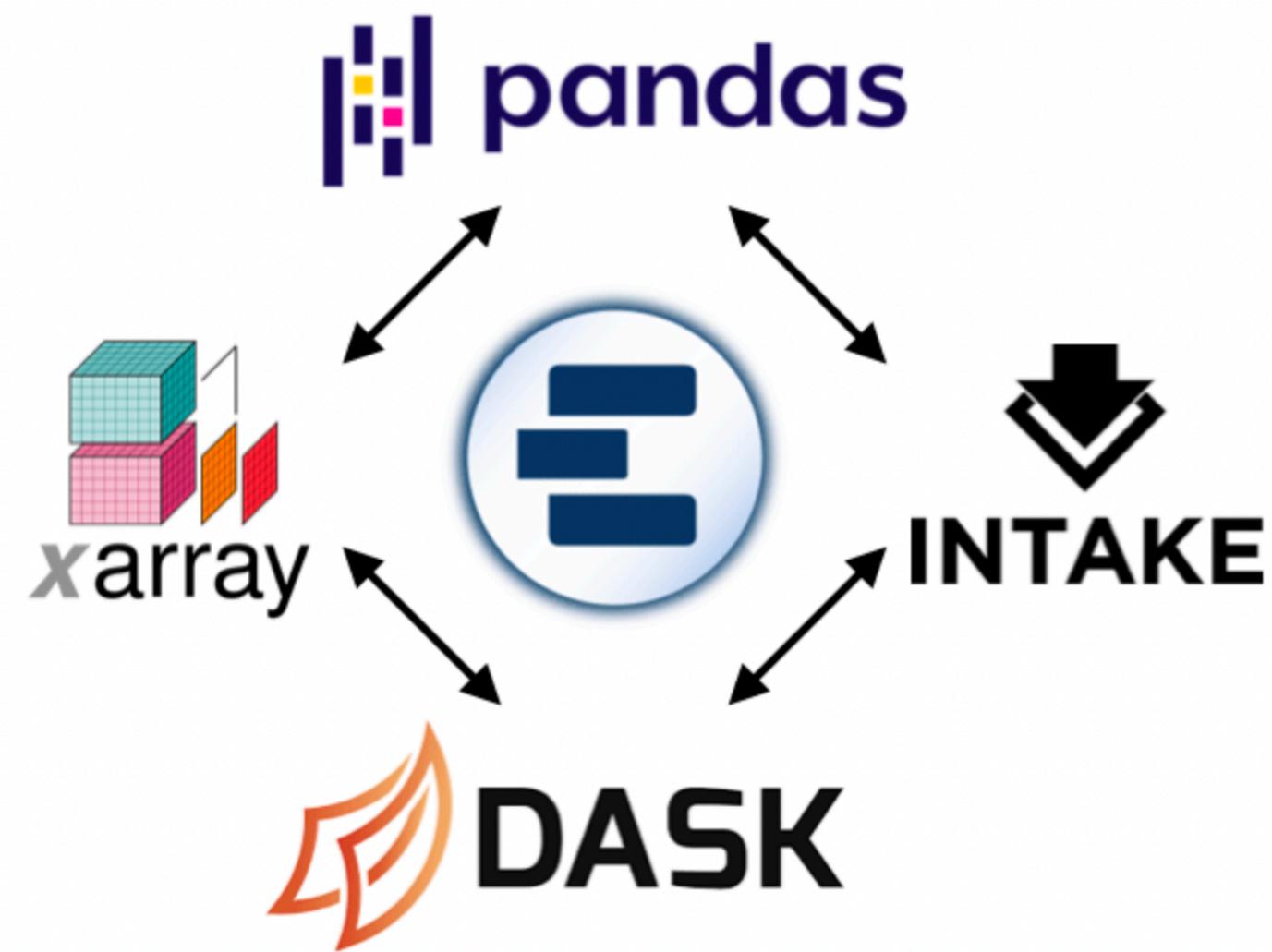


INTAKE
A lightweight package for finding, investigating, loading and disseminating data



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.

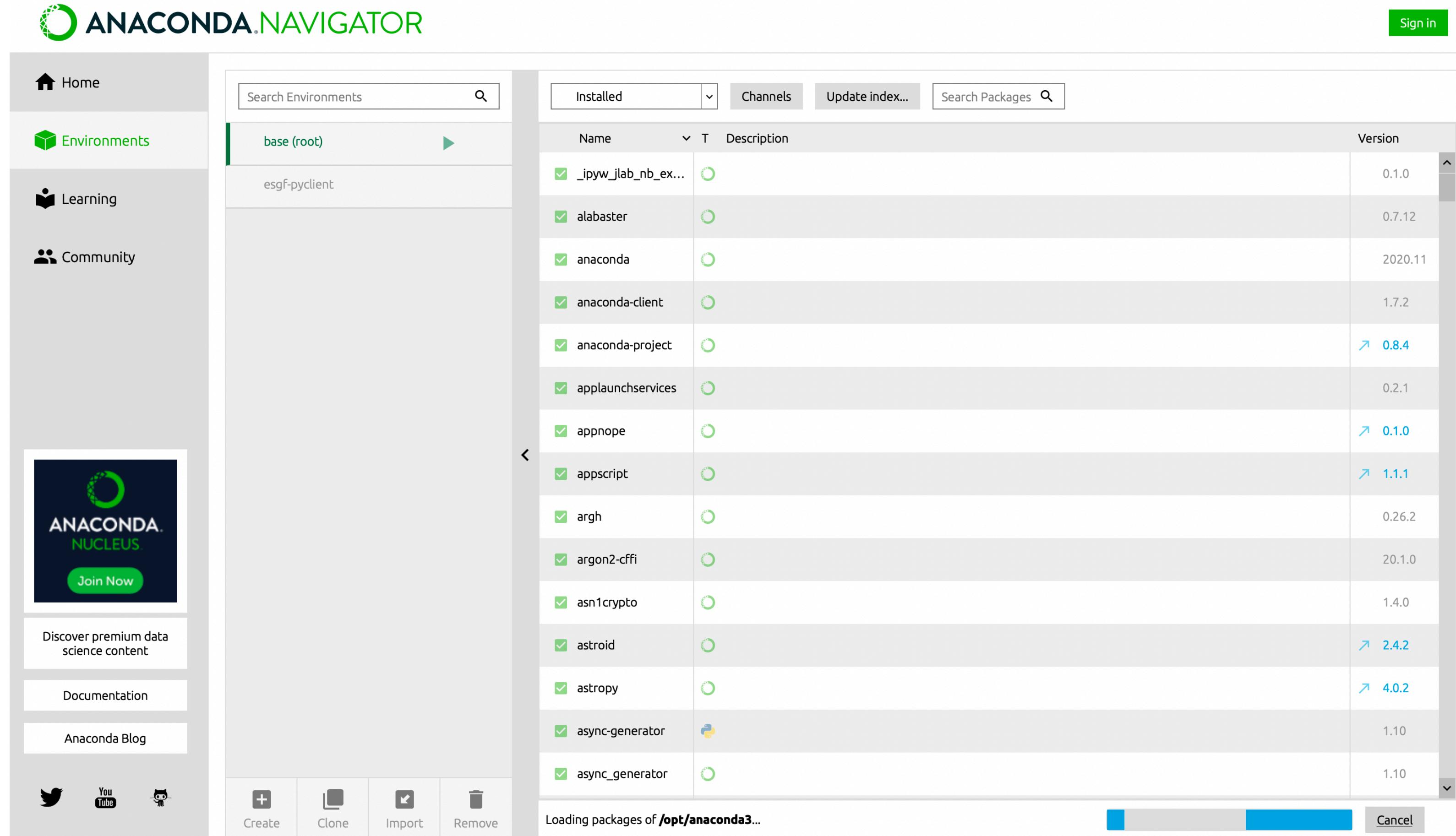
PYTORCH

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 Python IDE interface. On the left, the code editor displays a Python script named `main.py` containing functions for printing variables, getting values, and saving data to a matrix. A specific line of code is highlighted in yellow: `stations = pd.DataFrame(dataset.variables['MNSTAT'][0], columns=['M', 'N'])`. The right side of the interface includes a help browser for the `DataFrame` class, which provides its definition and type information. Below the help browser is the IPython console, which shows the Python version, license information, and a help command for the `object?` method.

```

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()
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
- 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 a blue header bar.

```
#!/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

- Kaggle (also GPU)
 - Deepnote
 - Google Colab
 - Microsoft Azure Notebooks -> Github Codespaces
 - Amazon SageMaker Studio Lab
 - <https://www.dataschool.io/cloud-services-for-jupyter-notebook/>
 - <https://www.kdnuggets.com/2022/04/top-5-free-cloud-notebooks-2022.html>
- [https://docs.google.com/spreadsheets/d/
12thaaxg1ldr3iwst8qyasndso8sjdpd6m9mbcgthfn0/edit#gid=1505836451](https://docs.google.com/spreadsheets/d/12thaaxg1ldr3iwst8qyasndso8sjdpd6m9mbcgthfn0/edit#gid=1505836451)

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.](#)

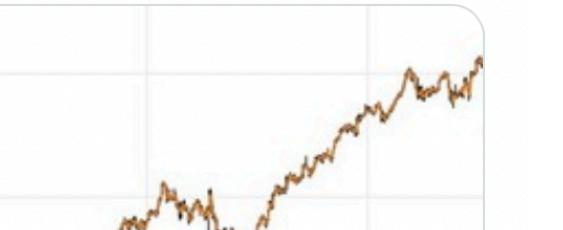
+ New Dataset

Search datasets Filters

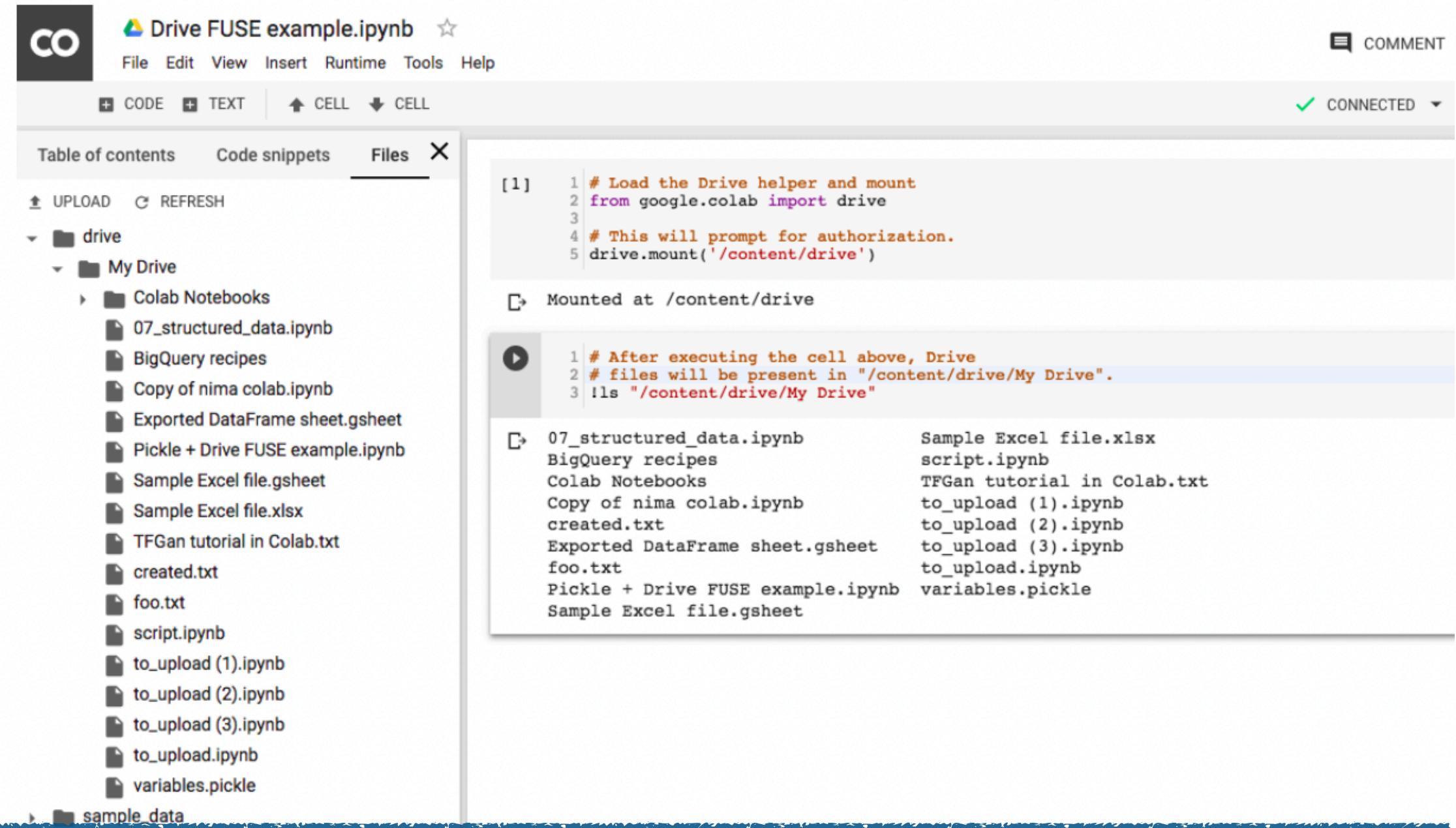
Datasets Tasks Computer Science Education Classification Computer Vision NLP Data Visualization

Trending Datasets

See All

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

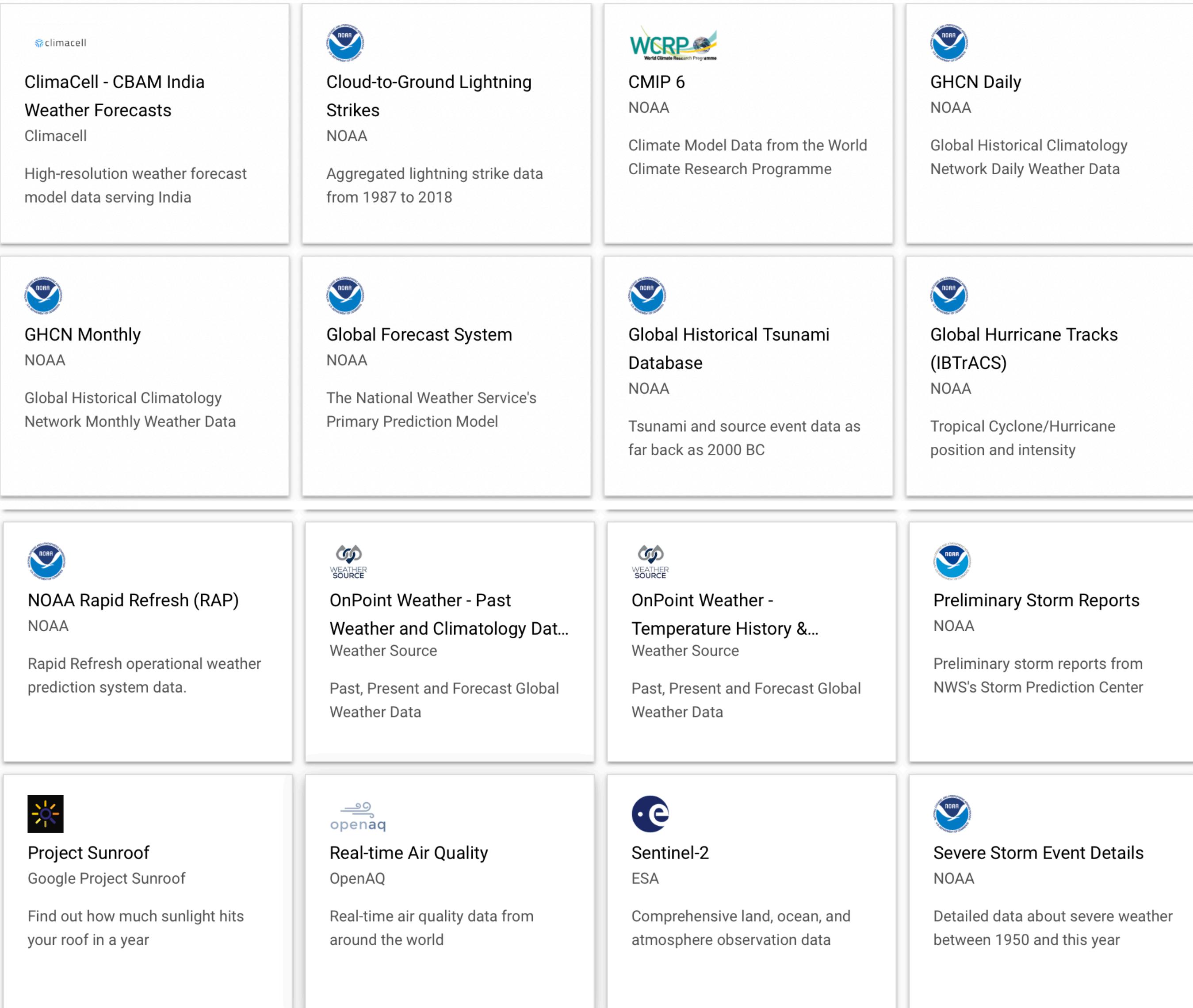
Colab does give you access to a GPU or a TPU. Up to 15 GB of disk space for storing your datasets. Sessions will shut down after 60 minutes of inactivity, though they can run for up to 12 hours.

*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.



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

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"
```

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.

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

Course docs

- https://github.com/andreall/python_course-dicca.git

Additional resources

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