Lecture 2
Tools for Machine Learning

Venkatakrishnan V K

ECE 208/408 - The Art of Machine Learning

1/28/2026

(Special thanks to Baotong Tian and Melissa Chen for their contributions to the slides)

UNIVERSITY of ROCHESTER

Audio Information Research Laboratory

Table of Contents

Table of Contents

Linux Shell Scripts

What is a Unix shell?

Command-line Interpreter [1]

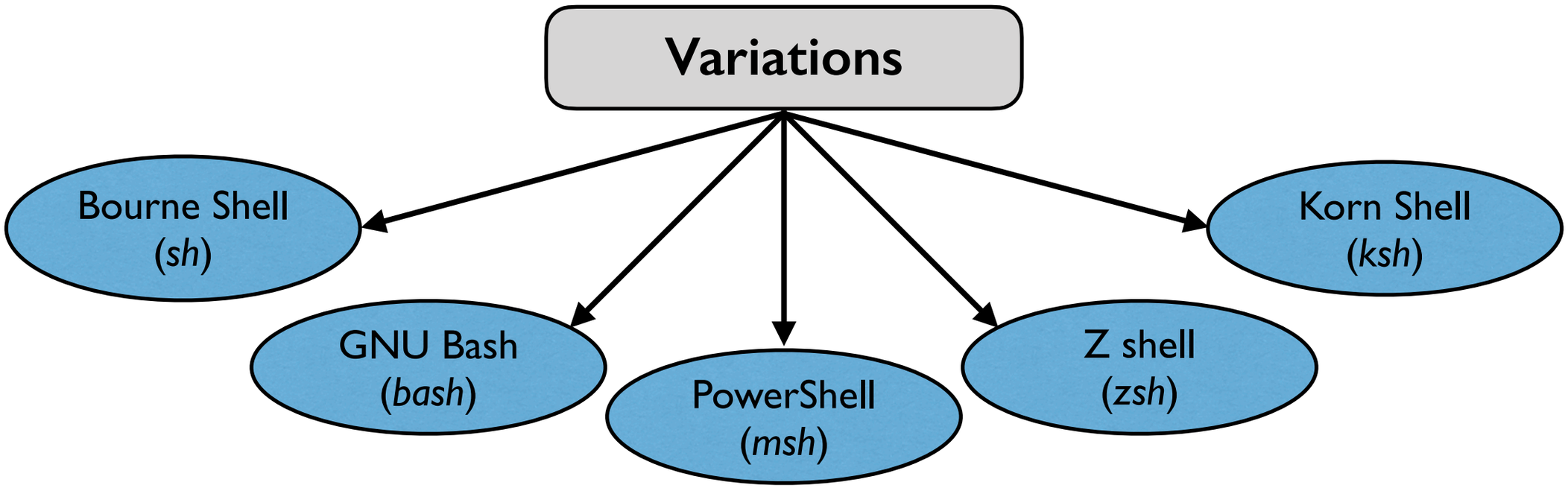
- "**Terminal**" is where you generally interact with the shell
- Examples of Unix-like systems —

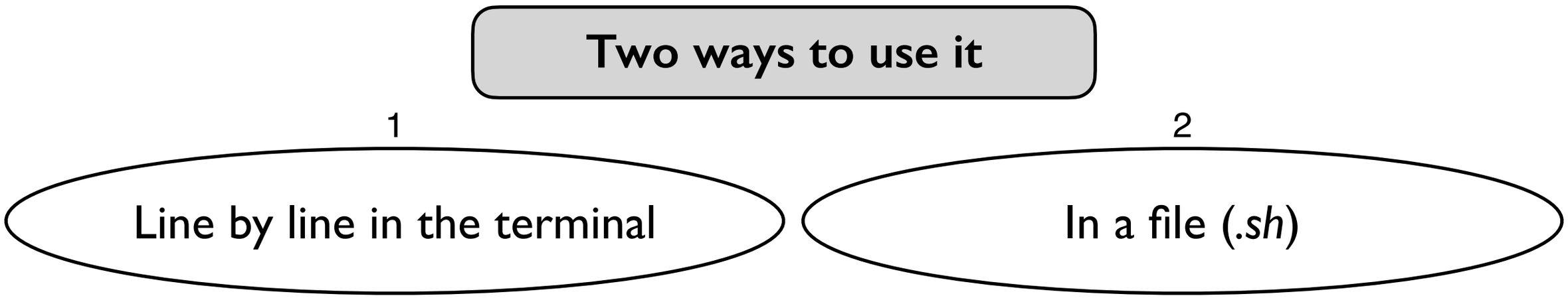
Linux **MacOS**

For Windows, read about "batch files"

[1] Kernighan, Brian W.; Pike, Rob (1984), "3. Using the Shell", The UNIX Programming Environment, Prentice Hall, Inc., p. 94, ISBN 0-13-937699-2

Linux Shell Scripts





Linux Shell Scripts

Usage

- ***Basic Structure***: `command -[option] parameter1 parameter2 …`
- ***Multiple Commands***: `command1 | command2 | command3 …`

Application Package Tool (APT)

- The main command-line package manager for Debian and its derivatives

Examples:

```
$ apt update && sudo apt upgrade
$ apt install xxx
$ apt remove xxx
```

Interact with the system

- Cheatsheet: https://github.com/RehanSaeed/Bash-Cheat-Sheet

Linux Shell Scripts

System and Hardware Monitor

CPU

`htop`

For Mac users with HomeBrew installed, run
`brew install htop`
to install `htop`

GPU

`nvidia-smi`

`watch -n 2 | nvidia-smi`

`gpustat` [2]
(Dynamic, recommended)

```
❯❯❯ gpustat -cp
dali.vision  Thu Jun  2 23:46:16 2016
[0] GeForce GTX TITAN X | 77'C,  96 % | 11848 / 12287 MB | python/52046(11821M)
[1] GeForce GTX TITAN X | 75'C,   9 % | 11848 / 12287 MB | python/52046(11821M)
[2] GeForce GTX TITAN X | 75'C,  39 % | 12015 / 12287 MB | python/52046(11821M) python/128424(165M)
```

Source: gpustat

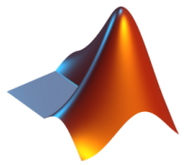
[2] https://github.com/wookayin/gpustat

Table of Contents

Why use Python?

Popular candidates


Python


MATLAB


R


Java


C++


C

Middle-level or High-level?	**C**	▸ Fast, efficient, portable ▸ Hard to write/understand
	Python	▸ Highly abstracted from computer hardware ▸ Easy to understand

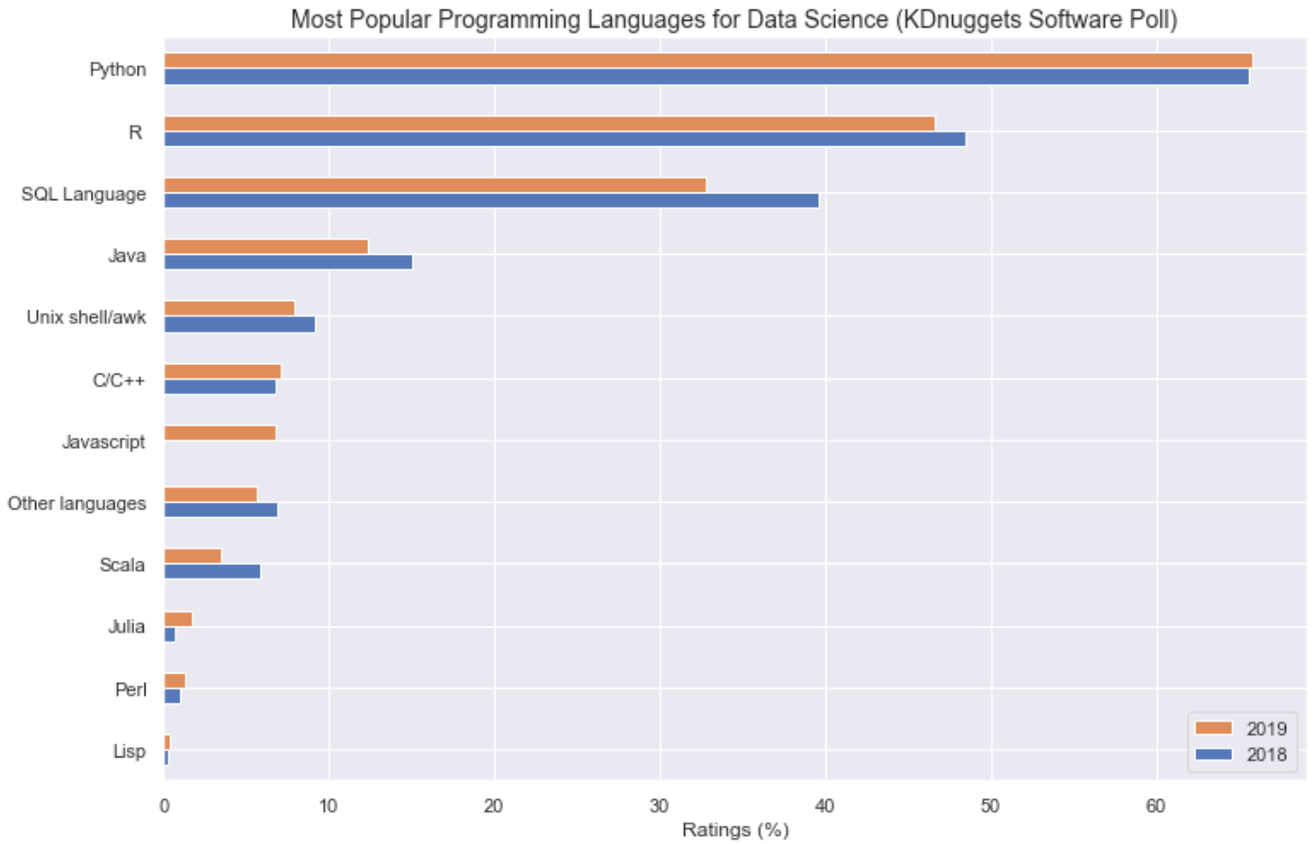
Compiled or Interpreted?	**Java/C++**	▸ Fast, protected source code ▸ Can be more difficult to debug
	Python/ MATLAB	▸ Interpreted line-by-line and on-the-fly ▸ Flexible and cross-platform

ML Ecosystem and Developer Community

Python has the most active ML developer community

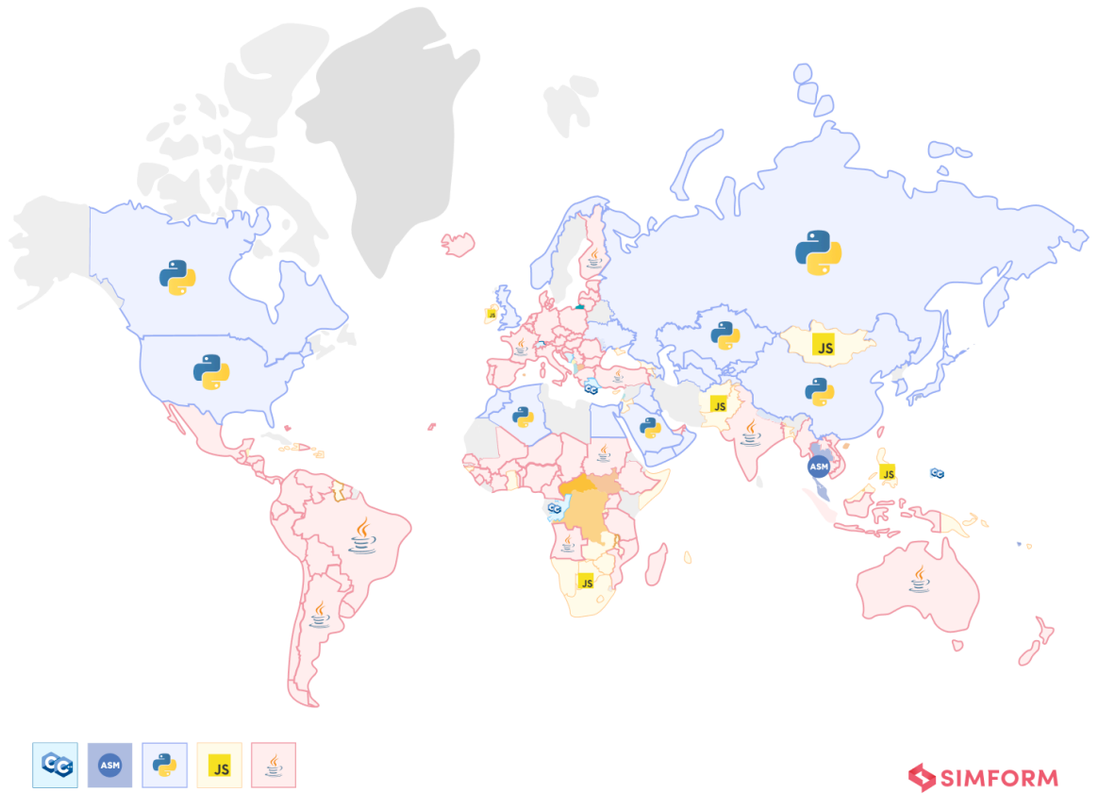
PyTorch, TensorFlow, Keras, …

Programming Language for ML



Source: KDnuggets

Most Popular Languages in Every Country



Source: Simform

There is no such thing as a "best language for machine learning."

The choice of language largely depends on specific applications and devices.

Python Basics

Installation

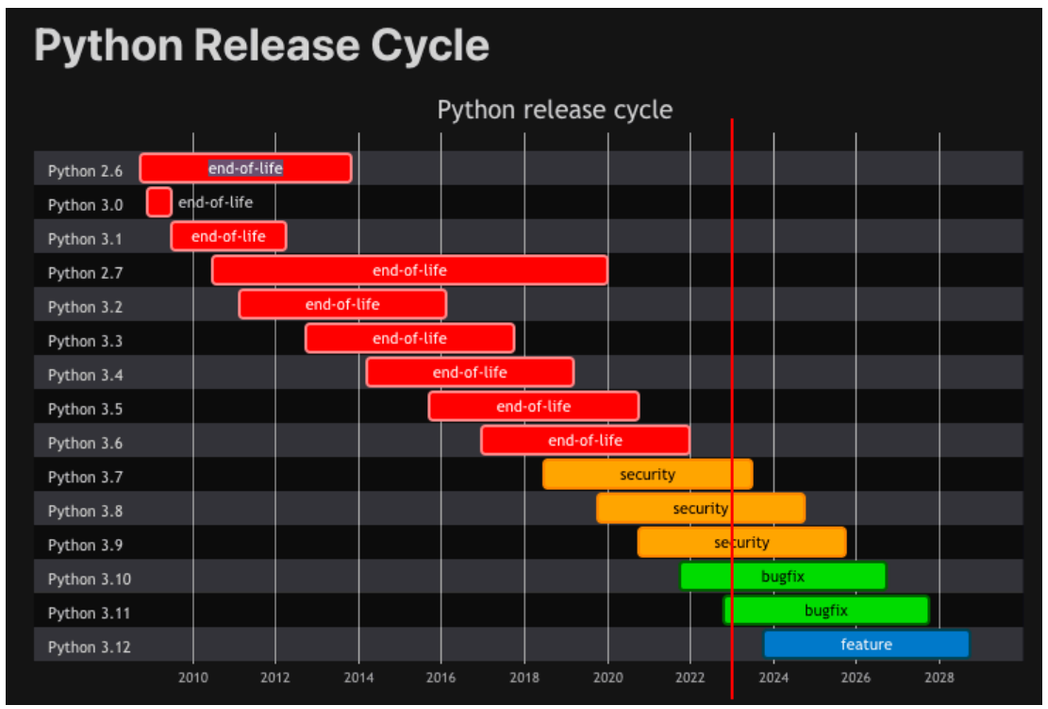
Download and double click
https://www.python.org/downloads/

Install packages from a requirements file

```
$ sudo apt-get update
$ sudo apt-get install python3.12
```

Install in Anaconda

```
$ conda install python=3.8
```



Source: https://devguide.python.org/versions/

Versions

Python Basics

Pip

- `pip` is the package installer for Python
- Included with modern versions of Python

Pip cheat sheet
https://opensource.com/sites/default/files/gated-content/cheat_sheet_pip.pdf

PyPI **Python Package Index** is the official third-party software repository for Python.

Install specific version

```
$ pip install requests==2.22.0
```

Install packages from a requirements file

```
$ pip install -r requirements.txt
```

Capture all currently installed versions in a text file

```
$ pip freeze > requirements.txt
```

Virtual Environment

Anaconda

Anaconda is a distribution of the Python and R for scientific computing [1]

- Aims to simplify package management and deployment
- For Windows, Linux, and macOS

Installation: https://www.anaconda.com/products/distribution#linux
Usage: https://docs.conda.io/projects/conda/en/4.6.0/_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf

Create a new environment named py35, install Python 3.5	`conda create --name py35 python=3.5`
Activate the new environment to use it	`WINDOWS: activate py35` `LINUX, macOS: source activate py35`
Get a list of all my environments, active	`conda env list`
Install a new package (Jupyter Notebook) in the active environment	`conda install jupyter`
Remove one or more packages (toolz, boltons) from a specific environment (bio-env)	`conda remove --name bio-env toolz boltons`
Delete an environment and everything in it	`conda env remove --name bio-env`
Deactivate the current environment	`WINDOWS: deactivate` `macOS, LINUX: source deactivate`

[1] https://en.wikipedia.org/wiki/Anaconda_%28Python_distribution%29

Project Management

UV

- ***PyPI Installation***: `pip install uv`
- ***Linux and MacOS Installation:***

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

- Extremely fast Project and Package Manager, written in **Rust**.
- Once installed, go to a directory to create the project and initialize it using —

```
uv init <project-name>
```

- This creates a few files, including `pyproject.toml`, which keeps track of all the project details.
- To create a virtual environment, use —

```
uv venv
```

- You can also create a `requirements.txt` file that contains all the dependencies with the versions to be installed, then run —

```
uv add -r requirements.txt
```

- For further details, visit: https://docs.astral.sh/uv/

Python Basics

Data Types

String (`str`)	▸ A **string** is a sequence of characters. ▸ Anything inside quotes [single quotes (‘…’) or double quotes (“…”)] is a **string**.
Boolean (`bool`)	▸ **True**/**False** values. ▸ Can be used as integer **1** or **0**.
Integer (`int`)	▸ **Whole numbers** and **negative numbers** without a decimal point. ▸ Can be **positive, negative,** or **zero.**
Decimal (`float`)	▸ Numbers that contain decimal points (32-bit precision). ▸ 64-bit double precision (`double`)

Encoding

UTF-8 (Default)

ASCII

Integer Operations

- **Addition** (`+`)
- **Subtraction** (`-`)
- **Multiplication** (`*`)
- **Quotient** (`//`)
- **Modulus** (`%`)
- **Exponent** (`**`)

String Concatentation

```
3*‘un’ + ‘ium’

= unununium
```

Python Basics

Data Structures

Lists [...]	Dictionaries {... : ...}	Sets {...}	Tuples (...)
▸ A **collection of items** that are *ordered* and *mutable* ▸ Lists can contain items of different types	▸ A **collection of items** that are *unordered*, *mutable* and *indexed* ▸ Each item is indexed by a **key**. ▸ The correspoding item is called the **value**	▸ A collection of items that are *unordered, immutable* and *unindexed* ▸ The set is **mutable**, but the elements contained in a set must be **unique** and **immutable** ▸ Sets seem very similar to lists, but they are very different	▸ A collection of items that are *ordered* and *immutable* ▸ Almost the same as **List**, but cannot be modified once created

Python Basics

Data Structures

Lists [...]	Dictionaries {... : ...}	Sets {...}	Tuples (...)

```
squares = [1, 4, 9, 16, 25]
# indexing
squares[0] # 1
# slicing
squares[-3:] # [9, 16, 25]
# appending
squares + [36, 49] # [1, 4, 9, 16, 25, 36, 49]
squares.append(216) # [1, 4, 9, 16, 25, 216]
# looping
for element in squares:
    print(element)
# 1
# 4
# 9
# 16
# 25
# 216
```

```
qunt = {'apple': 2, 'orange': 4}
# looping
for k, v in qunt.items():
    print(k, v)
# apple 2
# orange 4

# indexing
qunt["apple"] # 2
# add element
qunt["grape"] = 7
```

```
s1 = {1, 2, 3, 5}
s2 = set([1, 2, 3, 4])
# intersection
s_intersection = s1.intersection(s2) # {1, 2, 3}
s_intersection = s1 & s2 # {1, 2, 3}
# difference
s_difference = s1.difference(s2) # {5}
s_difference = s1 - s2 # {5}
```

```
t1 = (1, 2, 3, 4)
t2 = tuple([1, 2, 3, 4, 5])
```

Python Basics

Control Flow Tools and Functions

```
#·this·is·a·comment

"""
this·is·a·comment
written·in
several·lines
"""

#·if·statement
if·x·<·0:
····x·=·0
elif·x·==·0:
····print('Zero')
else:
····print('More')

#·for·loop
words·=·['cat',·'window',·'defenestrate']
for·w·in·words:
····print(w,·len(w))

#·range·function
for·i·in·range(0,·10,·3):
····print(i)

#·break
for·n·in·range(2,·10):
····for·x·in·range(2,·n):
········if·n·%·x·==·0:
············print(n,·'equals',·x,·'*',·n//x)
············break
····else:
········print(n,·'is·a·prime·number')


#·continue
for·num·in·range(2,·10):
····if·num·%·2·==·0:
········print("Found·an·even·number",·num)
········continue
····print("Found·an·odd·number",·num)
```

```
#·break
for·n·in·range(2,·10):
····for·x·in·range(2,·n):
········if·n·%·x·==·0:
············print(n,·'equals',·x,·'*',·n//x)
············break
····else:
········print(n,·'is·a·prime·number')


#·continue
for·num·in·range(2,·10):
····if·num·%·2·==·0:
········print("Found·an·even·number",·num)
········continue
····print("Found·an·odd·number",·num)


#·pass
#·The pass statement·does·nothing.·
#·It·can·be·used·when·a·statement·is·required·syntactically·
#·but·the·program·requires·no·action.
class·MyEmptyClass:
····pass


#·define·and·call·functions
def·fib(n):
····a,·b·=·0,·1
····while·a·<·n:
········print(a,·end='·')
········a,·b·=·b,·a+b

fib(2000)
```

Python Basics

Classes

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def myfunc(self):
    print("Hello my name is " + self.name)

p1 = Person("John", 36)
p1.name
p1.myfunc()
```

The `self` parameter is a reference to the current instance of the class, and is used to access variables and functions that belongs to the class.

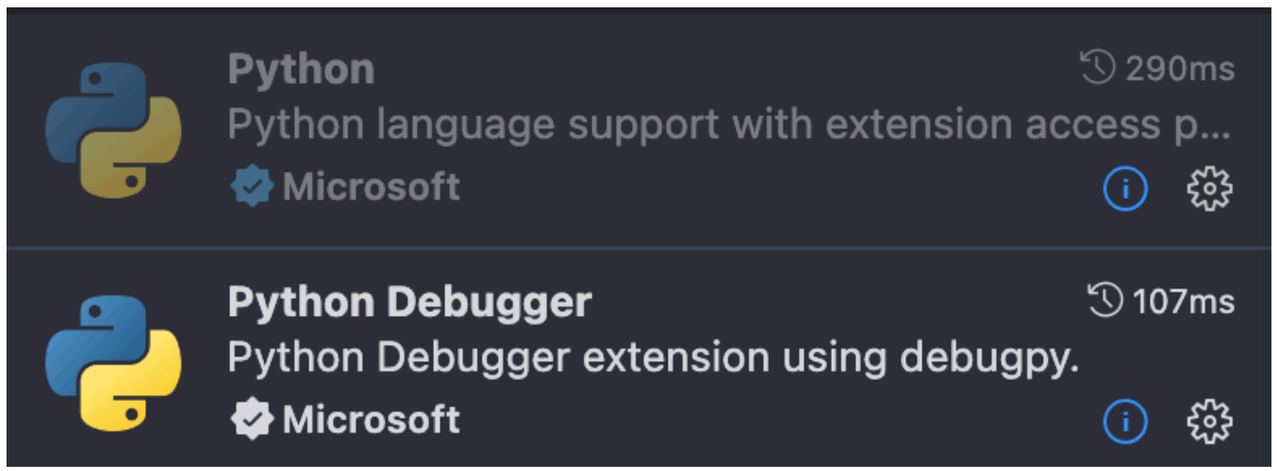
Debugging

Why Debug?

- Debugging is essential for identifying and fixing errors in programming
- Helps improve code reliability, maintainability, and performance

Debugging With IDE

- **Visual Studio Code** provides an extension
- Built-in **PyCharm** debugger



Debugging

PDB and IPDB

- **pdb**: Built-in Python debugger for stepping through code, setting breakpoints, and inspecting variables
- **ipdb**: An enhanced version of pdb with IPython features for a more interactive experience
- ***Installation***: `pip install ipdb`
- ***Example usage:*** `import ipdb; ipdb.set_trace()`

Advantages

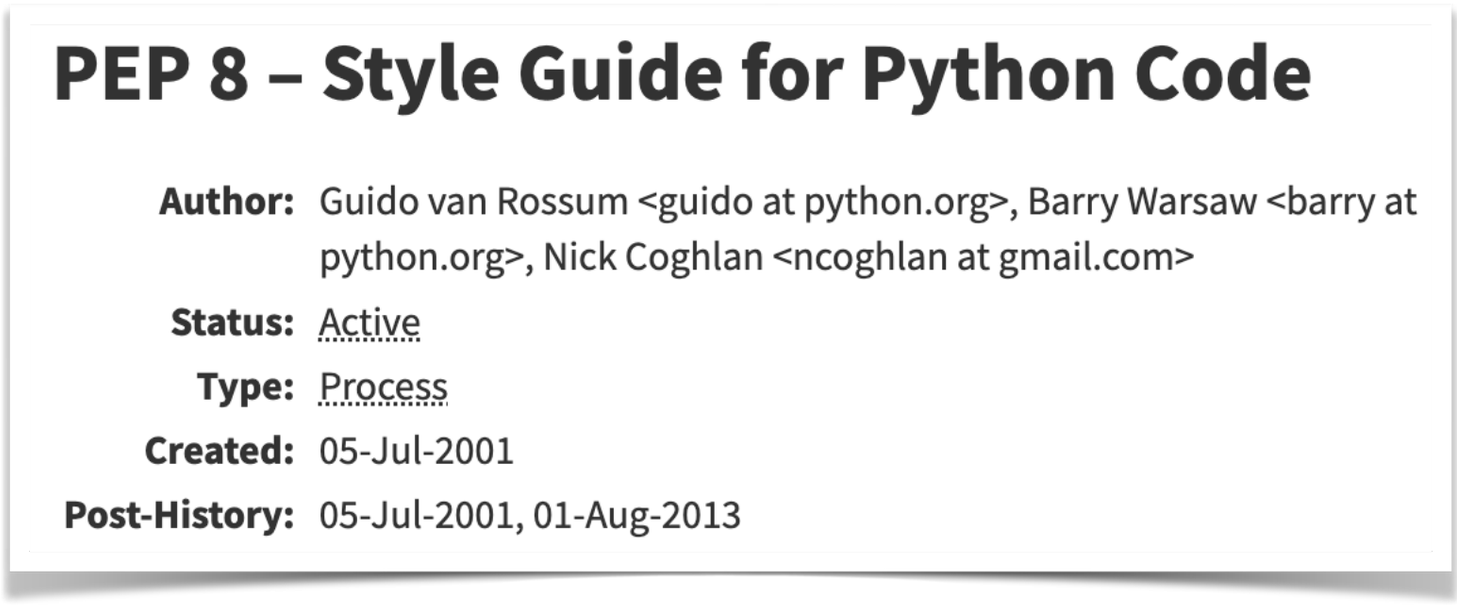
- Run scripts on command line with parameters

```
python train.py -c config.json
python train.py --batch-size 16 --learning_rate 1e-3
```

- (Also able to debug in IDE: e.g. create launch.json in the folder)

Python Style Guide

"Code is read much more often than it is written."
— Guido van Rossum



PEP 8 – Style Guide for Python Code

Author: Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>

Status: Active

Type: Process

Created: 05-Jul-2001

Post-History: 05-Jul-2001, 01-Aug-2013

Python Enhancement Proposal 8
Recommended by creators of Python
Intended to improve the readability of code and make it consistent

Source: https://peps.python.org/pep-0008/

How To Write Beautiful Code with Python

- Use 4 spaces per indentation level.
- Spaces are the preferred indentation method.
- Allow mixing tabs and spaces, but keep consistent!
- Continuation lines should align wrapped elements vertically.

```
# Align
foo = long_function_name(var_one, var_two,
                         var_three, var_four)

def long_function_name(
        var_one, var_two, var_three,
        var_four):
    print(var_one)

# Hanging indents should add a level
foo = long_function_name(
    var_one, var_two,
    var_three, var_four)
```

- Imports should usually be on separate lines.

```
import os
import sys
```

- Limit all lines to a maximum of 79 characters.

```
with open('/path/to/some/file/you/want/to/read') as file_1, \
        open('/path/to/some/file/being/written', 'w') as file_2:
    file_2.write(file_1.read())
```

- Single-quoted strings and double-quoted strings are the same.

Rule of thumb

- Use object-oriented programming style in multi-file complex projects.
- Many projects have their own coding style guidelines. Find one and start with a good example.
- Pick your rule and stick to it.

A Good example: https://github.com/brentspell/hifi-gan-bwe

Learning Resources

Step-by-step guide

https://www.w3schools.com/python/

Official document

https://docs.python.org/3.8/tutorial/index.html

Coursera python courses

https://www.coursera.org/search?query=python&

The Hitchhiker's Guide to Python

https://docs.python-guide.org/

Jupyter Notebook

- ***Installation***: `pip install notebook`

Description

Jupyter Notebook is a web-based interactive development environment (IDE)

- Contain live code, equations, visualizations, and narrative text
- Easy create and share documents

Jupyter Notebook is written in Python, but it supports over 40 programming languages, including **Python**, **R**, **Julia**, and **Scala**.

Usage

Intro example here:
https://jupyter.org/try-jupyter/notebooks/?path=notebooks/Intro.ipynb

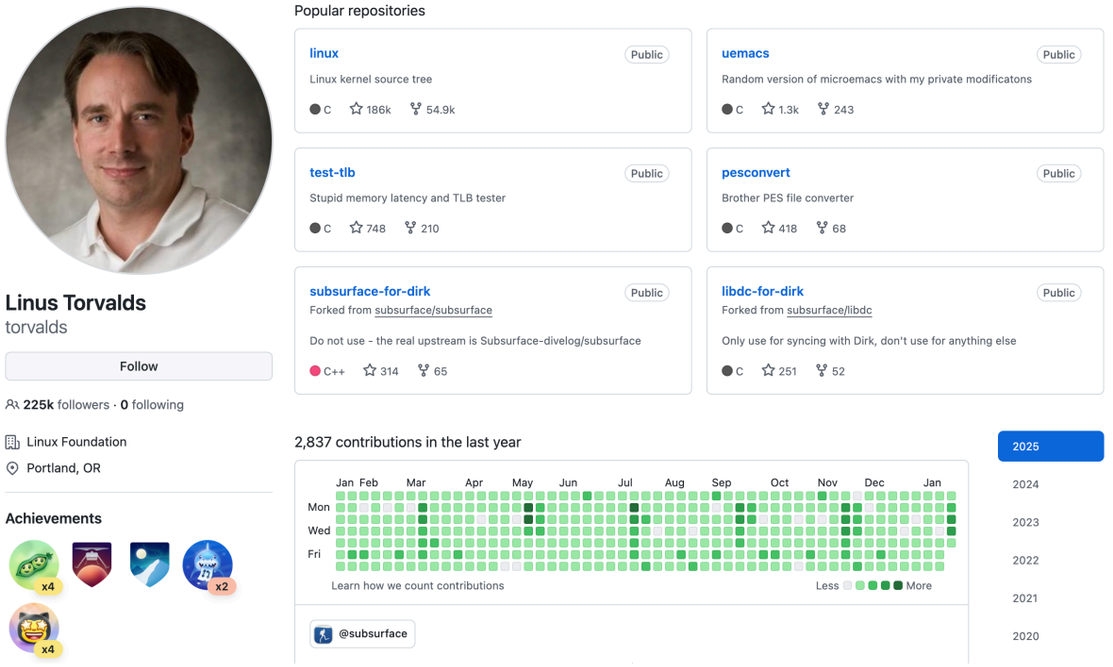
Version Control

Keep Track of Your Codes!

- You will not want to do “train.py”, “train_v1.py”, “train_v1.1_with_additional_data.py”…
- A lot of options: **Git**/Mercurial/SVN

Git: the most popular among individuals

- Good resource to learn
  - Tutorial videos: https://git-scm.com/videos
  - Visualized learning platform: https://learngitbranching.js.org/
- Code hosting
  - github, bitbucket, gitlab, etc.



Version Control

Basics

Setup

`git config --global user.name "[firstname lastname]"`
set a name that is identifiable for credit when review version history
`git config --global user.email "[valid-email]"`
set an email address that will be associated with each history marker

Initialize

`git init`
initialize an existing directory as a Git repository
`git clone [url]`
retrieve an entire repository from a hosted location via URL

Stage

`git status`
show modified files in working directory, staged for your next commit
`git add [file]`
add a file as it looks now to your next commit (stage)
`git commit -m "[descriptive message]"`
commit your staged content as a new commit snapshot

Branch and Merge

`git branch`
list your branches. a * will appear next to the currently active branch
`git branch [branch-name]`
create a new branch at the current commit
`git checkout`
switch to another branch and check it out into your working directory
`git merge [branch]`
merge the specified branch's history into the current one

Table of Contents

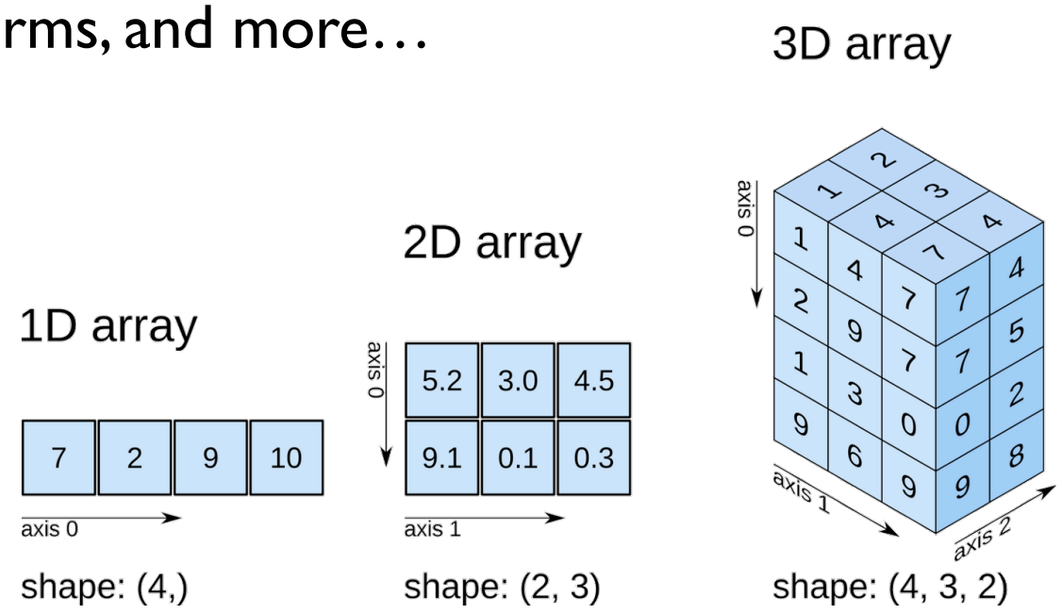
Numerical Computing (numpy)

NumPy

- ***Installation***: `pip install numpy`
- ***Conda installation:*** `conda install numpy`

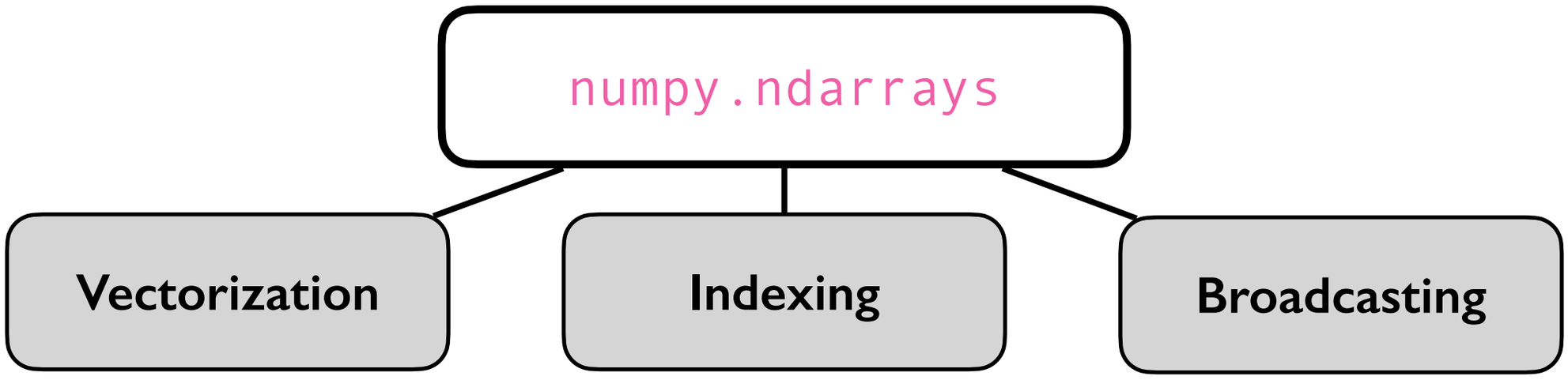
Description

- Fast and versatile
- Availability of -
  - Mathematical functions
  - Random number generators
  - Linear algebra routines
  - Fourier transforms, and more…



Numerical Computing (numpy)

Core concepts




```
>>> import numpy as np
>>> a = np.array([3, 1, 2])
>>> np.sort(a)
array([1, 2, 3])
>>> a.shape
(3,)
>>> b = np.array([4, 5, 6])
>>> a = np.concatenate((a, b), axis=0)
>>> a.shape
(6,)
>>> c = a.reshape(3, 2)
>>> c.shape
(3, 2)
>>> d = np.expand_dims(a, axis=1)
>>> d.shape
(6, 1)
```

Basic operations

```
>>> a
array([3, 1, 2, 4, 5, 6])
>>> e = np.array([100])
>>> a+e
array([103, 101, 102, 104, 105, 106])
```

Broadcasting

Data Analysis (pandas)

Pandas

- ***Installation***: `pip install pandas`

Description

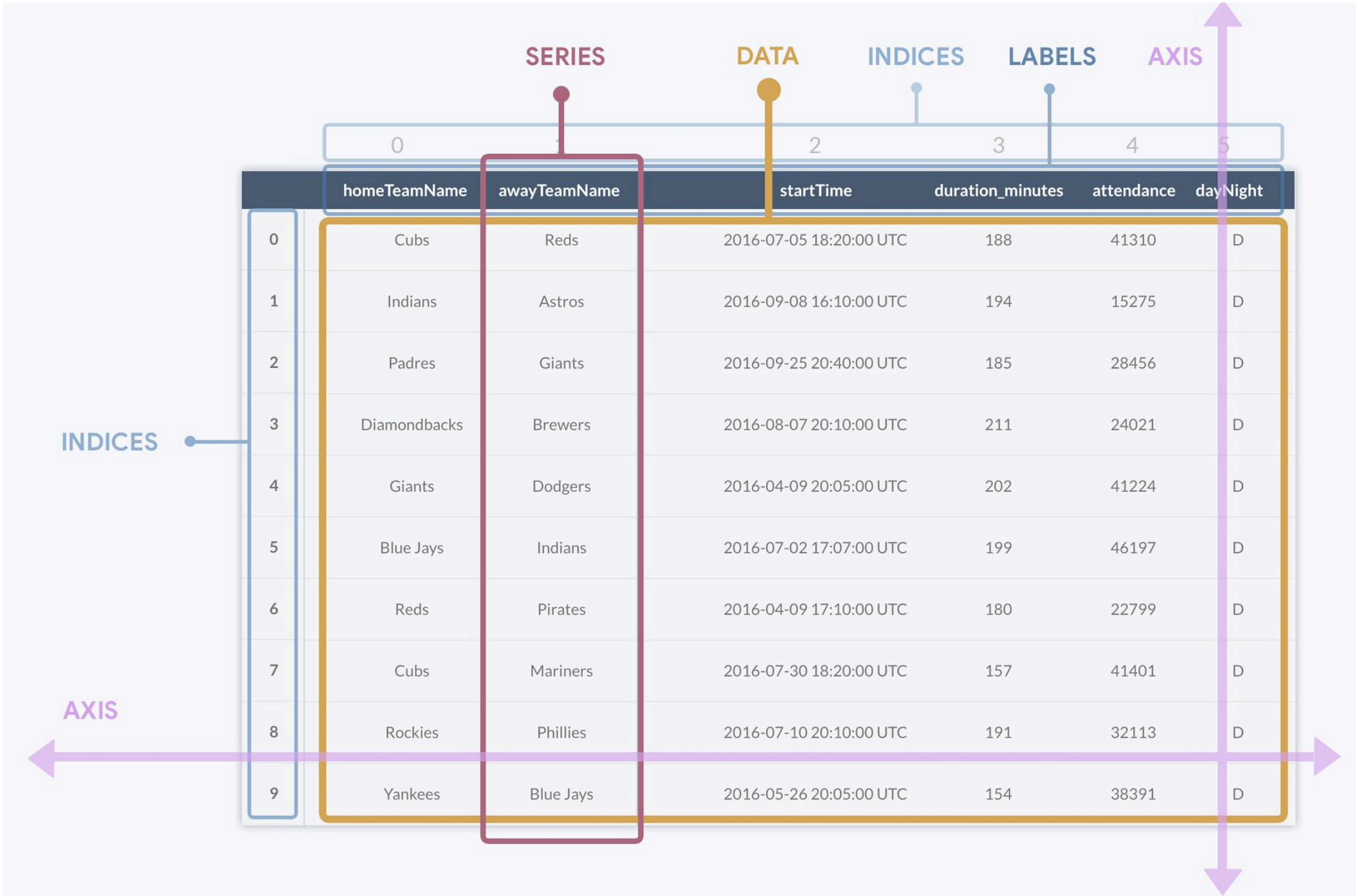
Pandas is a Python library for data manipulation and analysis.

- Provides data structures for efficiently storing and manipulating large datasets
- Allows easy data cleaning, filtering, manipulation, and analysis
- Built-in support for data I/O in a variety of file formats
- A more natural way to display data than list or numpy array
- Many cool and handy functions

Usage

https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf

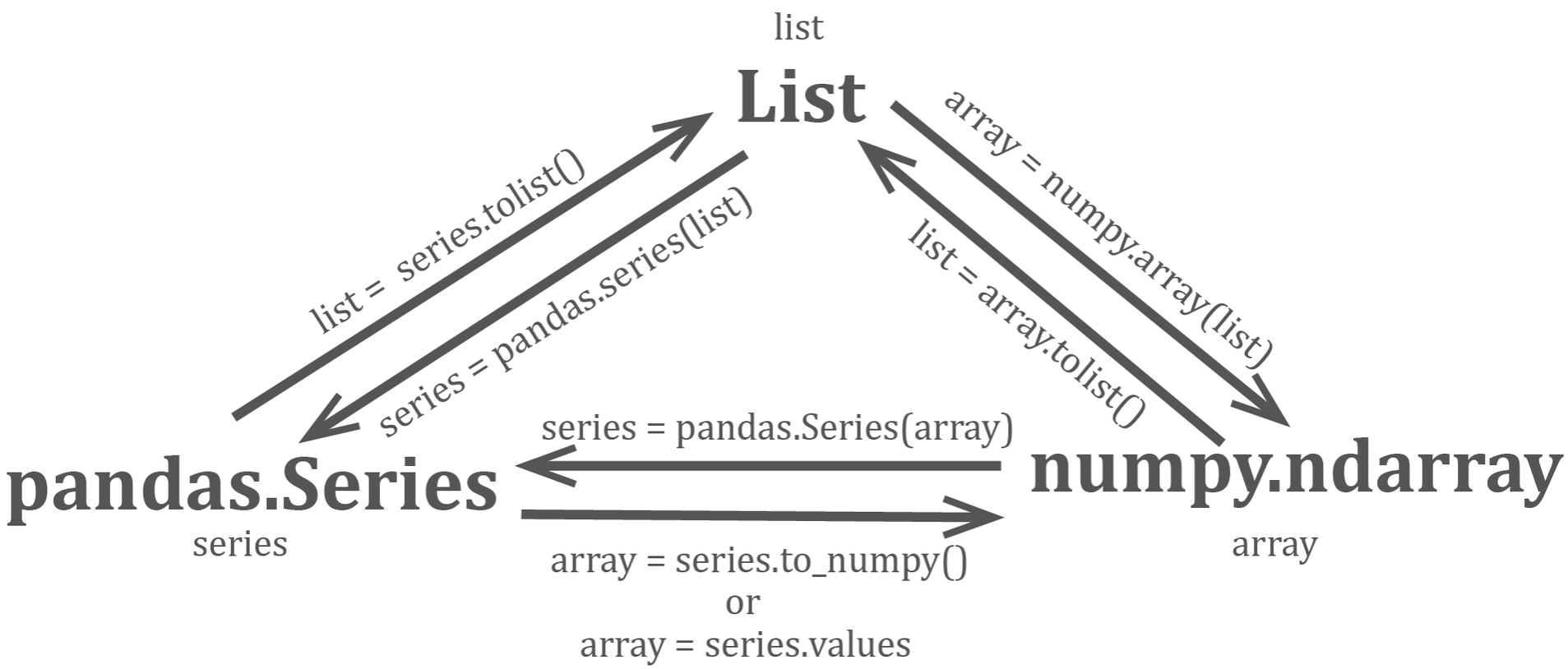
Data Analysis (pandas)



	homeTeamName	awayTeamName	startTime	duration_minutes	attendance	dayNight
0	Cubs	Reds	2016-07-05 18:20:00 UTC	188	41310	D
1	Indians	Astros	2016-09-08 16:10:00 UTC	194	15275	D
2	Padres	Giants	2016-09-25 20:40:00 UTC	185	28456	D
3	Diamondbacks	Brewers	2016-08-07 20:10:00 UTC	211	24021	D
4	Giants	Dodgers	2016-04-09 20:05:00 UTC	202	41224	D
5	Blue Jays	Indians	2016-07-02 17:07:00 UTC	199	46197	D
6	Reds	Pirates	2016-04-09 17:10:00 UTC	180	22799	D
7	Cubs	Mariners	2016-07-30 18:20:00 UTC	157	41401	D
8	Rockies	Phillies	2016-07-10 20:10:00 UTC	191	32113	D
9	Yankees	Blue Jays	2016-05-26 20:05:00 UTC	154	38391	D

Pandas DataFrame, source: https://devopedia.org/images/article/304

Data Analysis (pandas)



Data type conversion, source: https://devopedia.org/images/article/304

Practice pandas on LeetCode!

https://leetcode.com/studyplan/30-days-of-pandas/

Plots and Visualization

Matplotlib and **Seaborn**

- ***Installation***: `pip install matplotlib seaborn`

matplotlib

- A general purpose plotting library

Examples: https://matplotlib.org/stable/gallery/index.html

seaborn

- Built on top of `matplotlib`
- Specialized for statistical graphics.
- Seaborn working with DataFrames.

Examples: https://seaborn.pydata.org/examples/index.html

Import Statements

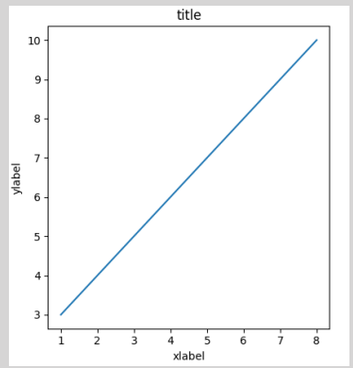
```
import matplotlib.pyplot as plt
import seaborn as sns
```

Example Code

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.title("title")
plt.xlabel("xlabel")
plt.ylabel("ylabel")
plt.show()
```



Machine Learning Algorithms (sklearn)

Scikit-learn

- ***Installation***: `pip install -U scikit-learn`

Description

- Scikit-learn is a machine learning library
- Built on **NumPy**, **SciPy**, and **Matplotlib**
- Designed to be easy to use and efficient.

Source: sklearn official website

Usage

https://scikit-learn.org/stable/user_guide.html
(You can find the source code of ML algorithms here)

```
>>> from sklearn import linear_model
>>> reg = linear_model.LinearRegression()
>>> reg.fit([[0, 0], [1, 1], [2, 2]], [0, 1, 2])
LinearRegression()
>>> reg.coef_
array([0.5, 0.5])
```

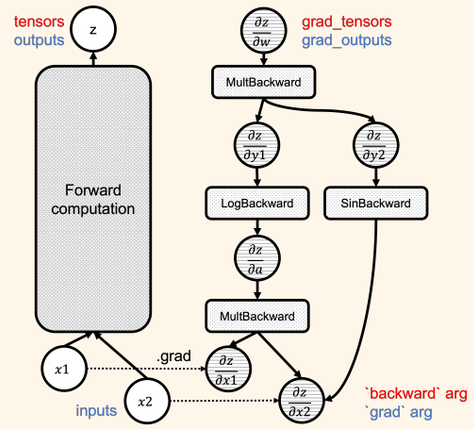
Linear regression example

Table of Contents

DL Platform Comparison

PyTorch

- Dynamic computational graph
- Changes the graph on the fly
- Easier to debug
- Best for development



An example of a computational graph, source: https://pytorch.org

Tensorflow

- Static computational graph
- Must define the entire computation graph before the model can run
- Optimized to make the models run faster
- More suitable for production

Keras

- built on top of other libraries like Tensorflow, Theano and CNTK
- quickly and easily build, train, and evaluate deep learning models with minimal code
- highly modular

PyTorch

Installation

- Version matters!
- Make sure PyTorch version matches with CUDA version.
- Check here for details: https://pytorch.org/get-started/locally/

Core Features

Tensors
`torch.Tensor`

- PyTorch's main data structure (similar to numpy's ndarrays)

Optimizers
`torch.optim`

- Stochastic Gradient Descent (SGD), Adam, RMSProp, etc. available

Neural Networks
`torch.nn`

- PyTorch provides a built-in module for building and training neural networks
- Includes **loss functions**

LR Schedulers
`torch.optim`

- Learning rate schedulers: ExponentialLR, StepLR etc. are available in `optim.lr_scheduler`

PyTorch

Core Features
(contd…)

Autograd
`.grad`

- A PyTorch feature that allows for automatic differentiation of tensors.
- Used to compute gradients for computational graphs

Data Loading Tools
`torch.utils.data`

- A PyTorch feature that allows for automatic differentiation of tensors.
- Used to compute gradients for computational graphs

Great step-by-step tutorial
https://pytorch.org/tutorials/

PyTorch-Lightning

PyTorch-Lightning is a wrapper library built on top of PyTorch

Great for researchers

Lightning AI

Build and train PyTorch models and connect them to the ML lifecycle using Lightning App templates, without handling DIY infrastructure, cost management, scaling, and other headaches.

Lightning Gallery • Key Features • How To Use • Docs • Examples • Community • Contribute • License

python 3.7 | 3.8 | 3.9 | 3.10 pypi package 1.8.6 downloads 38M conda v1.8.1 docker pulls 1.2M codecov 37%

docs passing slack chat License Apache 2.0

Easy to build, train, and evaluate deep learning models
Support for distributed training across multiple GPUs and machines.
Automated logging of training metrics, model architecture and other information.
Automated checkpointing and early stopping.
Support for mixed precision training
Built-in support for common callbacks
…
https://github.com/Lightning-AI/lightning

Table of Contents

Shell Essentials

Python Basics

Python Packages for ML and Visualization

Deep Learning Frameworks

MLOps Platform

MLOps Platform

TensorBoard

- Free
- Unlimited storage
- Developed by the Tensorflow team
- Need port forwarding if used on remote server
- https://www.tensorflow.org/tensorboard/get_started

Weights and Biases (wandb)

- https://wandb.ai/site
- An good example: wandb.ai/brentspell/hifi-gan-bwe

Benefits

Version control
Training monitoring
Find optimal models
Increase reproducibility
Share insights
Visualization
…

```python
# 0. install and import
import wandb

# 1. Start a new run
wandb.init(project="gpt-3")

# 2. Save model inputs and hyperparameters
config = wandb.config
config.learning_rate = 0.01

# 3. Log gradients and model parameters
wandb.watch(model)
for batch_idx, (data, target) in enumerate(train_loader):
    if batch_idx % args.log_interval == 0:
# 4. Log metrics to visualize performance
        wandb.log({"loss": loss})
```

An example of wandb

Thank you!