Data Analysis and Knowledge Discovery

Tools for Data Analysis

Jari Björne

University of Turku
Department of Information Technology

jari.bjorne@utu.fi

Outline

Section 1

Why the Python language

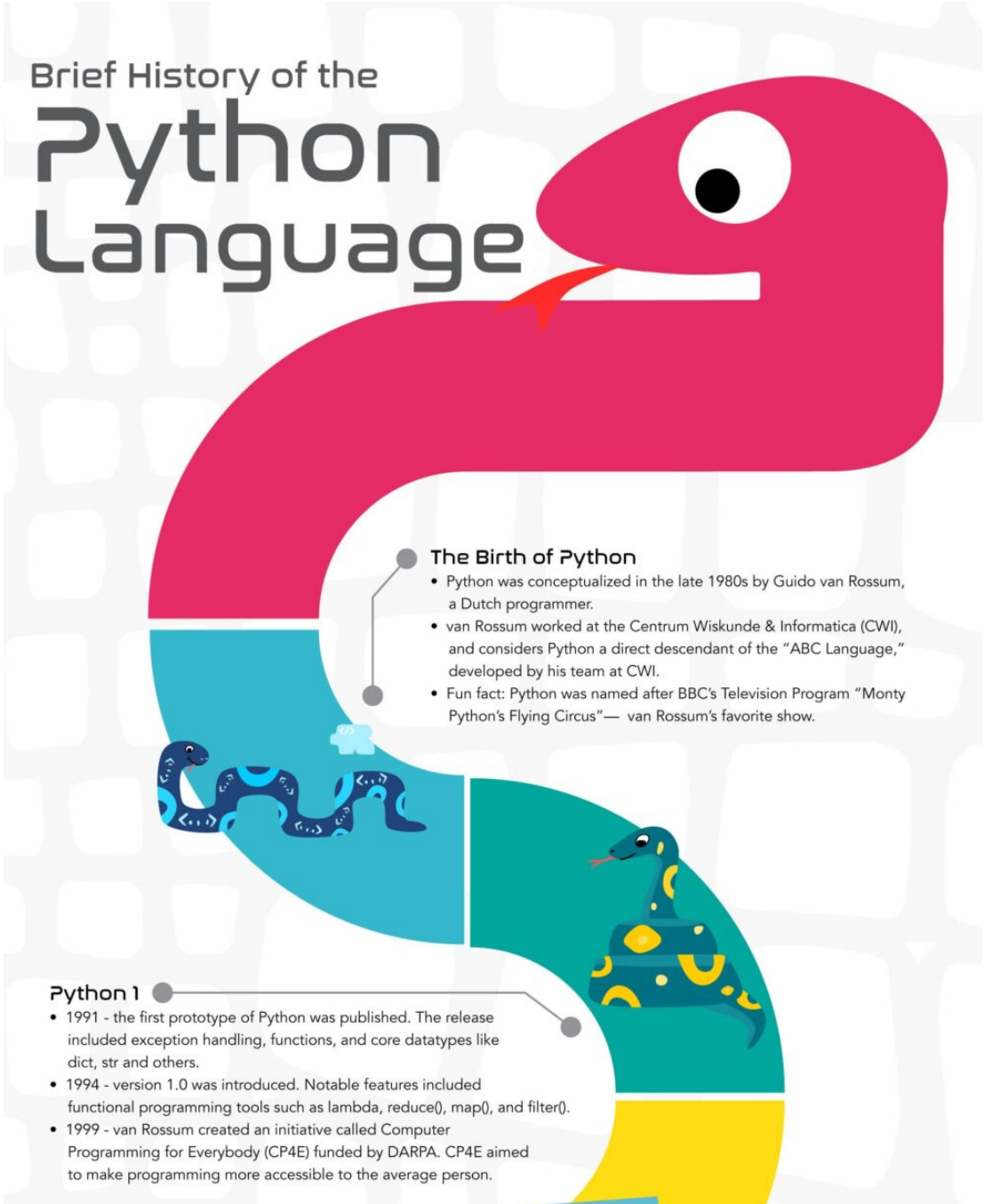
The Python language

- The Python language is used for the DAKD examples and exercises.
- Python has become one of the most common languages in data science.
- In other courses you will also use R etc.

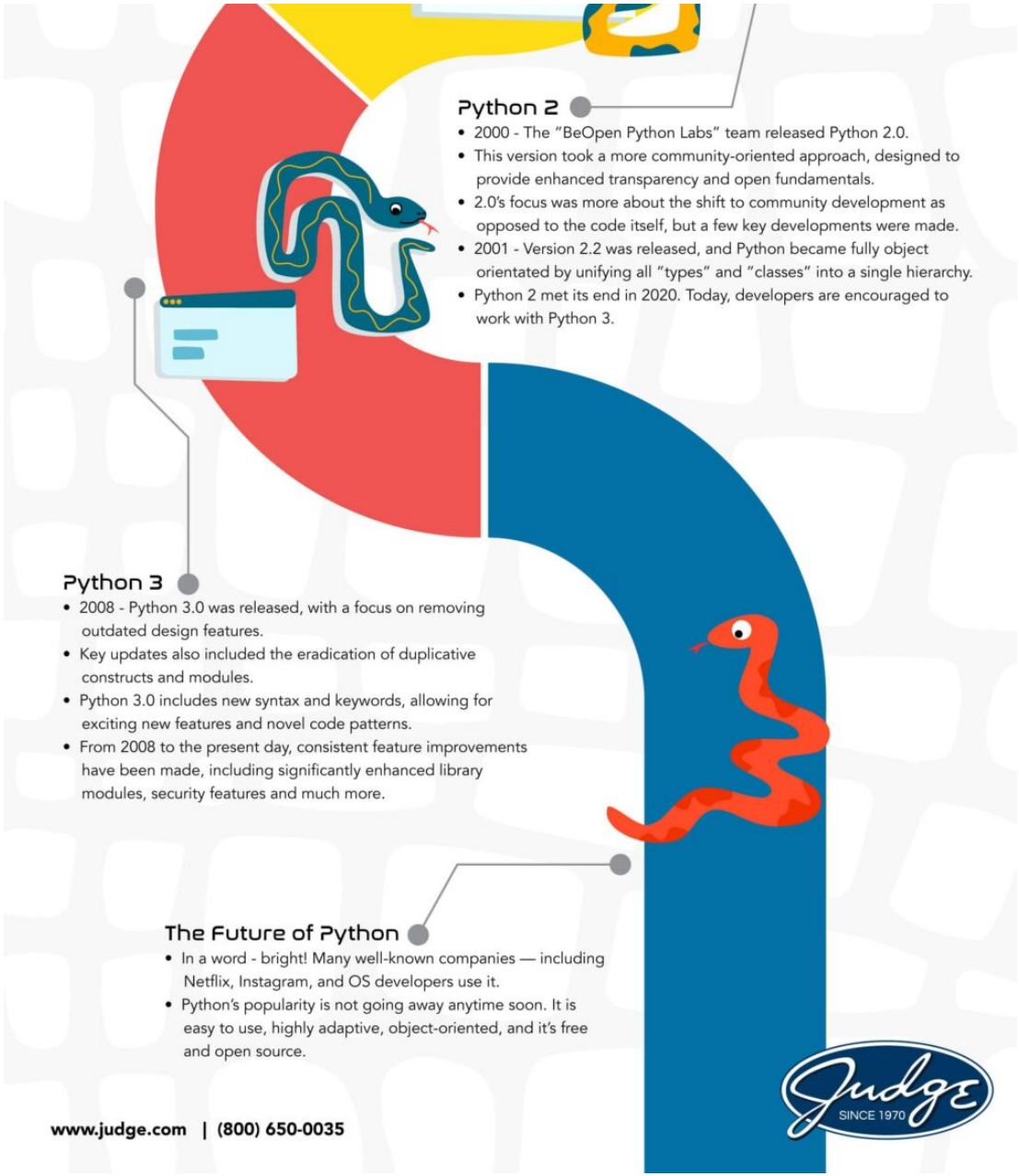
What is Python?

- A high-level, general-purpose programming language.
- The Python philosophy emphasizes code readability and ease of programming over speed.
- Python is a *dynamically typed* and *garbage collected* language.
- It can be used for both *procedural* and *object-oriented* programming.
- It has a large and comprehensive standard library ("batteries included")

A Brief History of the Python Language



A Brief History of the Python Language



Why is Python so Popular in Data Science?

- Easy and fast to write $\rightarrow$ good choice for rapid prototyping and experimentation
- Easier to use for statisticians, bioinformaticians, physicians etc. scientists not trained in programming
- An academically oriented Python open source community developed libraries relevant for data science

Other Languages in Data Science

- Matlab (originally from 1970s) is a commercial computational environment for mathematics
- Developed for statistics, the R language fulfils a similar role as an open source project
- Code written in languages like Fortran can still be present in the underlying libraries of R or Python

Section 2

The Python Interpreter

Python is an Interpreted Language

- No separate compilation step is required.
- Interpreted languages are commonly slow, especially on numeric operations.
- Fast to write, slow to run $\rightarrow$ Good for prototyping and research, limited in production environments

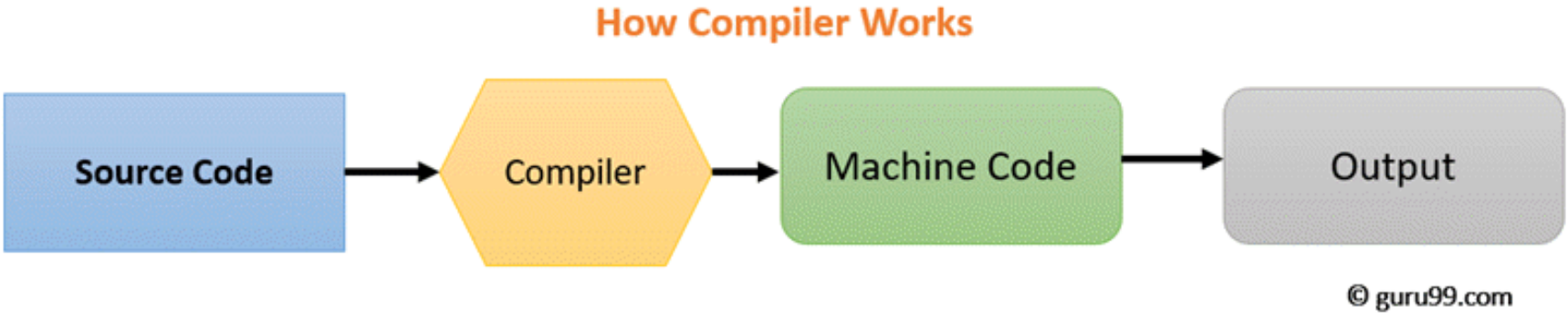


Image: John Smith, Compiler vs Interpreter
`https://www.guru99.com/difference-compiler-vs-interpreter.html`

The Python Interpreter

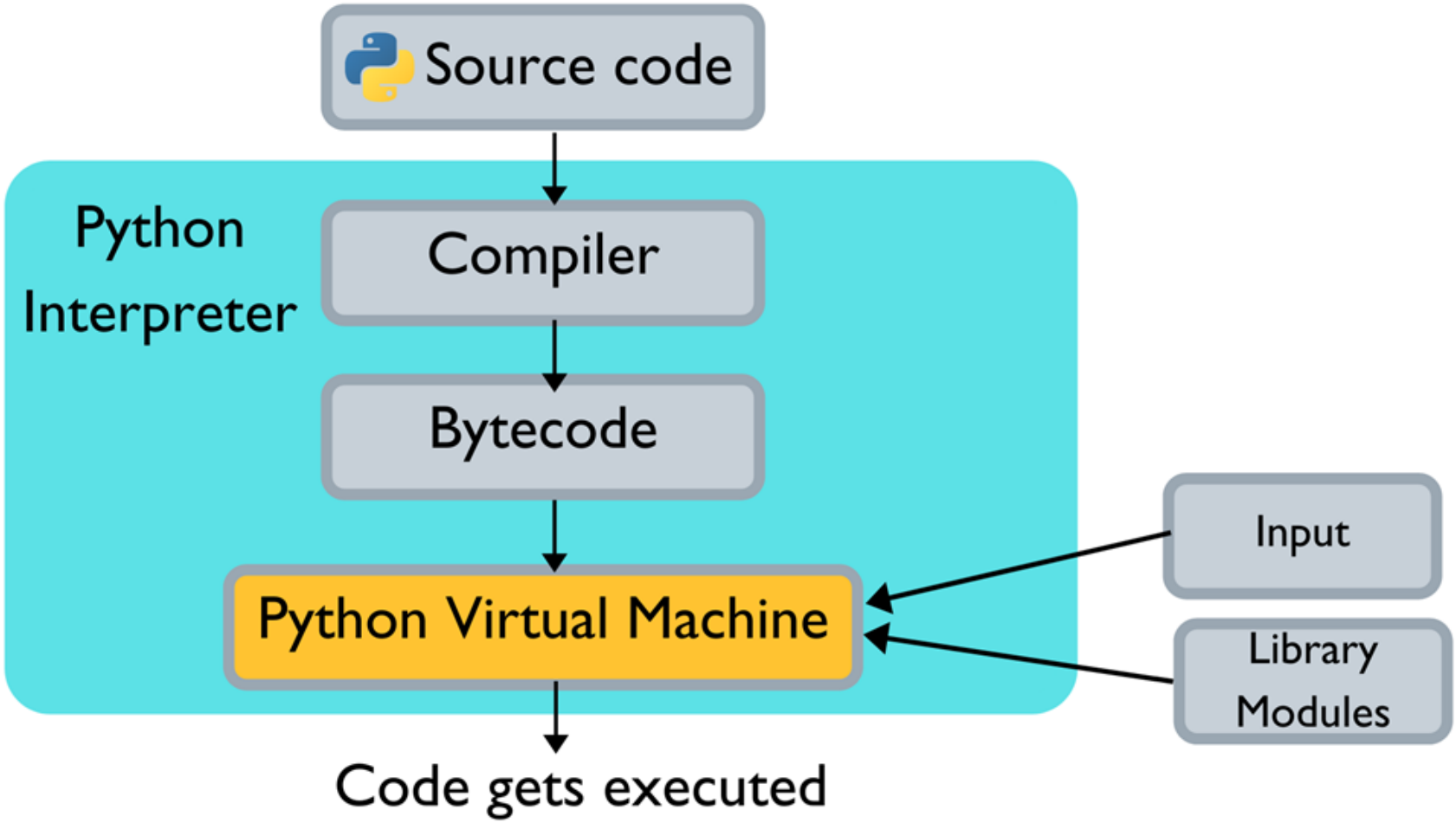


Image: Shalini Ravi, How Does Python Code Run
https://www.c-sharpcorner.com/article/why-learn-python-an-introduction-to-python/

Extending Python with Other Languages

- Through the CFFI (C Foreign Function Interface) Python can use native libraries written in C.
- Libraries written in C code are faster because they are *strongly typed*, *compiled* and use *manual memory management*
- Lots of existing code is in C libraries.
- While calculations inside a CFFI extension can be very fast, trasferring data between the Python program and the extension is slower.

Python CFFI

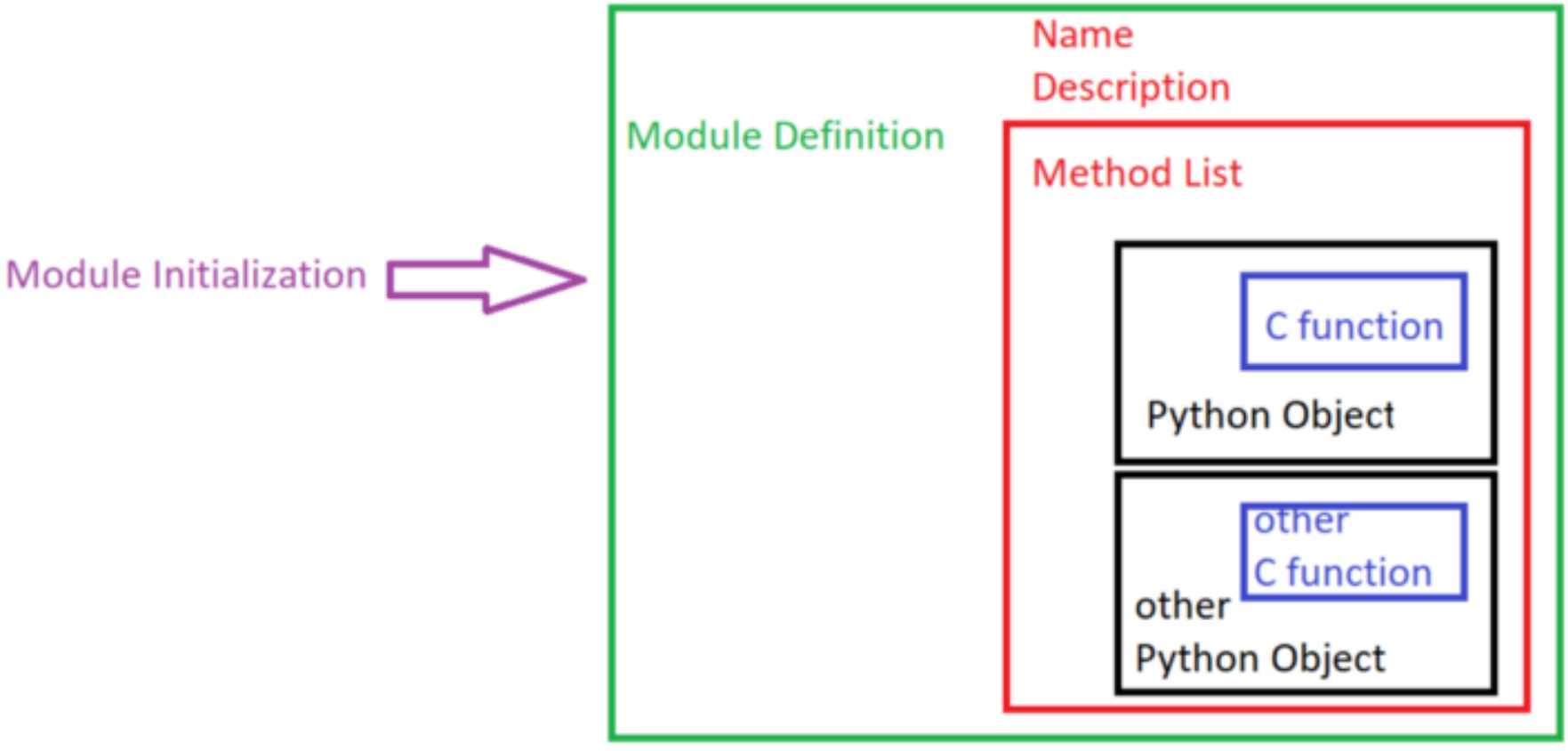


Image: Mike Huls, Write Your Own C-extension to Speed Up Python by 100x
https://towardsdatascience.com/write-your-own-c-extension-to-speed-up-python-x100-626bb9d166e7

Section 3

Python Libraries for Data Science

Numpy

- To overcome the limitations of Python, fast libraries for numerical array operations were developed
- Written in C, C++, Fortran etc, called by Python
- The less traffic there is between the Python and C etc. layers, the faster the program will be $\rightarrow$ algorithms defined for *arrays*, not for individual scalar variables
- The *n*-dimensional array can be seen as a vector or a matrix, depending on its shape.

The Data Matrix

Data matrix with m instances (rows) and n attributes (columns)

$$X = \begin{bmatrix} x_{1*} \\ \vdots \\ x_{m*} \end{bmatrix} = \begin{bmatrix} x_{*1}, ..., x_{*n} \end{bmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1n} \\ x_{21} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{pmatrix}$$

.

- Elements: x_{ij} refers to the j:th attribute of i:th instance
- Row vectors: x_{i*} corresponds to i:th instance
- Column vectors: x_{*j} corresponds to j:th attribute

From Linear Algebra to Numpy Programming

- Linear algebra equations translate directly to Python numpy (or Matlab) code
- A common technical issue: $i \rightarrow i - 1$ (Python indexing starts from 0, not 1)
- With some time and debugging it is possible to implement many data science algorithms with a few lines of Numpy code

Numpy Cheatsheet

math notation	numpy operation	explanation
$a_{i,j}$	`A[i-1,j-1]`	select element of A
a_{i*}	`A[i-1,:]`	select row from A
a_{*j}	`A[:,j-i]`	select column from A
A^T	`A.T`	transpose of A
AB	`A@B`	matrix multiplication
A^{-1}	`np.linalg.inv(A)`	matrix inversion
$A = V\Lambda V^T$	`np.linalg.eig(A)`	eigendecomposition
$Ax = b$	`np.linalg.solve(A,b)`	solve for x
μ (vector)	`np.mean(A, rows=0/1)`	row/col means
C (matrix)	`np.cov(A, rowvar=1/0)`	row/col covariances

When Working With Numpy

- Think in vectors and matrices, not in functions and loops.
- Be consistent in writing high performance code.
- Your data processing pipeline will be as slow as its slowest part.

Pandas

- A library for loading, preprocessing and analyzing datasets (**pan**el **da**ta)
- Based on the concept of a dataframe (a data matrix with typed columns)
- Pandas can import data from many formats, such as CSV, Excel or SQL tables.
- Interface relies heavily on operator overloading

An Example of Pandas

```python
# Import Pandas
import pandas as pd

# Load movies csv data into a Pandas DataFrame
metadata = pd.read_csv('movies_metadata.csv', low_memory=False)

# Calculate mean of vote average column
C = metadata['vote_average'].mean()

# Calculate the minimum number of votes required to be in the chart, m
m = metadata['vote_count'].quantile(0.90)

# Filter out all qualified movies into a new DataFrame
q_movies = metadata.copy().loc[metadata['vote_count'] >= m]

# Function that computes the weighted rating of each movie
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    # Calculation based on the IMDB formula
    return (v/(v+m) * R) + (m/(m+v) * C)

# Define a new feature 'score' and calculate its value with `weighted_rating()`
q_movies['score'] = q_movies.apply(weighted_rating, axis=1)

#Sort movies based on score calculated above
q_movies = q_movies.sort_values('score', ascending=False)

#Print the top 5 movies
q_movies[['title', 'vote_count', 'vote_average', 'score']].head(5)
```

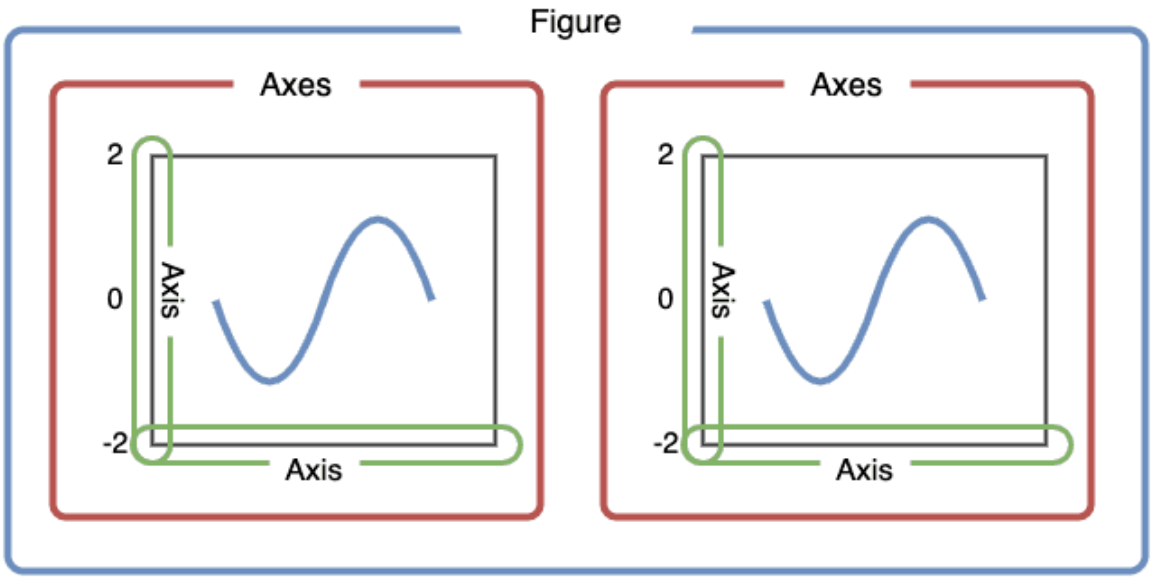
	title	vote_count	vote_average	score
314	The Shawshank Redemption	8358.0	8.5	8.445869
834	The Godfather	6024.0	8.5	8.425439
10309	Dilwale Dulhania Le Jayenge	661.0	9.1	8.421453
12481	The Dark Knight	12269.0	8.3	8.265477
2843	Fight Club	9678.0	8.3	8.256385

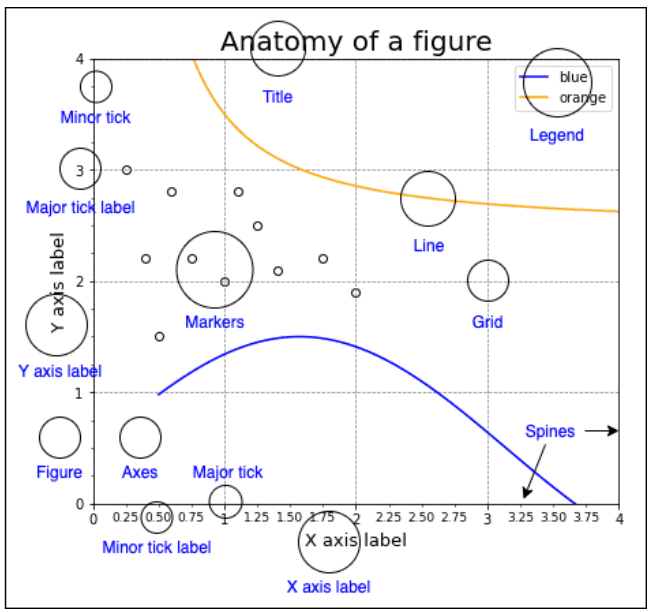
Source: https://www.coursereport.com/blog/3-real-world-examples-of-pandas-with-lighthouse-labs

Matplotlib

- Matplotlib is the most popular Python library for plotting graphs.
- Like many data science applications, it was inspired by Matlab.
- Matplotlib has two interfaces
  - The primary object oriented *pyplot* interface (recommended)
  - The Matlab style *pylab* state machine interface

Elements of a Matplotlib Figure

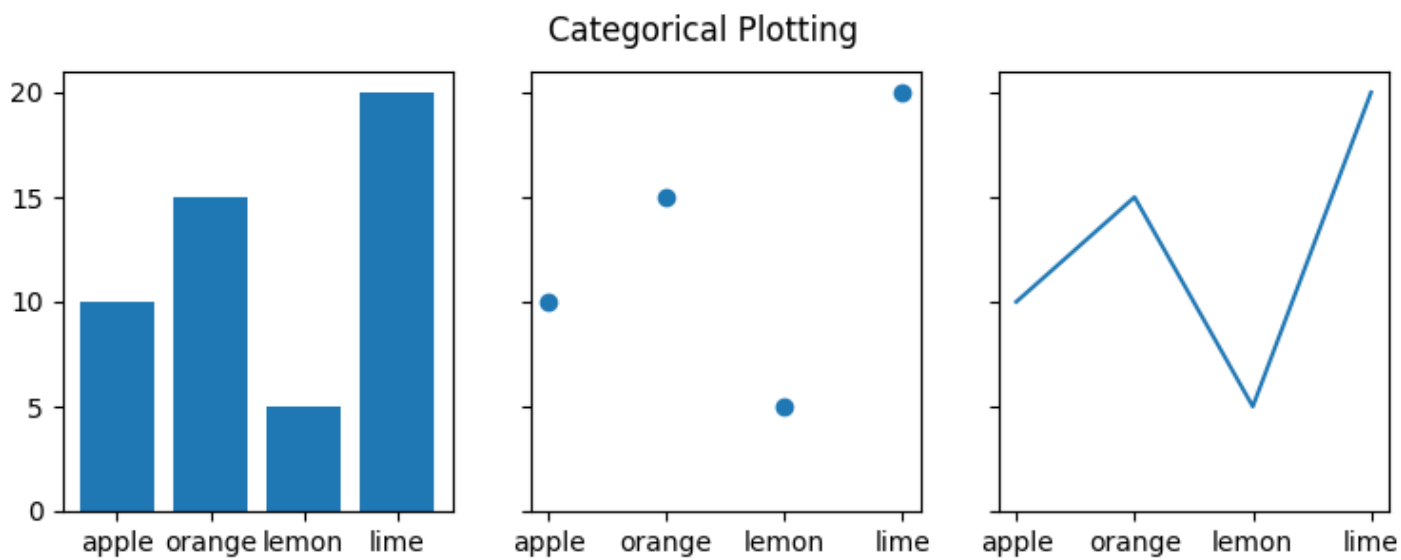




Images: Arthur Yanagisava (`https://www.skytowner.com/explore/getting_started_with_matplotlib`)

A Matplotlib Example: Categorical Plotting

```
import matplotlib.pyplot as plt

data = {'apple': 10, 'orange': 15, 'lemon': 5, 'lime': 20}
names = list(data.keys())
values = list(data.values())

fig, axs = plt.subplots(1, 3, figsize=(9, 3), sharey=True)
axs[0].bar(names, values)
axs[1].scatter(names, values)
axs[2].plot(names, values)
fig.suptitle('Categorical Plotting')
```



Source: https://matplotlib.org/stable/gallery/lines_bars_and_markers/categorical_variables.html

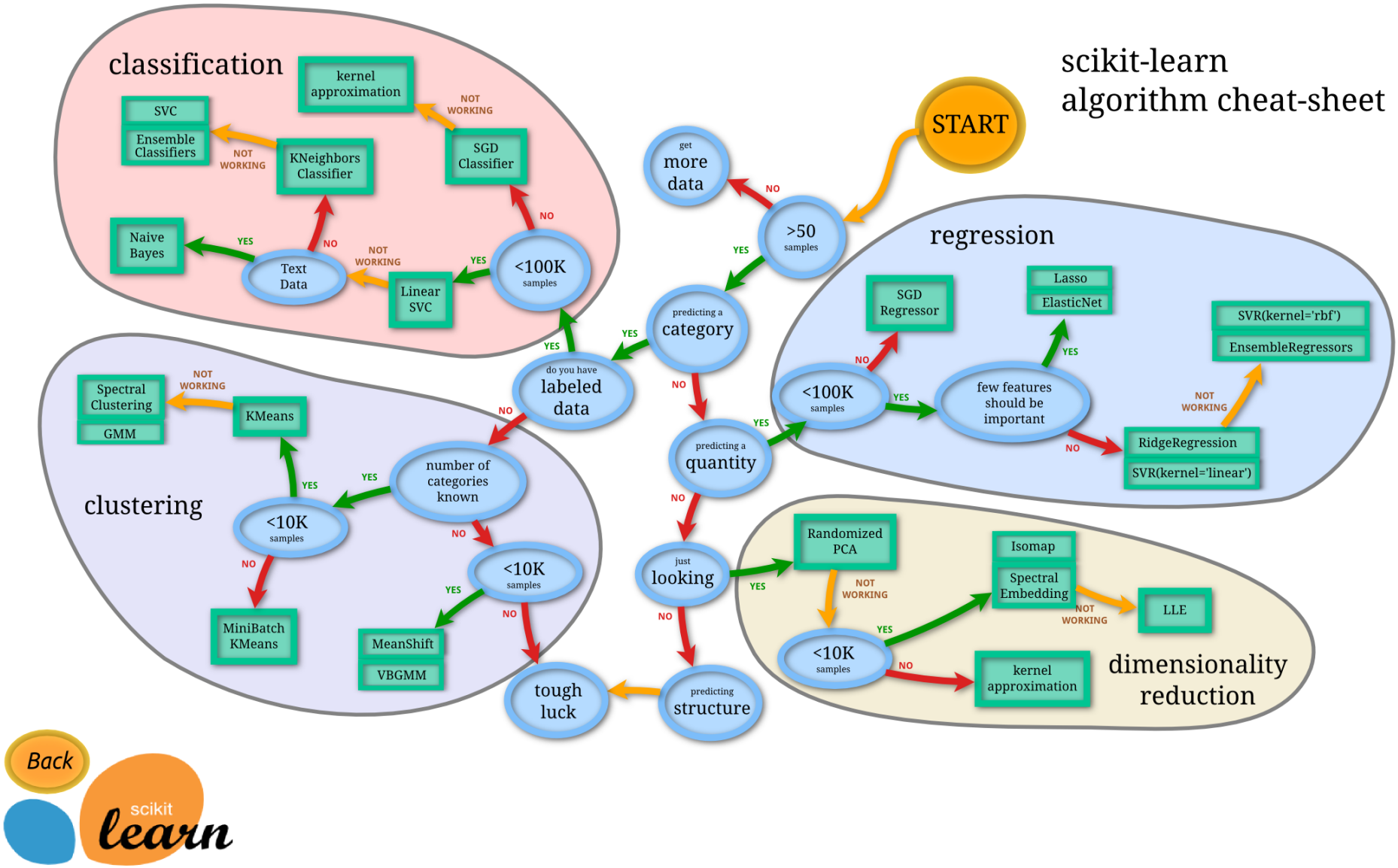
When Using Matplotlib

- Use the right plot for the right data. Make sure your lines, markers and colors are clear and relevant for the task.
- Add titles, text and a legend to make your figure understandable.
- Think about the scaling and limits of the figure $\rightarrow$ don't mislead by presentation.

Scikit-learn

- Widely used Python library for machine learning.
- Provides a common *estimator* interface through which different classifiers etc. can be used.
- Contains implementations for dozens of classifiers and machine learning algorithms
- Utilizes and builds on Numpy, Maplotlib, Scipy and other Python data science libraries.

Scikit-learn



A Scikit-learn Example

```
# Sample Decision Tree Classifier
from sklearn import datasets
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier
# load the iris datasets
dataset = datasets.load_iris()
# fit a CART model to the data
model = DecisionTreeClassifier()
model.fit(dataset.data, dataset.target)
print(model)
# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)
# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            presort=False, random_state=None, splitter='best')
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        50
          1       1.00      1.00      1.00        50
          2       1.00      1.00      1.00        50

avg / total       1.00      1.00      1.00       150

[[50  0  0]
 [ 0 50  0]
 [ 0  0 50]]
```

Source: https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/

Scipy

- Scientific Python, a library for scientific computing.
- Older, most features available in scikit-learn.
- Some algorithms (like dendrogram plotting) only in Scipy.

Section 4

Jupyter Notebook

What is it and why is it used in this course?

- A file format for intermixed writing and display of program code and explanatory text.
- Notebook files can be viewed as HTML pages where the embedded code blocks are run by a background Jupyter server.
- Jupyter Notebook is used as the format for this course's exercises.

A Jupyter Notebook in the Browser

jupyter IntroToJupyterPython (autosaved) Logout

File Edit View Insert Cell Kernel Help Not Trusted Python 2

Code

There are many shortcuts available. Some of these include:

Basic navigation: up-arrow, down-arrow, enter, shift-enter, up/k, down/j Saving the notebook: s Cell types: y (code), m (markdown), 1-6 (heading level)
Add cells: a (cell above), b (cell below) Cell editing: x (delete), c(copy)

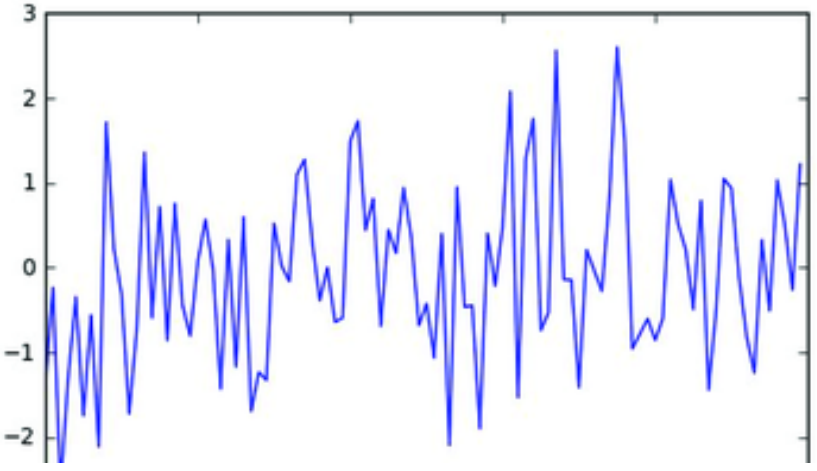
For a full list, see Help/Keyboard shortcuts on the top Menu.

Now let's try some examples! **To run: double-click in the code cell below, then press Shift+Enter:**

In [1]:
```
# This is a code cell. Click on it and press Shift+Enter. While it's executing, the prompt will turn into In [*]:

%pylab inline
print "Python is easy to learn!"

plot(randn(100))
```

```
Populating the interactive namespace from numpy and matplotlib
Python is easy to learn!
```

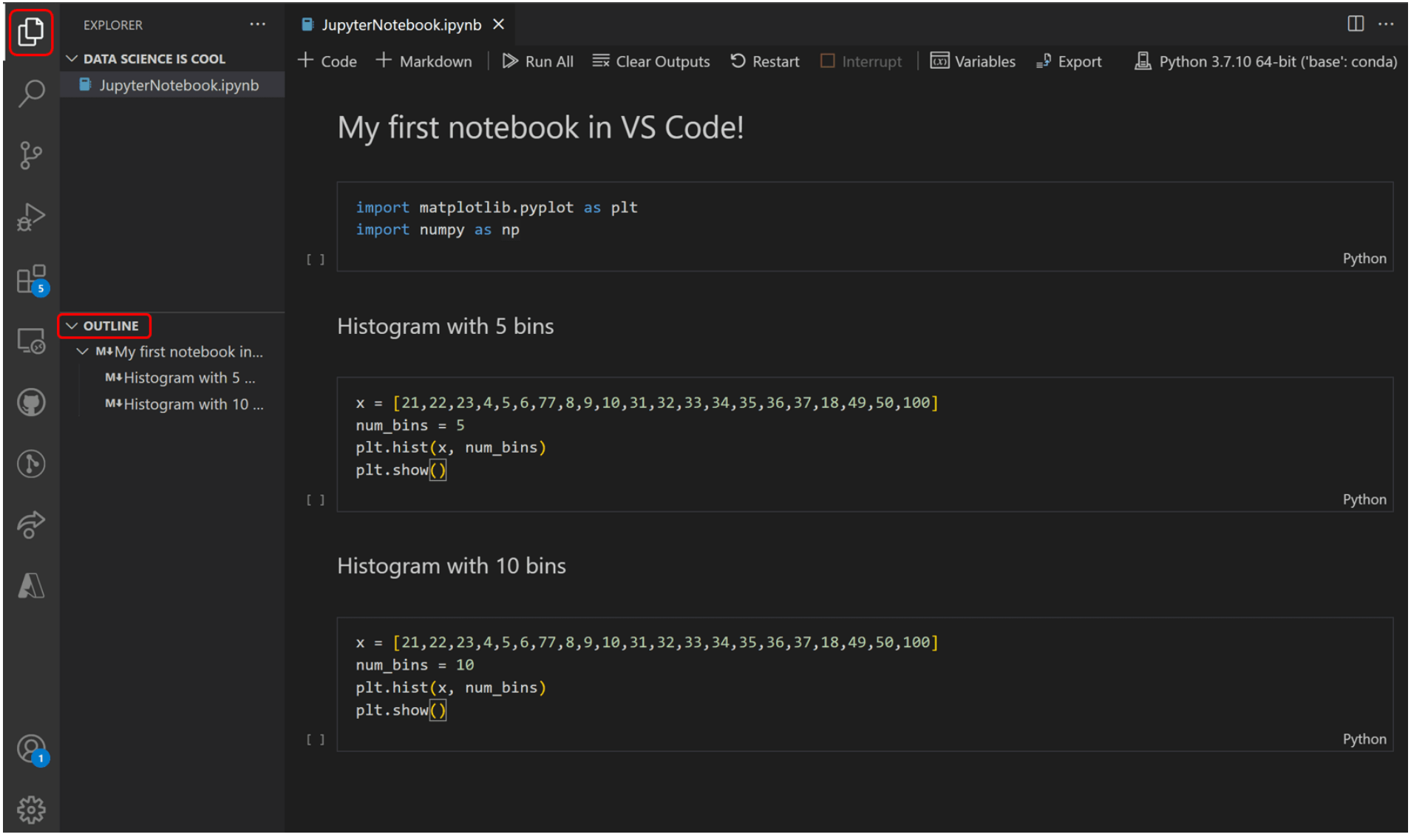
Out[1]:
```
[<matplotlib.lines.Line2D at 0x49a0d68>]
```



How to run notebooks

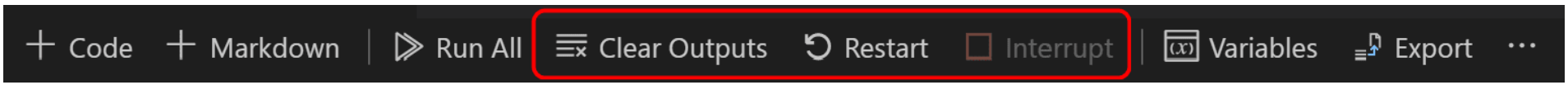
- Jupyter notebooks run on a local server. This server can be interacted with through a web browser interface.
- The easiest way to use Jupyter is through the free VSCode editor, or online at Google Colab.
- It can get a bit tricky to keep track of what is and isn't in memory, but you can always reset the notebook state to clear variables left over from running code elements.

A Jupyter Notebook in Visual Studio Code



What is the notebook state: Where do my variables go?

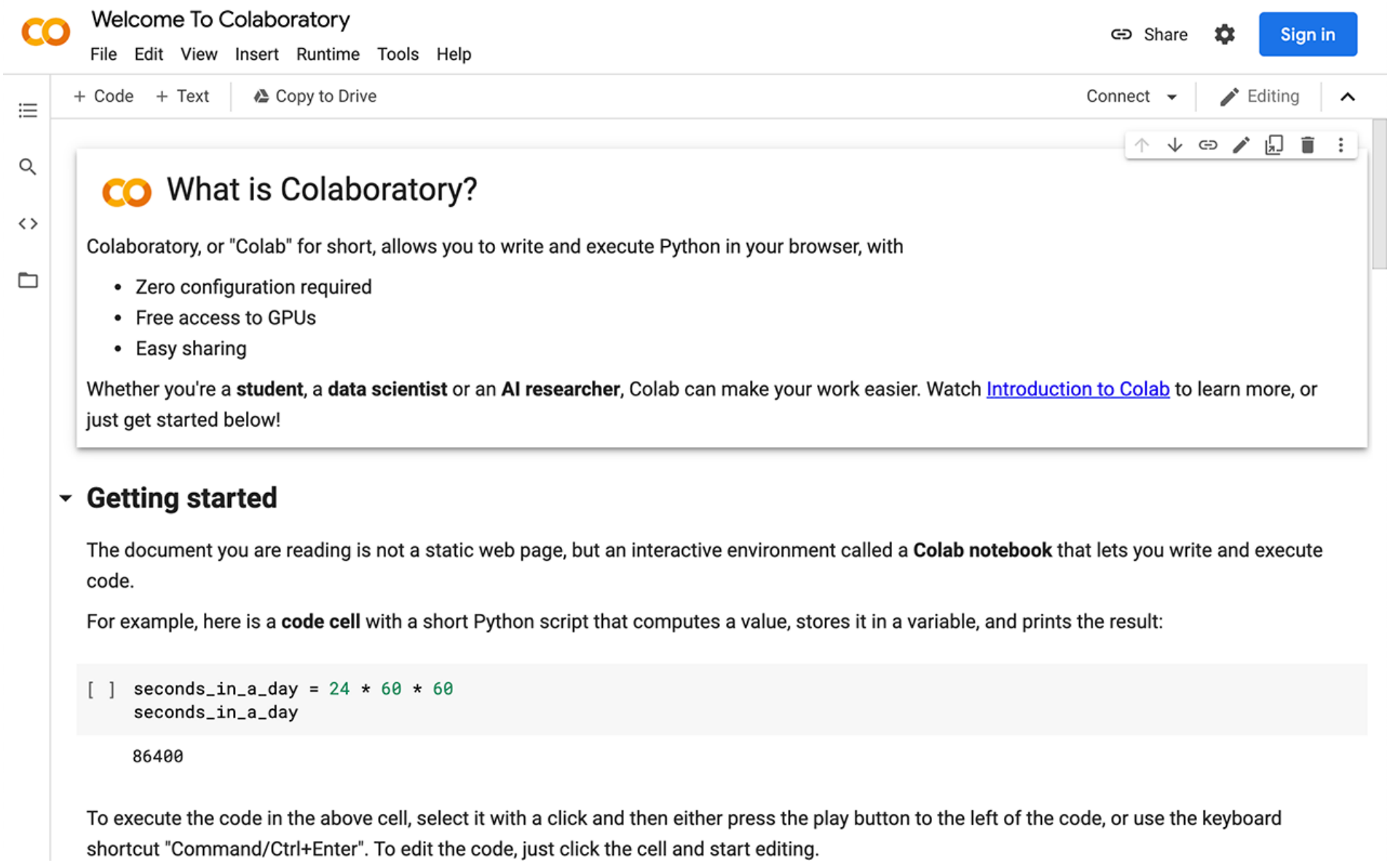
- When you run a Python program from the command line (`python myprogram.py`), it starts usually from a main function, this branches out into other functions, and finally the program ends. The program's variables exist in memory only while it runs.
- A Jupyter notebook has a persistent state: When you run a code element, the variables it defines stay in memory as long as the notebook is open and the associated Jupyter server is running.
- It can get a bit tricky to keep track of what is and isn't in memory, but you can always reset the notebook state to clear variables left over from running code elements.



Google Colab - Jupyter in the cloud

- Google Colab is Google's free cloud environment for running Jupyter Notebooks.
- Can be used for the exercises of this course.
- One of the few ways to get free GPU processing time for deep learning.

Google Colab - Jupyter in the cloud



Google Colab GPU:s

- Google Colab provides an NVidia Tesla K80 GPU which can be used with Python deep learning libraries.
- GPU:s are controlled with the CUDA language, but usually through a high-level library.
- GPU:s can be very expensive.

Section 5

Parallel Programming with Python

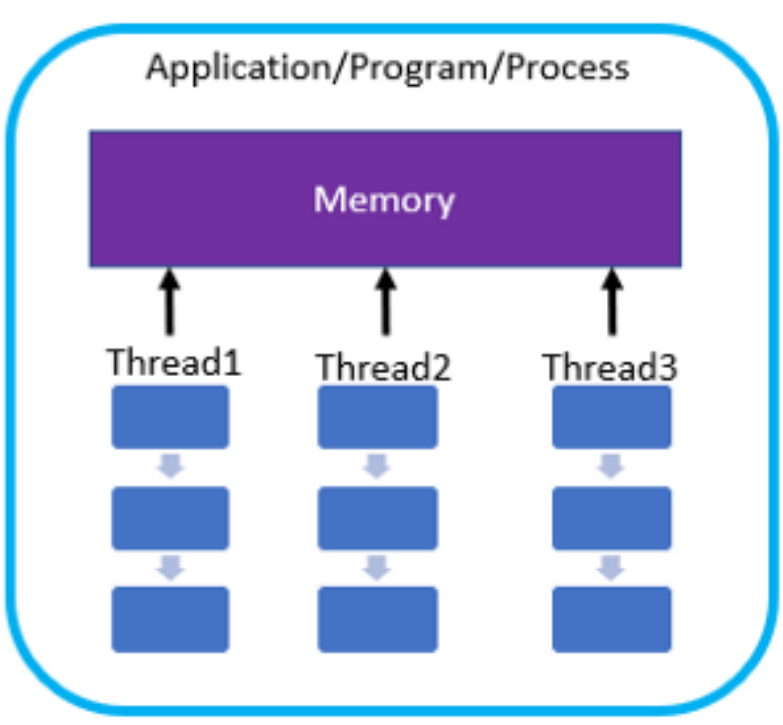
Parallel Programming with Python

- Python has problems with parallel processing due to its single-threaded architecture (GIL, Global Interpreter Lock)
- Additional libraries have been developed for parallelism through multiple processes, such as *multiprocessing* and *joblib* (included with scikit-learn)
- In cluster environments Python programs are often parallelized through the cluster environment's own tools such as SLURM

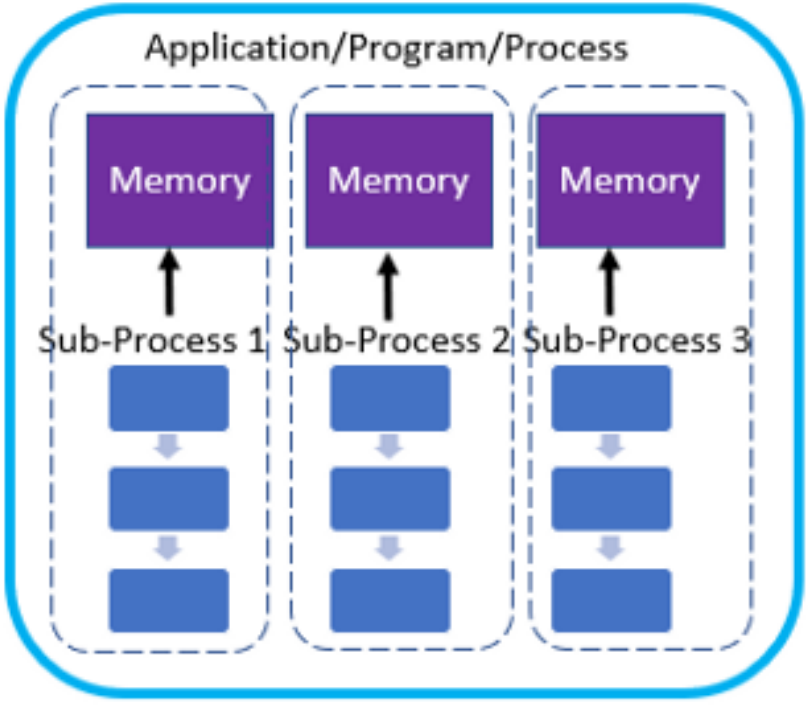
Multithreading vs. Multiprocessing

- Multithreading
  - An operating system mechanism for parallelism.
  - Threads can communicate and share memory.
  - Threads run on the same CPU (but can utilize multiple cores)
- Multiprocessing
  - Several independent processes (programs) are started by the main application process.
  - Subprocesses cannot share memory and usually don't communicate with each other.
  - Subprocesses can easily run on different CPU:s in a cluster environment
- Multiprocessing is often enough for data science
  - Many tasks in data science are "embarrassingly parallel".
  - There is just too much data to process sequentially.
  - Dividing the data into subsets and using cluster nodes to process each subset can massively reduce the real time needed.
  - No communication is needed between the cluster nodes.

Multithreading vs. Multiprocessing



Multi-Threading

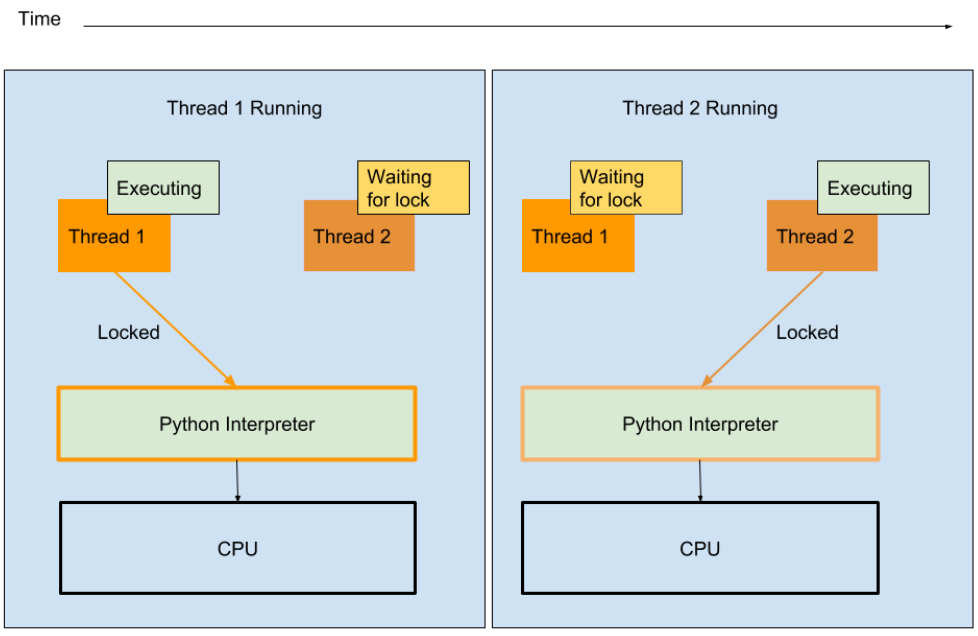


Multi-processing

Source: Renu Khandelwal, Multithreading and Multiprocessing in Python (`https://levelup.gitconnected.com/multi-threading-and-multiprocessing-in-python-3d5662f4a528`)

The Python Global Interpreter Lock

- Python has a thread library (module `threading`).
- The threads have normal mutexes (locks) and other thread synchronization methods.
- However, the Python interpreter will only one run thread at a time due to the Global Interpreter Lock (GIL).



Source: Vik Sunkavalli, The Python Global Interpreter Lock (`https://www.fulcircle.io/posts/the-python-gil/`)

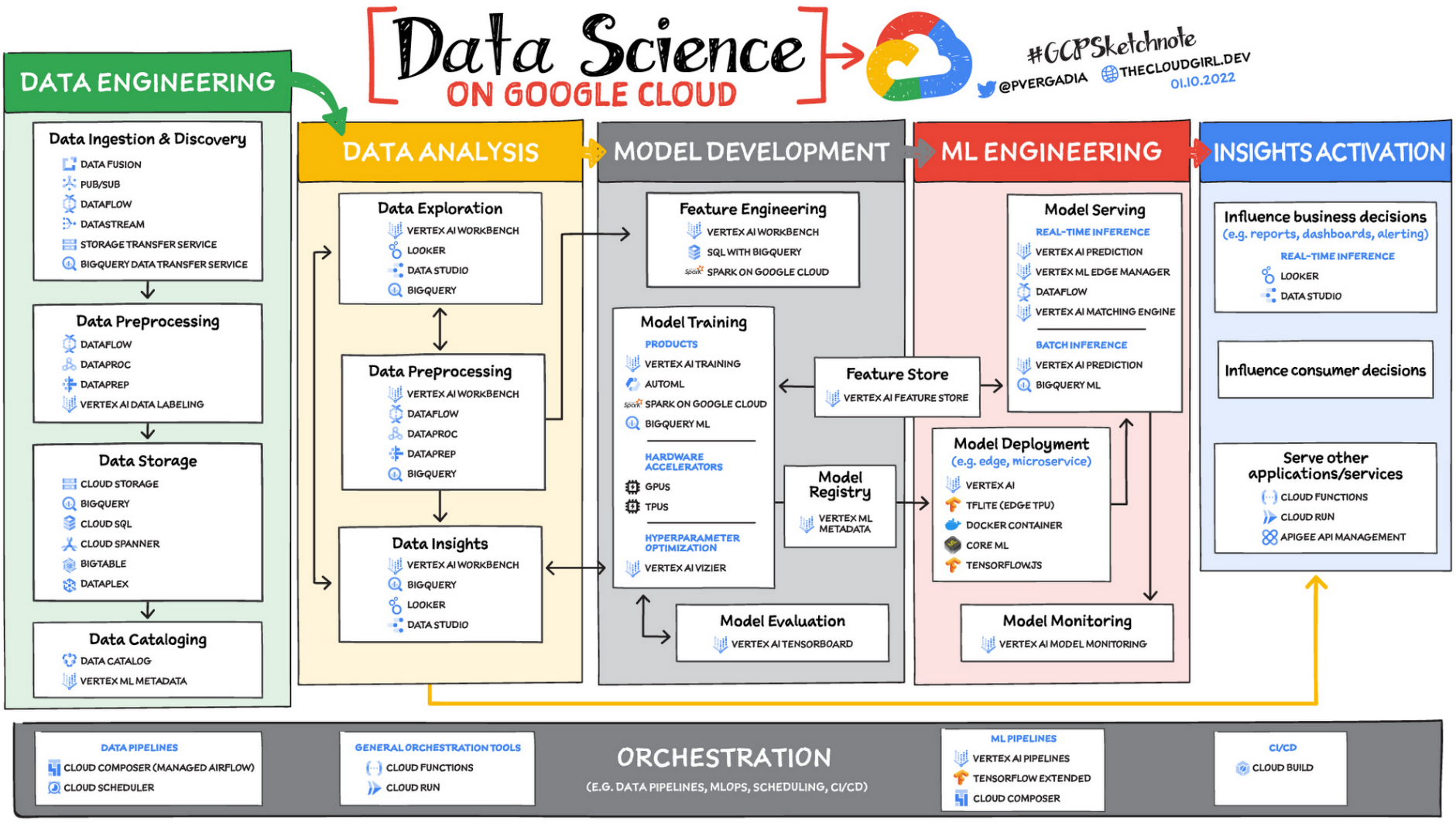
Section 6

Python in Cloud Computing and Deep Learning

Python in the Cloud: Google, AWS etc.

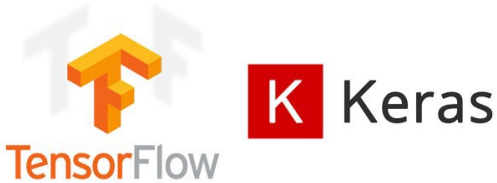
- Related to parallel processing, Python has today a role also in cloud computing
- Google Compute Engine can run Python programs
- Most cloud providers have some support for Python

Google Cloud for Data Science



Python in Deep Learning: Tensorflow, Keras and PyTorch

- The primary software libraries for deep learning are open source, often with Python interfaces
- The current leading implementations are Tensorflow (Google) and PyTorch (Facebook)
- Keras (included with Tensorflow 2.0) is a good choice for beginners

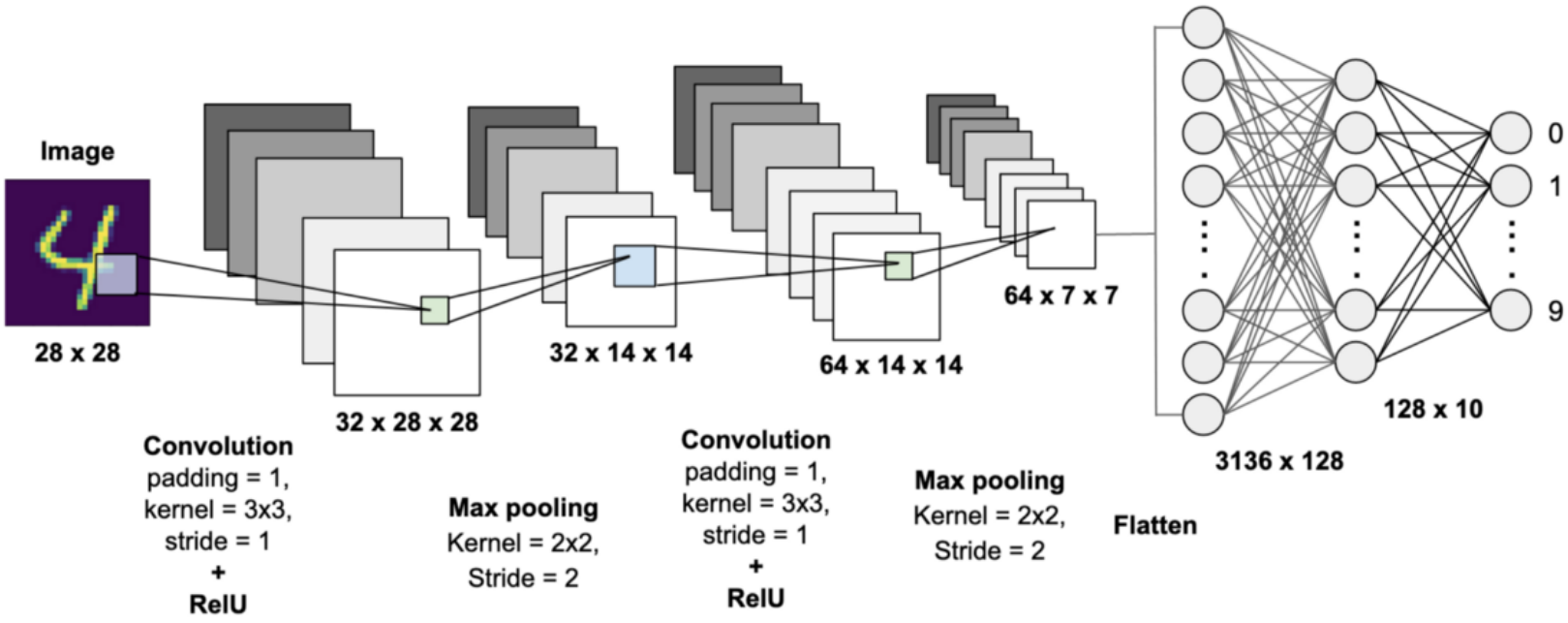


Defining a Neural Network in Keras

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Source: https://blog.tensorflow.org/2018/12/standardizing-on-keras-guidance.html

The MNIST Handwritten Digit Recognition Task



Source: `https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d`