LA TROBE UNIVERSITY

Building AI – Module 2

Tools for Building AI
24 May 2022

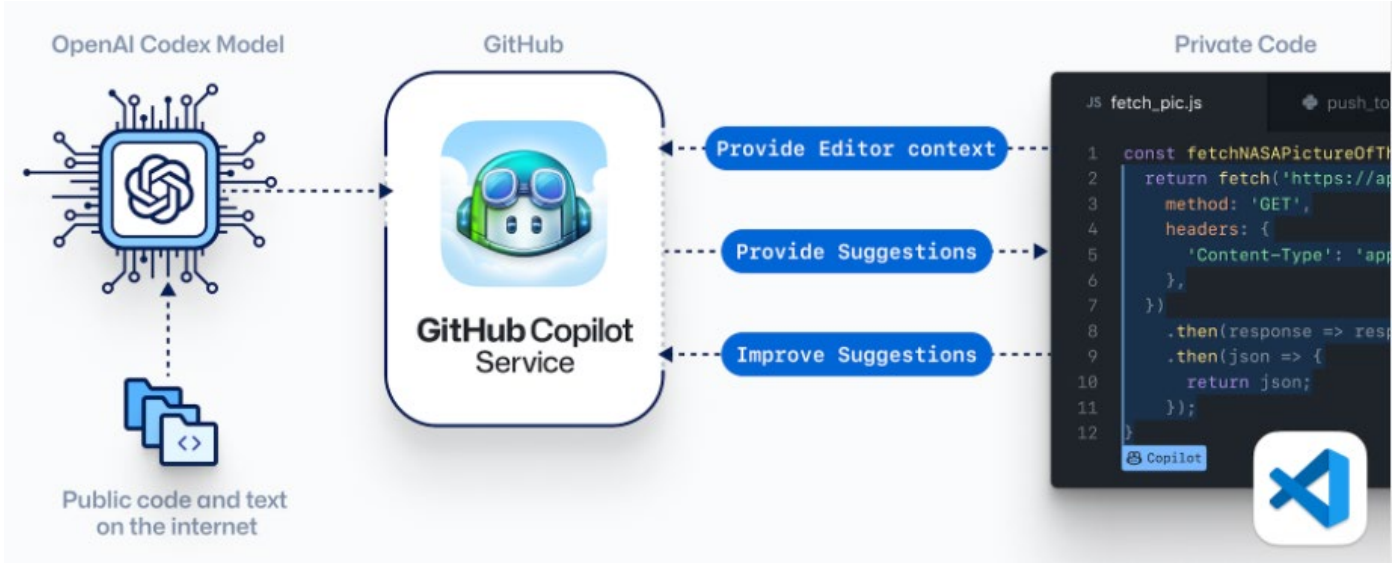
Optus U

Agenda

1. Tools for Building AI
2. Python Programming
3. Jupyter Notebooks
4. Google Colab

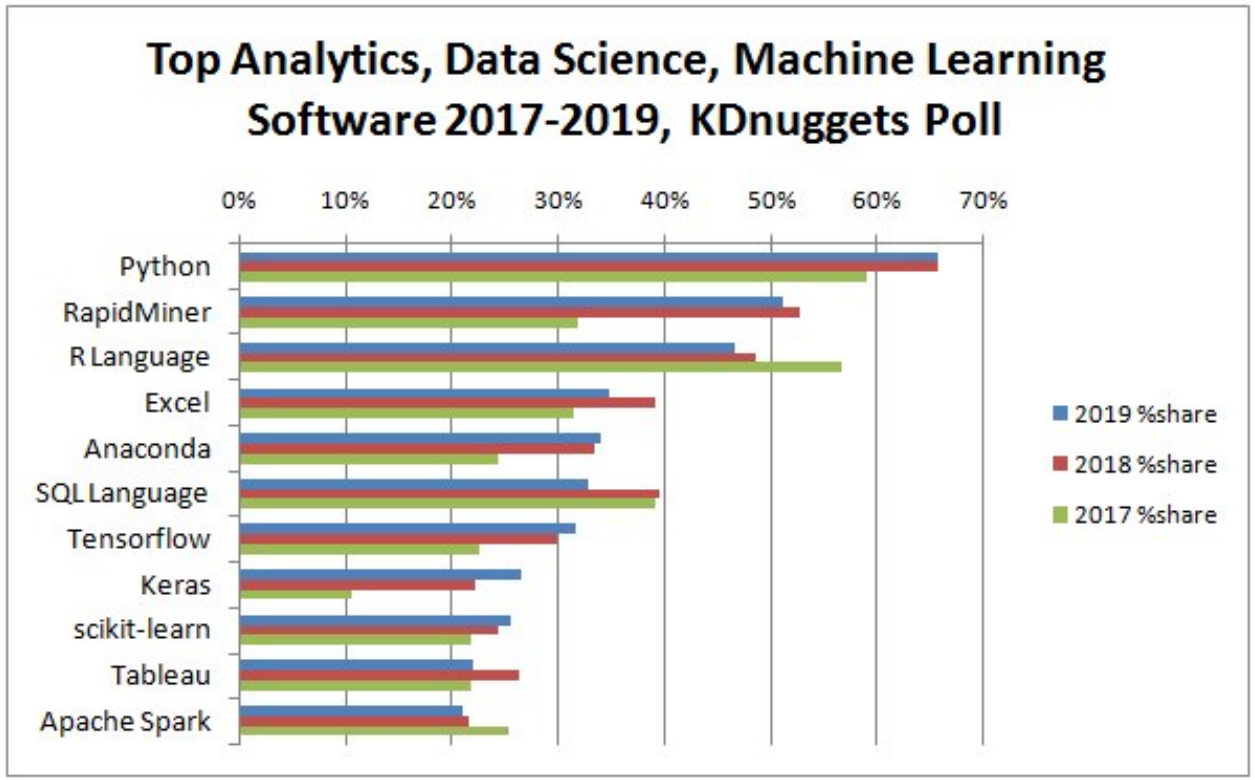
GitHub Copilot

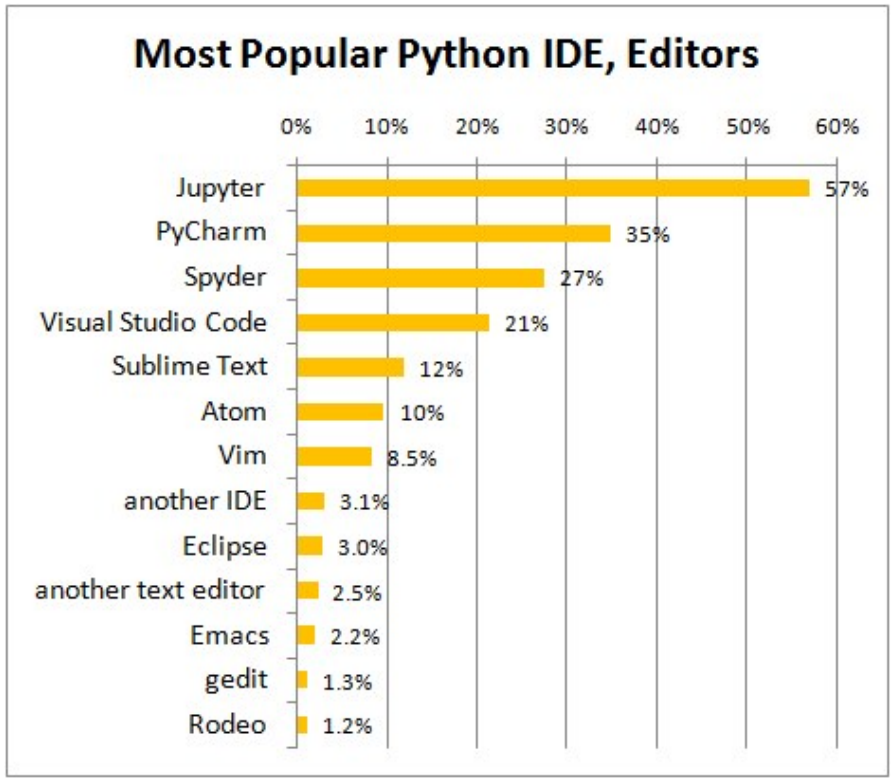
- GitHub Copilot - Automatically generates source code - makes code recommendations in any language
- No lengthy doc files or searching online for examples
- OpenAI and GitHub (Microsoft) – algorithm vs data?
- Distinction between public vs private code
- OpenAI Codex, based on GPT-3
- GPT-3 demo? https://bellard.org/textsynth/
- https://copilot.github.com/

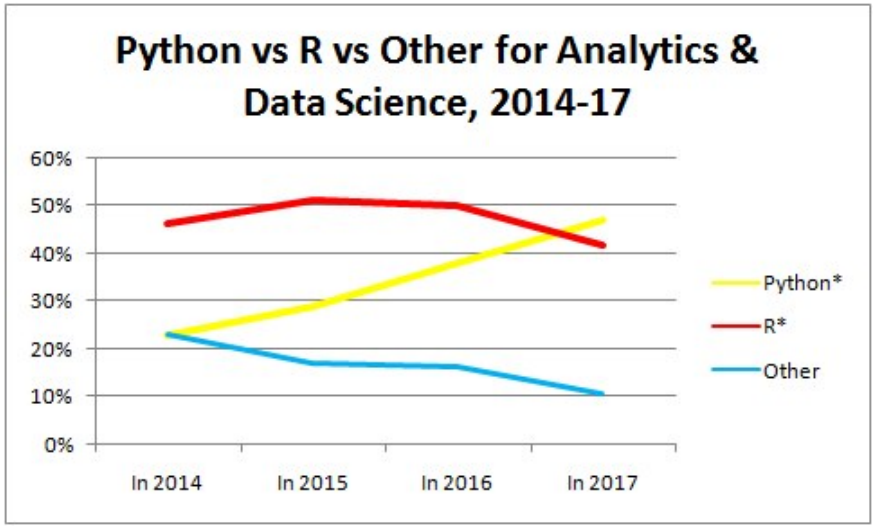


- Also, Power Apps Studio – fine tunes OpenAI GPT-3 to automatically generate Power Fx formulas
- https://powerapps.microsoft.com/en-us/blog/introducing-power-apps-ideas-ai-powered-assistance-now-helps-anyone-create-apps-using-natural-language/

Tools for Building AI

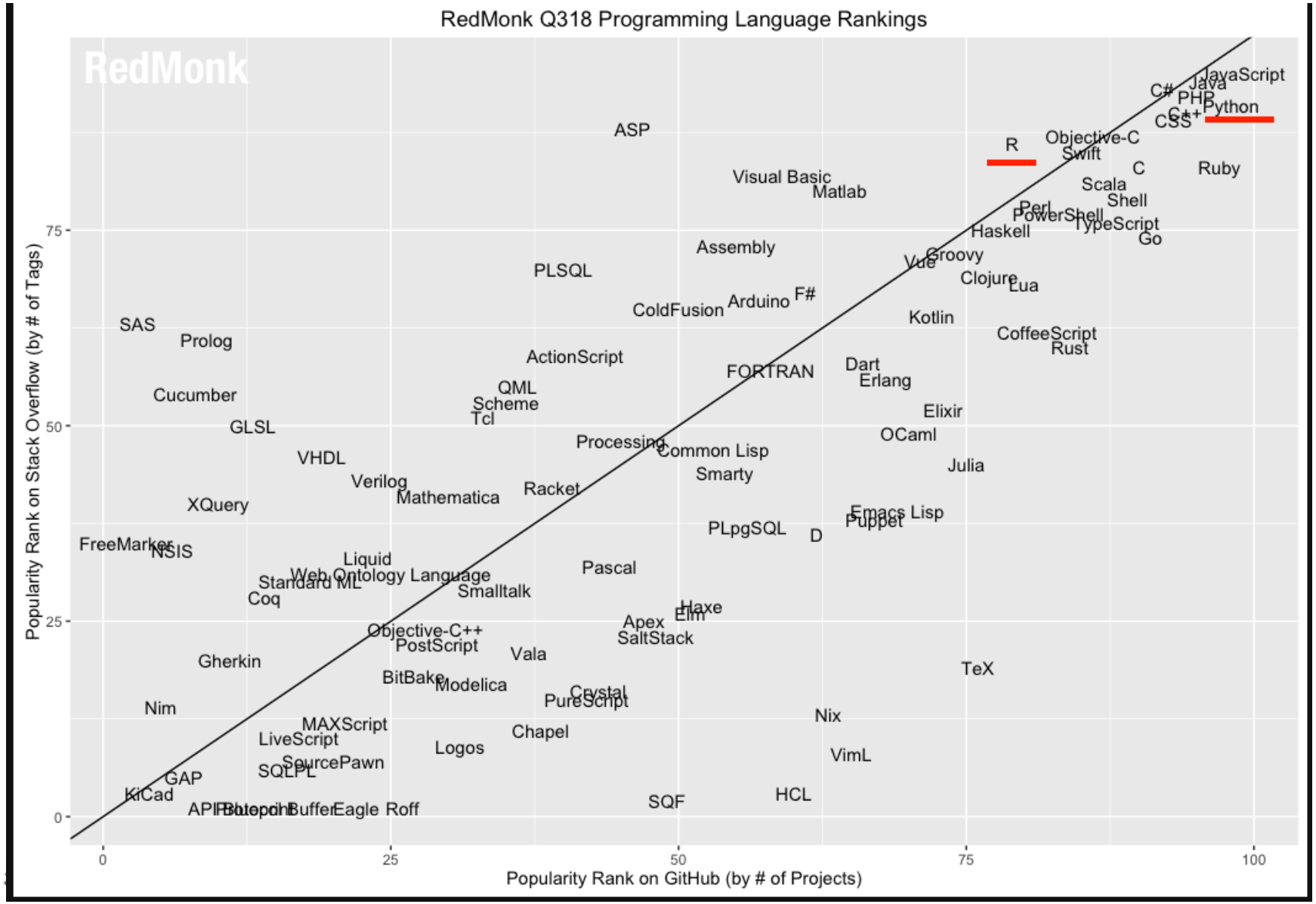






https://www.kdnuggets.com/2019/05/poll-top-data-science-machine-learning-platforms.html

Another take..



- www.stackoverflow.com
- www.github.com

Python

- Invented in 1989 and released in 1991 by Guido Van Rossum, a Dutch programmer.
- It is an ‘easy to adapt’ scripting language built on 19 concept pillars (Zen of Python)
    - Beautiful is better than ugly
    - Explicit is better than implicit
    - Simple is better than complex
    - Complex is better than complicated
    - Readability counts …
- Adopted by many large tech organisations as well as start-ups for their programming and data analytics tasks. (Google, Facebook, Instagram, Spotify, Quora, Netflix, Dropbox)
- A large community, more than 200,000 - https://pypi.org/
- Multiple versions
    - Version 2 (v2.7) – from 2000 to 2020
    - Current, version 3 (commonly used version is 3.6) / latest is 3.8

Zen of Python

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>> _
```

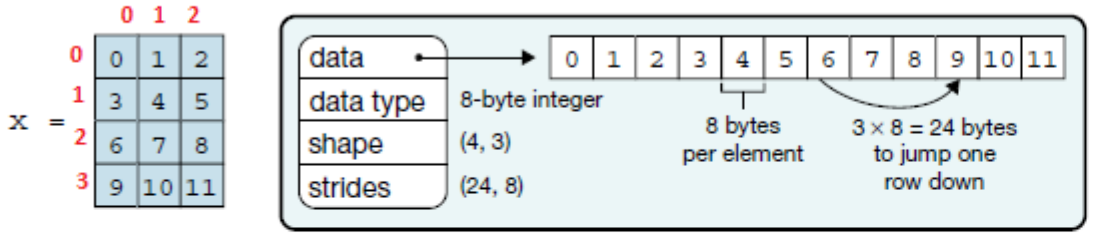
- "import this"
- Tim Peters, 1999
- A 20th principle for Guido to fill in (still empty)
- Do these explain why Python is popular for AI?
  - Large community
  - Rich AI libraries
  - Low barriers to entry
  - Effective prototyping
  - Simple and easy

Python libraries - NumPy

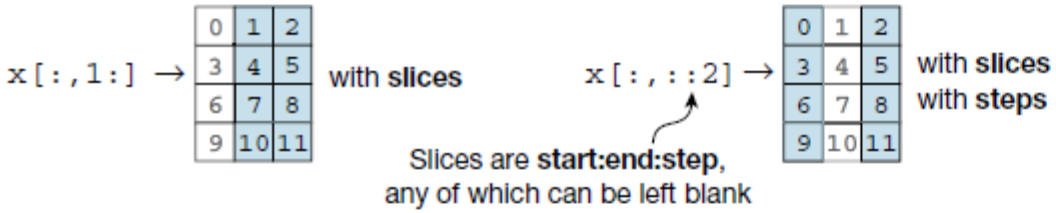
- NumPy (Numerical Python)
  - Primary array programming library for Python – although not part of Python's standard library
  - Is an enabler for most scientific or numerical computation libraries - matplotlib, pandas, scikit-learn etc
  - Crucial for research analysis pipelines - used in the discovery of gravitational waves and the first imaging of a black hole
  - The NumPy array is a data structure that efficiently stores and accesses multidimensional arrays (or tensors)
  - Consists of a pointer to memory and metadata used to interpret the data - data type, shape and strides
  - Only limitation is its in-memory CPU based model, which cannot capitalise on GPUs, TPUs, FPGAs
  - This gap has been addressed by some of the deep learning frameworks with their own own arrays; PyTorch, Tensorflow, Apache MXNet
  - But also, projects like Dask, which build upon NumPy arrays as data containers and advance its capabilities into distributed and parallel processing.
  - An interoperability layer between array computation libraries and APIs

NumPy array

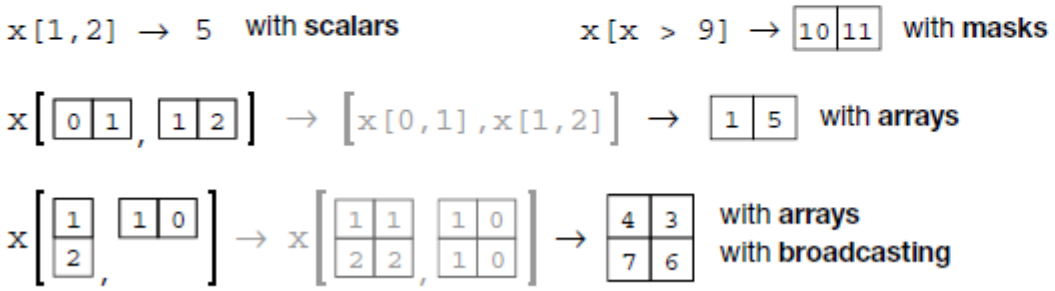
a Data structure



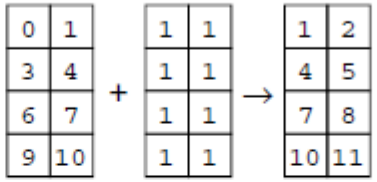
b Indexing (view)



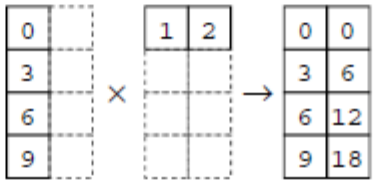
c Indexing (copy)



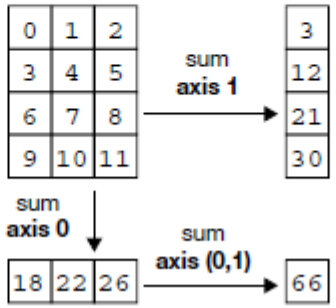
d Vectorization



e Broadcasting



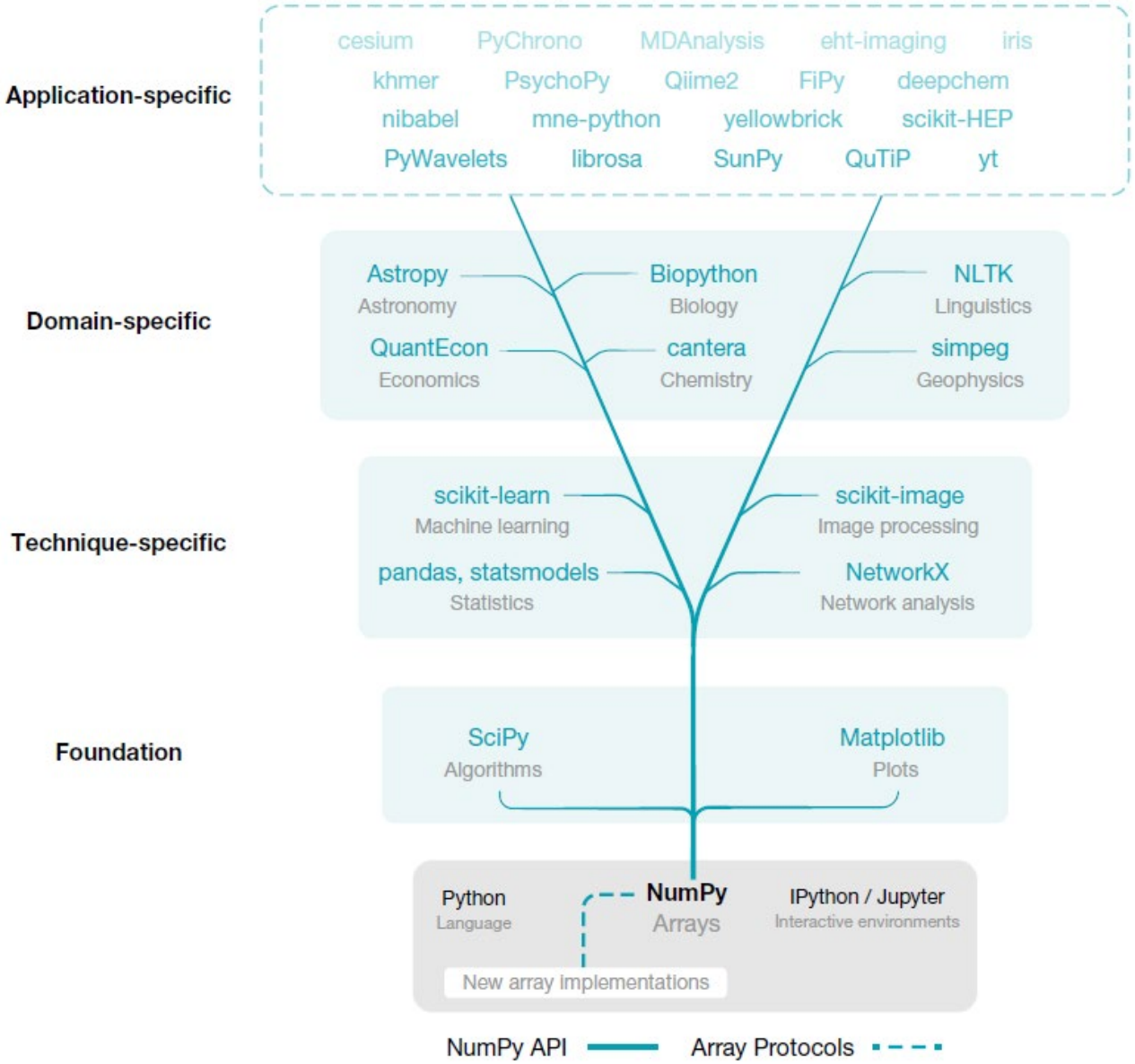
f Reduction



g Example

```
In [1]: import numpy as np

In [2]: x = np.arange(12)

In [3]: x = x.reshape(4, 3)

In [4]: x
Out[4]:
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

In [5]: np.mean(x, axis=0)
Out[5]: array([4.5, 5.5, 6.5])

In [6]: x = x - np.mean(x, axis=0)

In [7]: x
Out[7]:
array([[-4.5, -4.5, -4.5],
       [-1.5, -1.5, -1.5],
       [ 1.5,  1.5,  1.5],
       [ 4.5,  4.5,  4.5]])
```

NumPy as an enabler

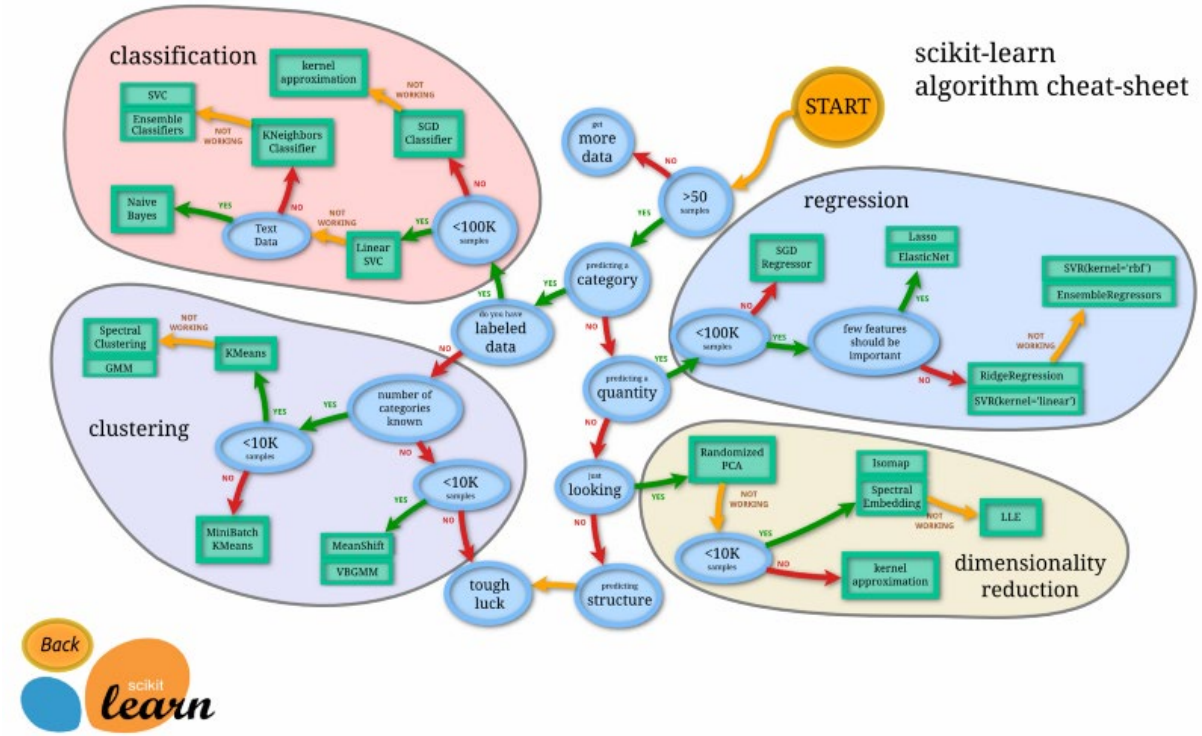


Pandas, MatplotLib and Seaborn

- Pandas
  - Built on NumPy
  - Data wrangling (like SQL)
  - Two data structures: DataFrame (series of columns and rows) and Series (a 1-dimensional array)
  - Contains a wide range of data import and export functions, as well as for manipulating data
  - Functions for selecting, reshaping, subset, merging, splitting, summarizing, grouping
  - Handling missing data (drop or fill)
- Matplotlib and Seaborn
  - Data visualisation - more control over elements on a visualisation
  - Seaborn is an extension built upon matplotlib, for more visually appealing graphs

Scikit-learn

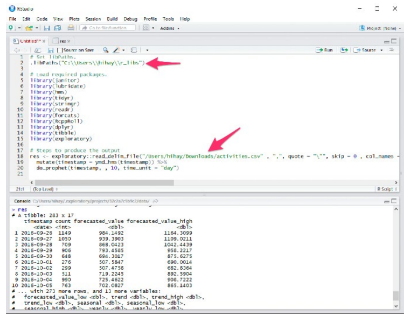
- Scikit-learn – AI and machine learning
- Minimal dependencies (NumPy and SciPy)
- Distributed under the simplified BSD license
- Can also select parameters using cross-validation
- Distributes computation to several cores
- Time in seconds on the Madelon dataset (, 4400 instances and 500 attributes)



	scikit-learn	mlpy	pybrain	pymvpa	mdp	shogun
Support Vector Classification	**5.2**	9.47	17.5	11.52	40.48	5.63
Lasso (LARS)	**1.17**	105.3	-	37.35	-	-
Elastic Net	**0.52**	73.7	-	1.44	-	-
k-Nearest Neighbors	0.57	1.41	-	**0.56**	0.58	1.36
PCA (9 components)	**0.18**	-	-	8.93	0.47	0.33
k-Means (9 clusters)	1.34	0.79	⋆	-	35.75	**0.68**
License	BSD	GPL	BSD	BSD	BSD	GPL

-: Not implemented. ⋆: Does not converge within 1 hour.

Coding in Python – IDE options



1. Local IDE

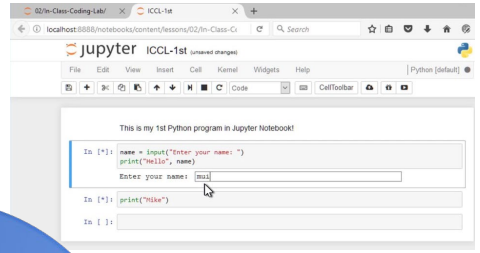
1. Install Python on your computer and use an IDE (Integrated Development Environment)

e.g. PyCharm, Visual Code, IDLE, Atom

Python Environments

3. Use Jupyter Notebook on Google Colab – no installations on your computer.

3. Jupyter on Google Colab



2. Jupyter on localhost

2. Install Python and Jupyter on your computer and then use Jupyter Notebooks on localhost.

Notebooks

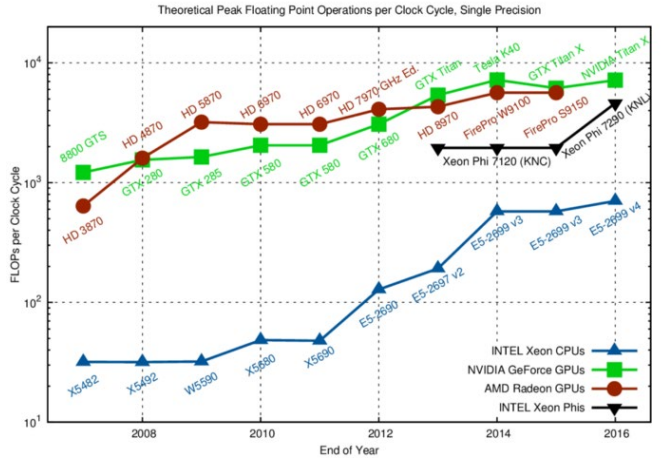
- Computational notebooks are an outcome of the Big Data/Cloud computing era
  - “move the computer to the data than the data to the computer”
- Jupyter is one type of computational notebook widely used by AI researchers and data scientists
  - Jupyter is derived from the main languages that are supported: Julia, Python, and R
  - IPython Notebook is Jupyter’s predecessor, retains the file extension .ipynb
  - Other types include R Markdown, Apache Zeppelin, and Spark Notebook.
  - 2.5 million public Jupyter notebooks on GitHub in Sep 2018
- It is an interactive IDE or programming environment that combines coding, scripting, documenting and presenting on a single web interface.
  - software code, computational output, explanatory text and multimedia resources

Jupyter Notebooks

- A Jupyter Notebook has two main components:
- An interactive front-end web page (just click run)
  - Inputs: software code, explanatory text, multimedia resources
  - Output: computational output, error messages
  - The page is saved as a Notebook file with a ".ipynb" extension.
- A back-end kernel
  - Executes the code and returns the result (separate kernel for each language)
  - Usually one notebook runs only one kernel in one language, but not always.
  - The kernel can reside in your own computer, on a remote server or a cloud service (like Google Colab)

Google Colab

- Google Colaboratory (Colab) is a free cloud service to write, store and run Jupyter notebooks.
- Notebooks and datasets in Google Drive
- Access to GPU (Graphical Processing Units) and TPU (Tensor Processing Units)
  - Originally used to render graphics/computer games.
  - GPU has dedicated memory – handles large datasets, whereas CPU conducts multiple reads
  - GPU has more cores than CPU – enables multiple simultaneous computations
  - CPU vs GPU - Serial vs parallel - https://www.youtube.com/watch?v=-P28LKWTzrI
- TPUs specifically designed for Google's TensorFlow framework
  - Customised for AI, 3.5x faster than GPU



Jupyter – other features

- JupyterLab - a “next-generation web interface”
  - Extends the notebook with drag-and-drop, file browsers, file/csv viewers, text editors
  - More than a kernel, an entire computing environment/ VM
- JupyterHub - identical computing environments to a large pool of users (DevOps)
- Binder / Code Ocean – integrates Jupyter notebooks with GitHub on a web browser without having to install the software or any programming libraries
  - Colab option - prepend https://colab.research.google.com/github to the GitHub URL of a notebook

Jupyter – limitations

- Lack of modularity – linear declaration of functions/libraries/classes/variables
- Hidden state of execution - running code out of sequence, deletions, edits after running
- Limited testing pathways - unit tests, module tests
- Sandboxed from technical issues - requires extra testing
- Limitations when collaborating and lack of version control
- Poor coding practice – notebook names, variable settings