

PYTHON SETUP

Python is becoming the most popular programming language for ML

TechVidvan

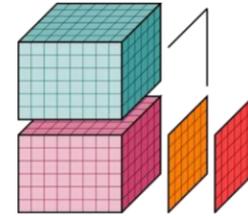
Why Python is the Most Popular Language used for Machine Learning

- Easy & simple
- Efficient
- Diverse libraries & framework
- Versatile
- Vast community
- Portable & extensible
- Flexible
- Documentation

The infographic features a central illustration of a person sitting at a desk with a laptop, surrounded by icons representing search, code, and data. The background is a light green and yellow gradient.

Source: TechVidvan

Python libraries



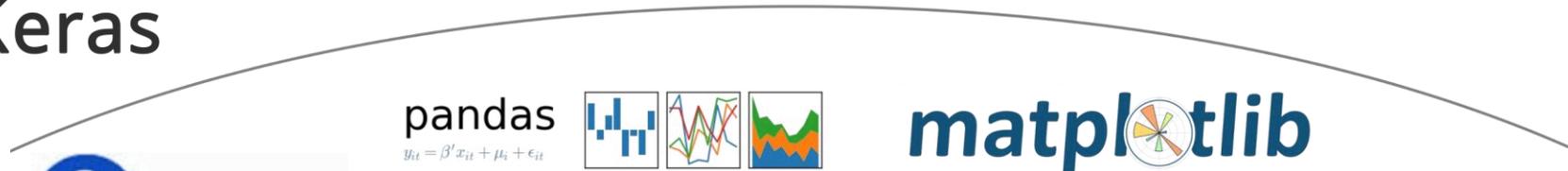
xarray



django



And many,
many more...



Write your first python program interactively

- IPython is a powerful interactive shell for Python programming.
- In 2014, Jupyter project (<http://jupyter.org/>) was created as a spin-off project from IPython. It is language-agnostic.
 - **JU**lia + **PY**thon + **R**
- Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations.
- Jupyter notebook system allows you to author content in Markdown to create a rich documentation with code and text.

<https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>

- We will be using **JupyterLab**, the next generation of the Jupyter Notebook.

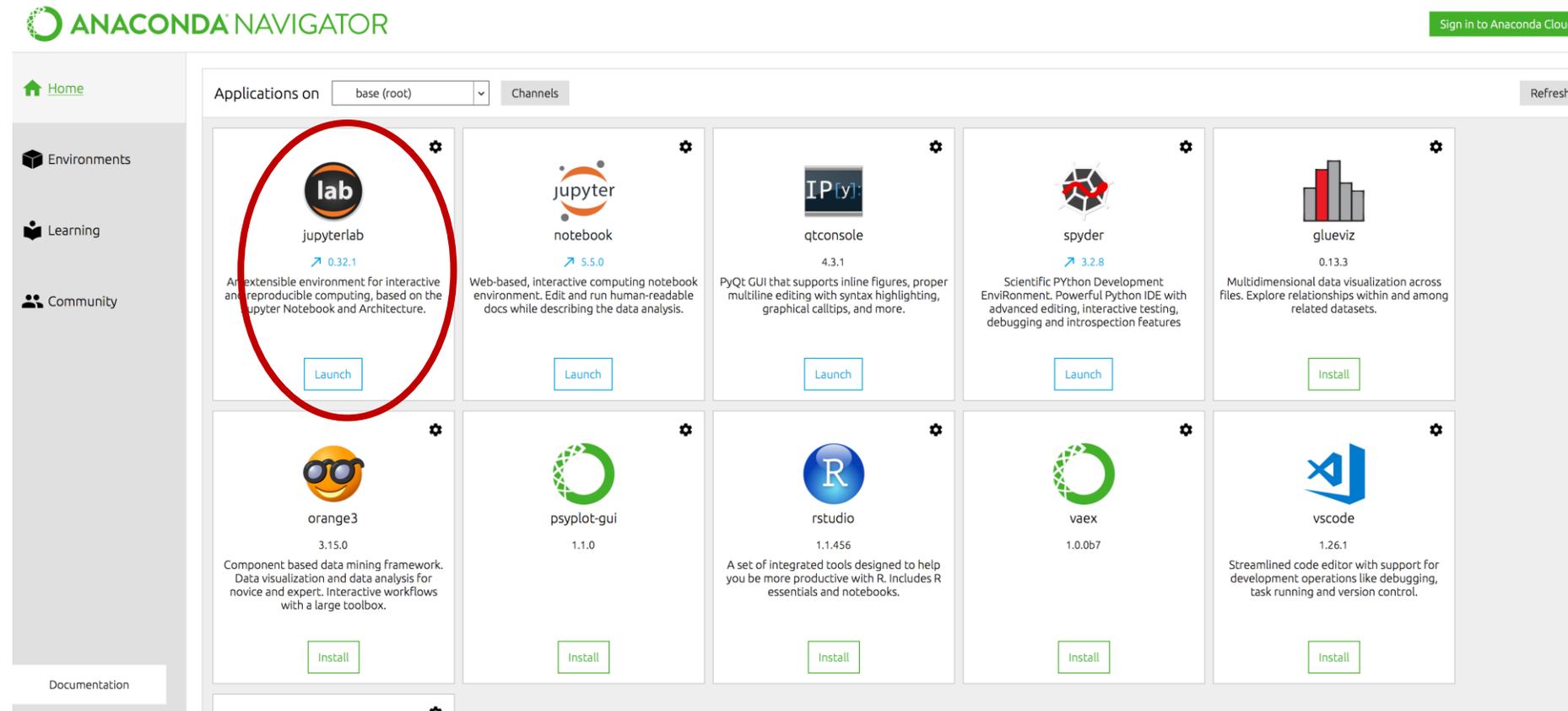
So, how do we get started with python?

- A **software distribution** is a pre-built and pre-configured collection of packages that can be installed and used on a system.
- A **package manager** is a tool that automates the process of installing, updating, and removing packages.
- **Anaconda** is a Python and R distribution platform, contains a package manager **Conda**.
- Many data science packages come preinstalled with Anaconda.
- You can download a free Anaconda Distribution at:
<https://www.anaconda.com/products/distribution>

What does interface look like?

Anaconda comes with a suite of graphical tools called Anaconda Navigator

- **JupyterLab** is an interactive programming environment (execute-explore vs. edit-compile-run): *experiment and evaluation*
- **Spyder** is an integrated development environment (IDE): *module development*



Conda seems ok, but I am used to pip. Is there a difference?

- **Pip** (Pip Installs Packages) is Python's officially-sanctioned package manager.
- Pip vs Conda:
 - Pip is a general-purpose manager for Python packages; conda is a language-agnostic cross-platform environment manager
 - Pip installs python packages within any environment; conda installs any package within conda environments.
 - For *our* use, pip and conda are mostly interchangeable.

Nice explanations of anaconda details:

<https://jakevdp.github.io/blog/2016/08/25/conda-myths-and-misconceptions/>

Virtual environments

- Python has different versions, packages have different versions. What if for two projects you need different versions of the same package?
- **Virtual environment** is an isolated environment that allows us to keep these dependencies in separate “sandboxes”.
- We can have many different environments, as they take up little space, each with separate package versions.

Let's get started

STEP 1: Install Anaconda

- Install Anaconda environment on your laptop
 - Download open-source Individual Edition Anaconda distribution for Python according to your OS (<https://www.anaconda.com/download/>)
 - It is free for solo practitioners, students, and researchers
 - Follow the installation instruction (<https://docs.anaconda.com/anaconda/install/>)
 - If your computer is short in storage, you may also consider Miniconda (no packages pre-installed)
 - To verify everything is working, open Anaconda prompt and write:

```
python --version
```

```
conda --version
```
 - Confirm that you have the latest version of conda:

```
conda update conda
```

(Update any package, if necessary, by typing y to proceed)

Step 2: Create conda virtual environment for this class

- To **create a virtual environment** named *ml2025*, type from Anaconda Prompt

```
conda create -n ml2025 python=3.12
```

when conda asks you to proceed, type y

- **Activate your environment:**

```
conda activate ml2025
```

the active environment---the one you are currently using---is shown in parentheses () or brackets [] at the beginning of your command prompt:

- **List all packages in environment**

Environment active:

```
conda list
```

Environment not active

```
conda list -n ml2025
```

- **Check if a specific package is installed:**

Environment active:

```
conda list <package name>
```

Environment not active

```
conda list -n ml2025 <package name>
```

-n, --name

Name of environment.

Step 2: Create conda virtual environment for this class

- Deactivate your environment

```
conda deactivate
```

- Delete an environment (no need to do now)

```
conda env remove -n ml2025
```

- List all environments

```
conda env list
```

Anaconda Navigator

File Help

ANACONDA.NAVIGATOR

Update Now Connect

Home

Environments

Learning

Community

Search Environments

base (root)

Anaconda3

ml2023

ml2025

prophet

python39

Installed Channels Update index...

Search Packages

Name	Description	Version
✓ _anaconda_depends	Simplifies package management and deployment of anaconda	2024.10
✓ aext-assistant	Anaconda extensions assistant library	4.1.0
✓ aext-assistant-server	Anaconda extensions assistant server	4.1.0
✓ aext-core	Anaconda extensions core library	4.1.0
✓ aext-core-server	Anaconda toolbox backend lib core server component	4.1.0
✓ aext-panels		4.1.0
✓ aext-panels-server		4.1.0
✓ aext-project-filebrowser-server		4.1.0
✓ aext-share-notebook		4.1.0
✓ aext-share-notebook-server		4.1.0
✓ aext-shared	Anaconda extensions shared library	4.1.0
✓ aext-toolbox		4.1.0
✓ aiobotocore	Async client for aws services using botocore and aiohttp	2.12.3
✓ aiohappyeyeballs		↗ 2.4.0
✓ aiohttp	Async http client/server framework (asyncio)	↗ 3.10.5
✓ aiortools	Asyncio version of the standard multiprocessing module	0.7.1

532 packages available

Create Clone Import Backup Remove

Your Voice Matters!
Take a survey to join a live Q&A with our CAIO/Co-founder, Peter Wang
Take Survey

Documentation
Anaconda Blog

X

Step 3: Install the packages we will use during this course

- Install a package

Environment active: `conda install <package_name>`

Environment not active `conda install -n ml2025 <package name>`

- Install multiple packages

```
conda install pandas numpy
```

- Upgrade a package

```
conda update <package name>
```

- Install a package with a specific version

```
conda install <package name> = <version number>
```

- Remove a package

```
conda remove <package name>
```

Some of the packages we will use during this course

- **pandas** data manipulation and analysis
- **numpy** mathematical functions
- **scikit-learn** machine learning (conda install conda-forge::scikit-learn)
Referred to as **sklearn** when importing, example: from **sklearn.metrics** import mean_squared_error
- **xgboost, lightgbm, catboost** gradient boosting
Xgboost Installed as: conda install -c conda-forge py-xgboost
lightgbm installed as: conda install -c conda-forge lightgbm
catboost installed as: as conda install -c conda-forge catboost
- **pytorch** and **torchvision** neural networks (conda install pytorch torchvision cpuonly -c pytorch)
- **shap** interpreting ML models
Installed as: conda install -c conda-forge shap
- **matplotlib, seaborn** data visualization
- **spacy, nltk** text analysis
conda install -c conda-forge spacy
conda install nltk
- **imbalanced learn** classification with imbalanced classes
conda install -c conda-forge imbalanced-learn
- **aequitas** Bias and Fairness Audit Toolkit
Installed as: pip install aequitas

Important note

- As each student may have their own hardware and software configuration, we cannot guarantee that it will be possible to install all the packages on all the configurations.
- If a problem is encountered:
 - Follow the recommended steps in the notebook or lecture notes
 - Google for similar errors (sometimes a package should be upgraded or downgraded to work within a specific setting)
 - Consider making a new environment to test installation of a new package before you start downgrading or upgrading already installed packages in a working environment
 - Ask TA or instructor for help
 - Use Google colab notebook, it will be accepted for homework and project submissions.

Step 4: Use the conda environment in your jupyter notebook

- Install ipykernel in *ml2025* environment (allows Jupyter to recognize the environment as a kernel)

```
conda install ipykernel
```

- Add the *ml2025* environment as a kernel for Jupyter

```
python -m ipykernel install --user --name=ml2025
```

- Deactivate the environment

```
conda deactivate
```

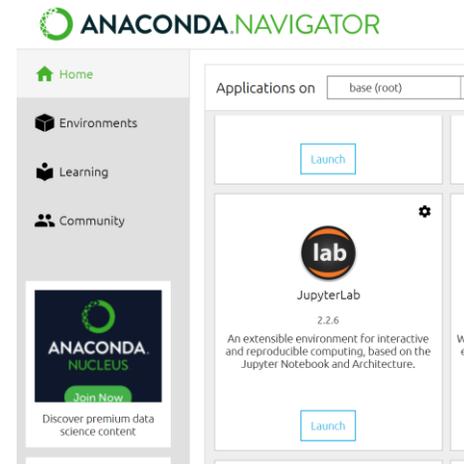
To list existing kernels

```
jupyter kernelspec list
```

Note: A notebook **kernel** is a “computational engine” that executes the code contained in a Notebook document

Step 4: Use the conda environment in your jupyter notebook

- From Anaconda Navigator Launch **JupyterLab** (not Jupyter Notebook!)



- Select from list of kernels the kernel *ml2025*

Select Kernel

Select kernel for: "Untitled.ipynb"

ml2025

Always start the preferred kernel

No Kernel

Select

Scikit-learn: Machine Learning in python

scikit-learn.org/stable/

scikit-learn Install User Guide API Examples Community More

1.6.1 (stable)

scikit-learn

Machine Learning in Python

Getting Started **Release Highlights for 1.6**

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: [Gradient boosting](#), [nearest neighbors](#), [random forest](#), [logistic regression](#), and [more...](#)

Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, stock prices.

Algorithms: [Gradient boosting](#), [nearest neighbors](#), [random forest](#), [ridge](#), and [more...](#)

Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, grouping experiment outcomes.

Algorithms: [k-Means](#), [HDBSCAN](#), [hierarchical clustering](#), and [more...](#)

Examples

Dimensionality reduction

Reducing the number of random variables to consider

Model selection

Comparing, validating and choosing parameters and

Preprocessing

Feature extraction and normalization

Good reference on coding how-to's

<https://jakevdp.github.io/PythonDataScienceHandbook/>

Python Data Science Handbook

Jake VanderPlas



Jake VanderPlas

[3. Data Manipulation with Pandas](#)

- [Introducing Pandas Objects](#)
- [Data Indexing and Selection](#)
- [Operating on Data in Pandas](#)
- [Handling Missing Data](#)
- [Hierarchical Indexing](#)
- [Combining Datasets: Concat and Append](#)
- [Combining Datasets: Merge and Join](#)
- [Aggregation and Grouping](#)
- [Pivot Tables](#)
- [Vectorized String Operations](#)
- [Working with Time Series](#)
- [High-Performance Pandas: eval\(\) and query\(\)](#)
- [Further Resources](#)

[5. Machine Learning](#)

- [What Is Machine Learning?](#)
- [Introducing Scikit-Learn](#)
- [Hyperparameters and Model Validation](#)
- [Feature Engineering](#)
- [In Depth: Naive Bayes Classification](#)
- [In Depth: Linear Regression](#)
- [In-Depth: Support Vector Machines](#)
- [In-Depth: Decision Trees and Random Forests](#)
- [In Depth: Principal Component Analysis](#)
- [In-Depth: Manifold Learning](#)
- [In Depth: k-Means Clustering](#)
- [In Depth: Gaussian Mixture Models](#)
- [In-Depth: Kernel Density Estimation](#)
- [Application: A Face Detection Pipeline](#)
- [Further Machine Learning Resources](#)

Good reference on coding how-to's

<https://vedraiyni.github.io/notes-1/ipynb/index.html>

Preprocessing Structured Data

- Convert Pandas Categorical Data For Scikit-Learn
- Delete Observations With Missing Values
- Deleting Missing Values
- Detecting Outliers
- Discretize Features
- Encoding Ordinal Categorical Features
- Handling Imbalanced Classes With Downsampling
- Handling Imbalanced Classes With Upsampling
- Handling Outliers
- Impute Missing Values With Means
- Imputing Missing Class Labels
- Imputing Missing Class Labels Using k-Nearest Neighbors
- Normalizing Observations
- One-Hot Encode Features With Multiple Labels
- One-Hot Encode Nominal Categorical Features
- Preprocessing Categorical Features
- Preprocessing Iris Data
- Rescale A Feature
- Standardize A Feature

Trees And Forests

- Outlier Detection With Isolation Forests
- Adaboost Classifier
- Decision Tree Classifier
- Decision Tree Regression
- Feature Importance
- Feature Selection Using Random Forest
- Handle Imbalanced Classes In Random Forest
- Random Forest Classifier
- Random Forest Classifier Example
- Random Forest Regression
- Select Important Features In Random Forest
- Titanic Competition With Random Forest
- Visualize A Decision Tree

Nearest Neighbors

- Identifying Best Value Of k
- K-Nearest Neighbors Classification
- Radius-Based Nearest Neighbor Classifier

Feature Engineering

- Dimensionality Reduction On Sparse Feature Matrix
- Dimensionality Reduction With Kernel PCA
- Dimensionality Reduction With PCA
- Feature Extraction With PCA
- Group Observations Using K-Means Clustering

Feature Selection

- ANOVA F-value For Feature Selection
- Chi-Squared For Feature Selection
- Drop Highly Correlated Features

Model Evaluation

- Accuracy
- Create Baseline Classification Model
- Create Baseline Regression Model
- Cross Validation Pipeline
- Cross Validation With Parameter Tuning Using Grid Search
- Cross-Validation
- Custom Performance Metric
- F1 Score
- Ge
- Ne
- Pl
- Pl
- Plot The Validation Curve
- Precision
- Recall
- Split Data Into Training And Test Sets

Model Selection

- Find Best Preprocessing Steps During Model Selection
- Hyperparameter Tuning Using Grid Search
- Hyperparameter Tuning Using Random Search
- Model Selection Using Grid Search
- Pipelines With Parameter Optimization

Linear Regression

O'REILLY



Python
Machine
Learning
Cookbook

PRACTICAL SOLUTIONS FROM PREPROCESSING TO DEEP LEARNING

Chris Albon



Welcome To Colaboratory

File Edit View Insert Runtime Tools



Table of contents



Getting started

Data science



Machine learning



More Resources

Featured examples



+ Section

Change runtime type

Runtime type

Python 3

Hardware accelerator



CPU



T4 GPU



A100 GPU



V100 GPU



TPU

Want access to premium GPUs? [Purchase additional compute units](#)

Using colab

- Google Colab (<https://colab.research.google.com/>) is an extension of Jupyter notebook that runs on the Google Cloud. This platform provides various different computing resources, such as CPUs, **GPUs** free of charge.
- Colab allows you to use and share Jupyter notebooks with others without having to download, install, or run anything.
- Google Colab has a ‘maximum lifetime’ limit of running notebooks that is 12 hours with the browser open, and the ‘Idle’ notebook instance is interrupted after 90 minutes.
- Colab notebooks can be shared with other users and opened by multiple users at a time. If one person makes a change, the others will be able to see the change after a short delay. However, if two people edit the document at the same time, one person’s changes must be discarded upon refreshing.
- <https://research.google.com/colaboratory/faq.html>