Lecture 2
Tools for Machine Learning

Baotong Tian

ECE 208/408 - The Art of Machine Learning

1/29/2025

(Special thanks to Melissa Chen for her contributions to the slides)

Table of Contents

1. Shell Essentials

Linux Shell Scripts

- Unix shell is a command-line interpreter [1]. It is a program that takes commands from the keyboard and gives them to the operating system to perform.
- "Terminal" is where you can interact with the shell.
- Unix-like systems: Linux, MacOS. For Windows, see "batch files".

Variations

Bourne shell (sh), GNU Bash (bash), PowerShell (msh), Z shell (zsh), Secure Shell (ssh)

Two ways to use it

Line-by-line in terminal or In a file

Usages

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

Interact with the system:
Cheatsheet: https://github.com/RehanSaeed/Bash-Cheat-Sheet

Package management:
Advanced Package Tool (or APT), the main command-line package manager for Debian and its derivatives.

APT Examples:
$ apt update && sudo apt upgrade
$ apt install xxx
$ apt remove xxx

System and hardware monitor:
CPU: 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

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

2. Python Basics

Why use Python?

Popular candidates: Python, Matlab, R, Java, C, C++

Middle-level or High-level

C: Fast, efficient, portable, but hard to write/understand

Python: Highly abstracted from the computer hardware, easy to understand

Compiled or Interpreted

Java/C++: Fast, protect source code, but can be more difficult to debug

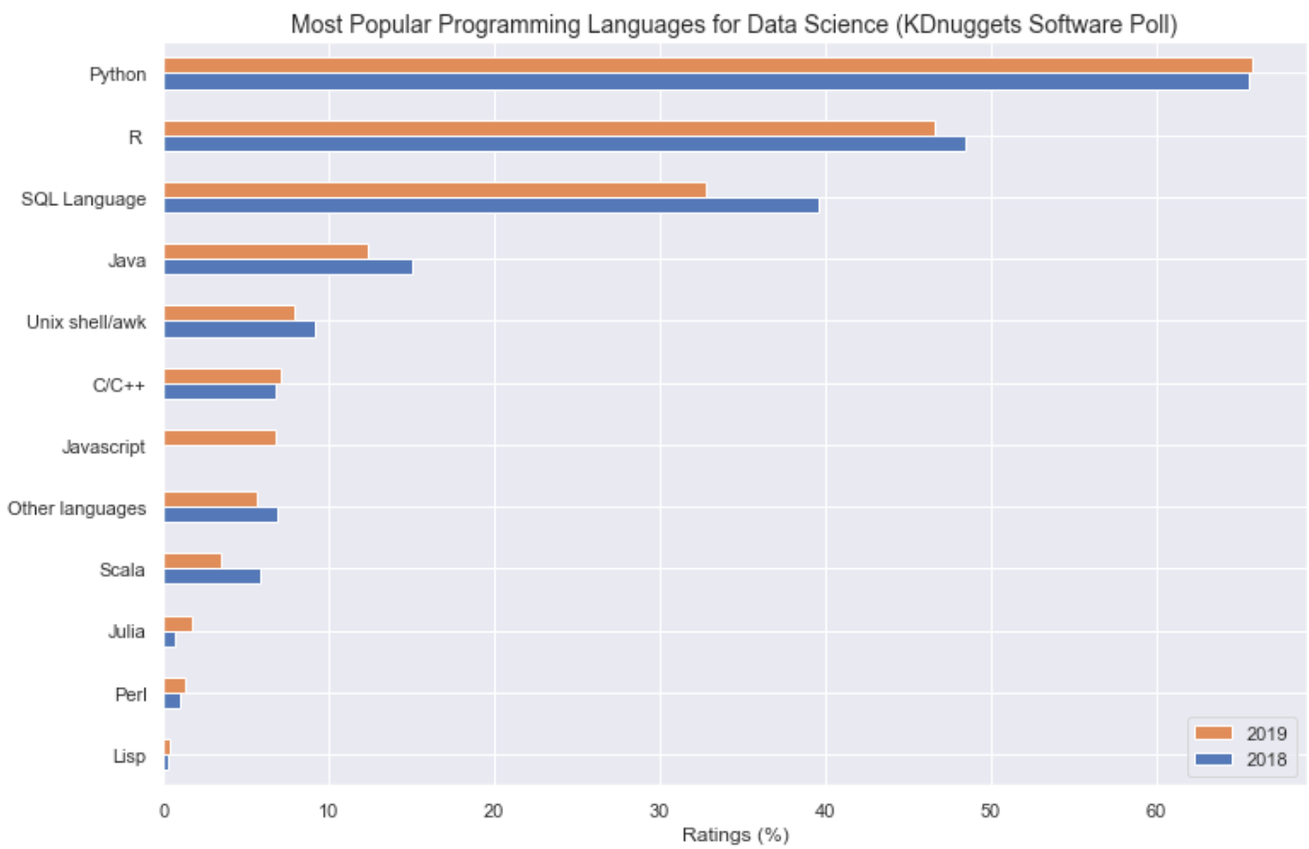
Python/Matlab: Interpreted line-by-line and on-the-fly, flexible, cross-platform

ML Ecosystem and Developer Community

Python has the most active ML developer community

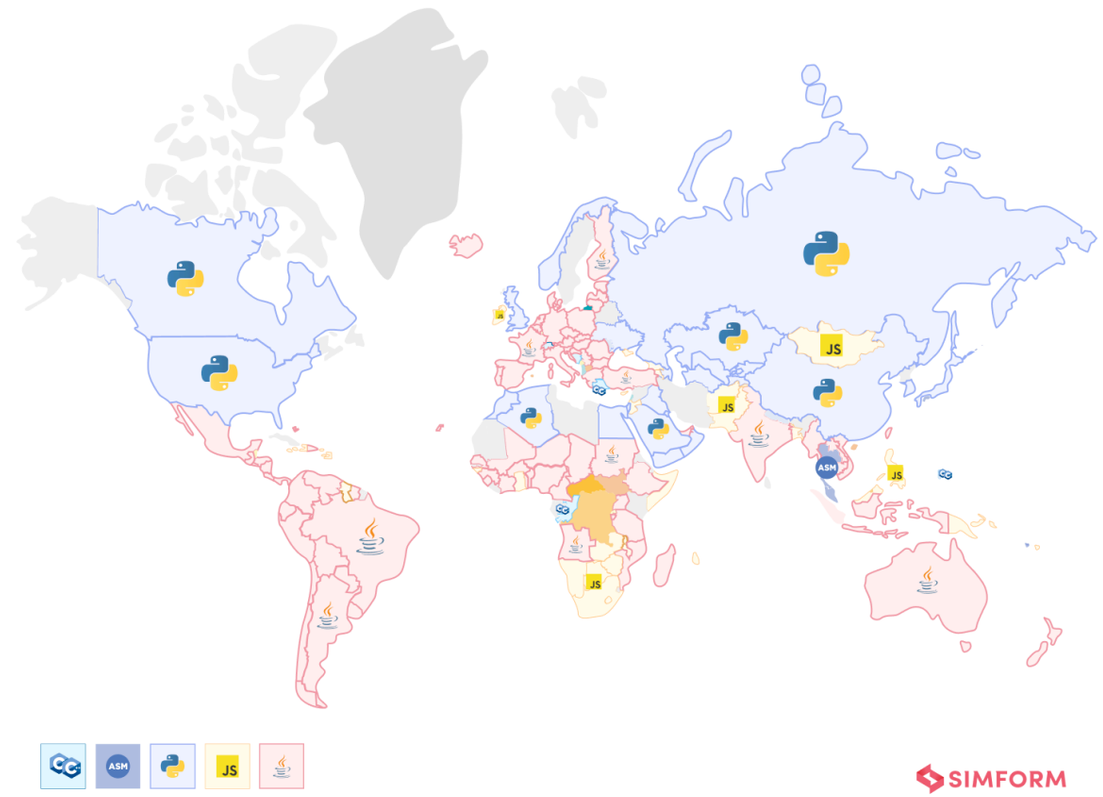
PyTorch, TensorFlow, Keras, …

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.

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

Python Package Management

Pip

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

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
```

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

Python Basics

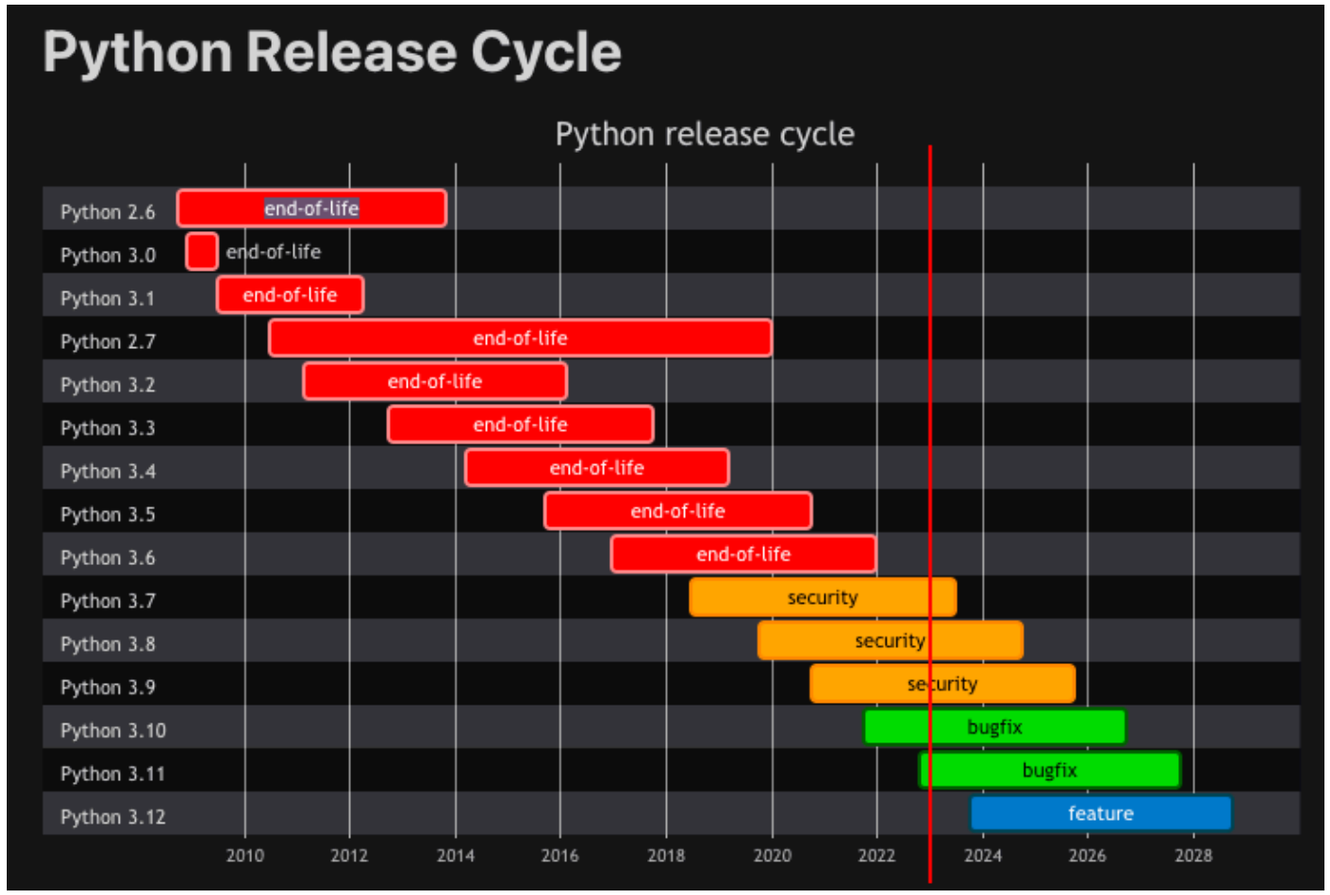
Installation

Download and double click: https://www.python.org/downloads/
Install in shell using apt or homebrew:

```
$ sudo apt-get update
$ sudo apt-get install python3.6
```

Install in Anaconda: conda install python=3.8

Versions



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

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 1/0.
  - Integer (int): Whole numbers without a decimal point. Can be positive, negative, or zero.
  - Float (float): Numbers that contain floating decimal points. 64-bit double-precision.
- Encoding

  UTF-8 (Default), or plain ASCII
- Integer operations:

  Addition +
  Subtraction –
  Multiplication *
  Division /
  Exponents **
- String concatenation:

  3 * 'un' + 'ium'

Python Basics

Data Structures

Lists

- A collection of items that are **ordered** and **changeable**
- Lists might contain items of different types, but usually the items all have the same type

```python
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
```

Dictionaries

- A collection of items that are **unordered**, **changeable** and **indexed**
- Contain a collection of keys, and values associated with them

```python
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
```

Sets

- A collection of items that are **unordered** and **unindexed**
- The elements contained in a set must be unique and **unchangeable**
- Sets seem very similar to lists, but they are very different

```python
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}
```

Tuples

- A collection of items that are **ordered** and **unchangeable**
- Almost the same as List, bur cannot be modified once created

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

Python Basics

Control Flow Tools

```python
# 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)
```

```python
# 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 Debugging Tools?

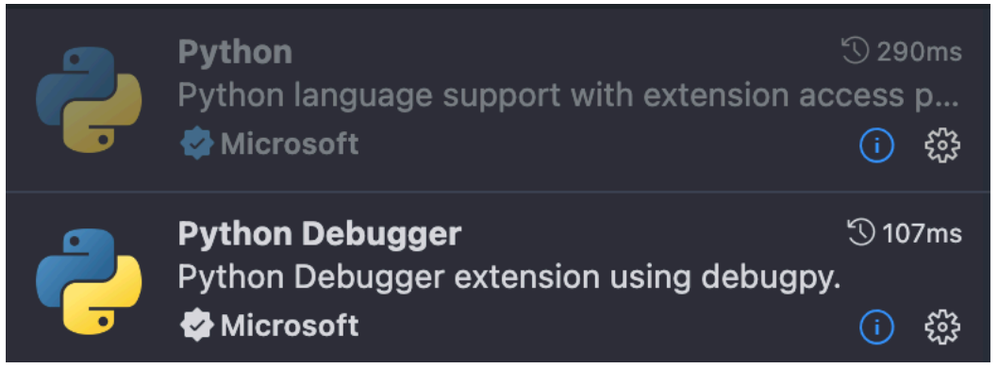
- Debugging is essential for identifying and fixing errors in programming
- Helps improve code reliability, maintainability, and performance

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()
- Advantage: 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)

Debugging With IDE

- VSCode Extension
- Built-in PyCharm debugger



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 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

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.

Installation

PyPI distribution: pip install notebook

Anaconda distribution available

Intro example here:

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

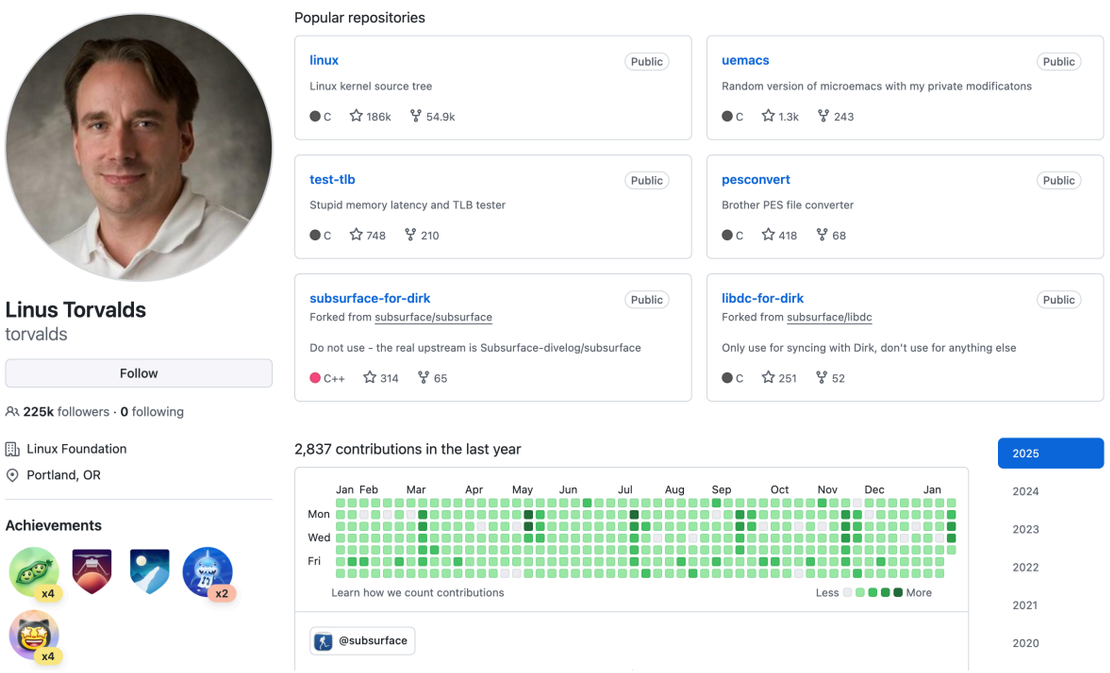
Version Control

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 for Git

- 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

- Init

`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 & 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

3. Python Packages for ML

NumPy

Numerical computing tool

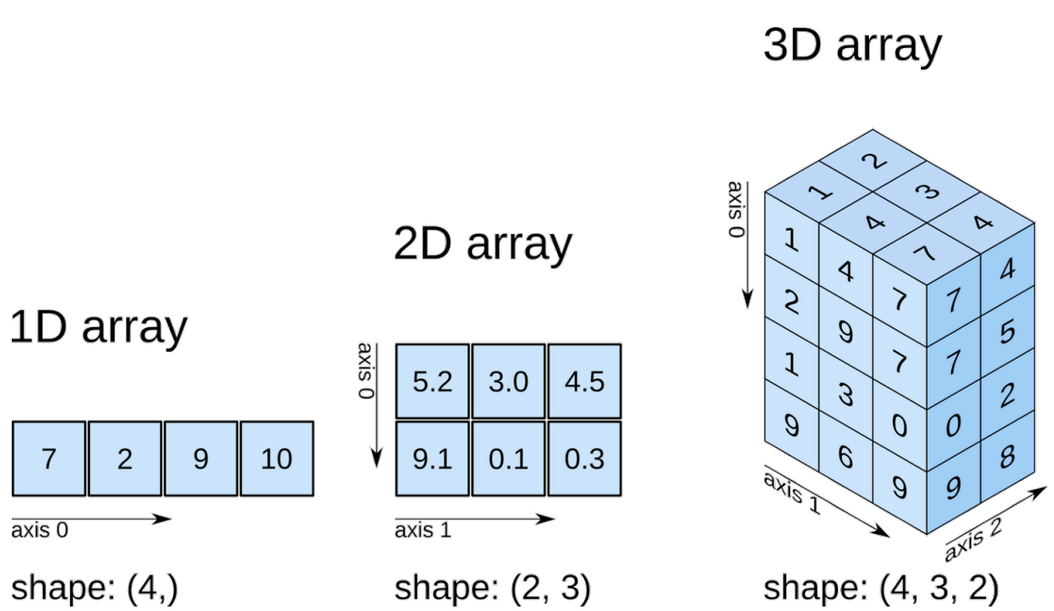
- Fast and versatile
- Mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

Installation

conda install numpy
pip install numpy

Core concepts: numpy.ndarrays

NumPy vectorization, indexing, and broadcasting




```
>>> 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

Pandas

Pandas is a Python library for data manipulation and analysis.

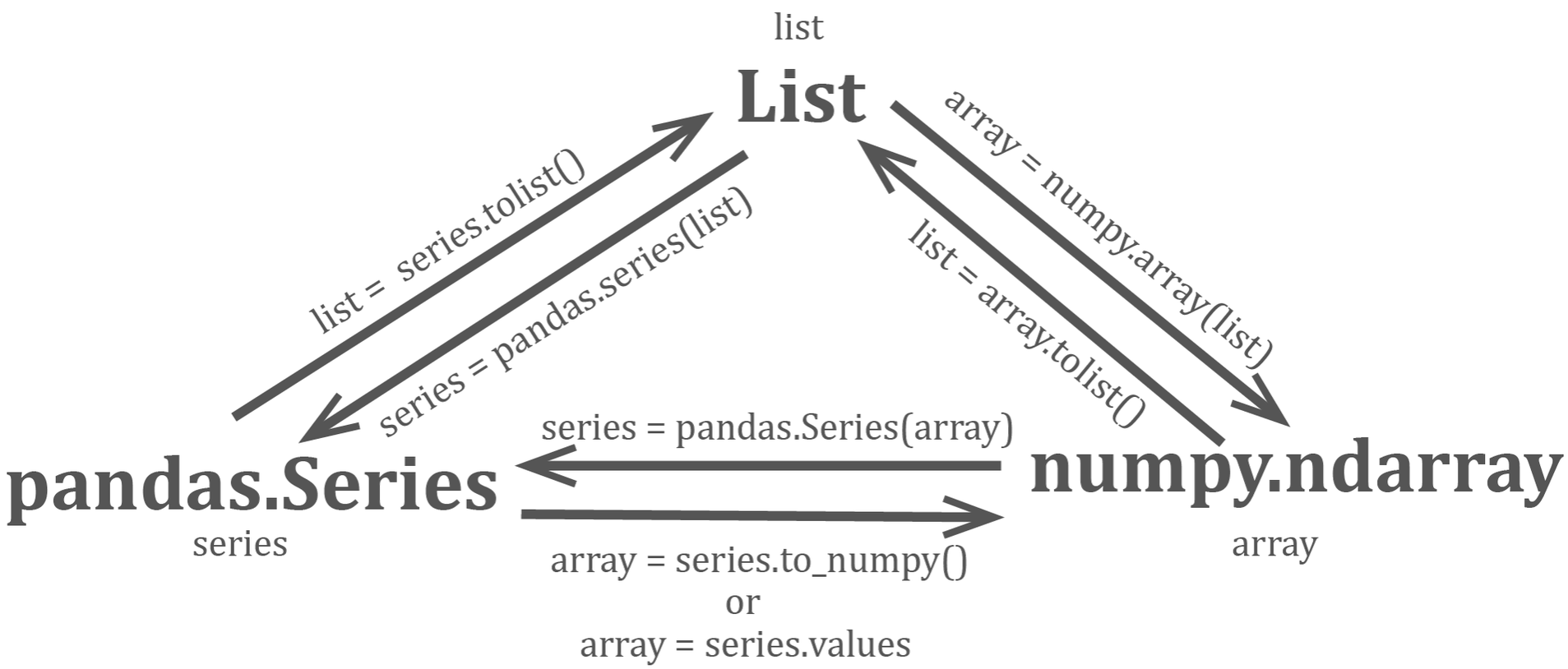
- Provides data structures for efficiently storing and manipulating large datasets
- Allows easy data cleaning, filtering, manipulation, and analysis
- Built-in support for data I/O in a variety of file formats
- A more natural way to display data than list or numpy array
- Many cool and handy functions

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

	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

Pandas



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

Scikit-learn

Scikit-learn

Scikit-learn is a machine learning library built on NumPy, SciPy, and matplotlib, and is designed to be easy to use and efficient.

Installation
Pip: pip3 install -U scikit-learn
Conda version available

Usage:
https://scikit-learn.org/stable/user_guide.html
You can find 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

Source: sklearn official website

Plotting Tools

Matplotlib and Seaborn

Installation: pip

Matplotlib is a general purpose plotting library
Seaborn is built on top of Matplotlib and is specialized for statistical graphics.
Seaborn working with DataFrames.

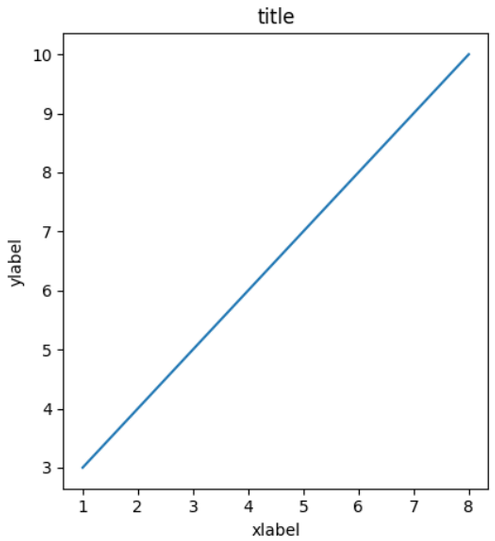
Matplotlib examples: https://matplotlib.org/stable/gallery/index.html
Seaborn examples: https://seaborn.pydata.org/examples/index.html

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

```
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()
```

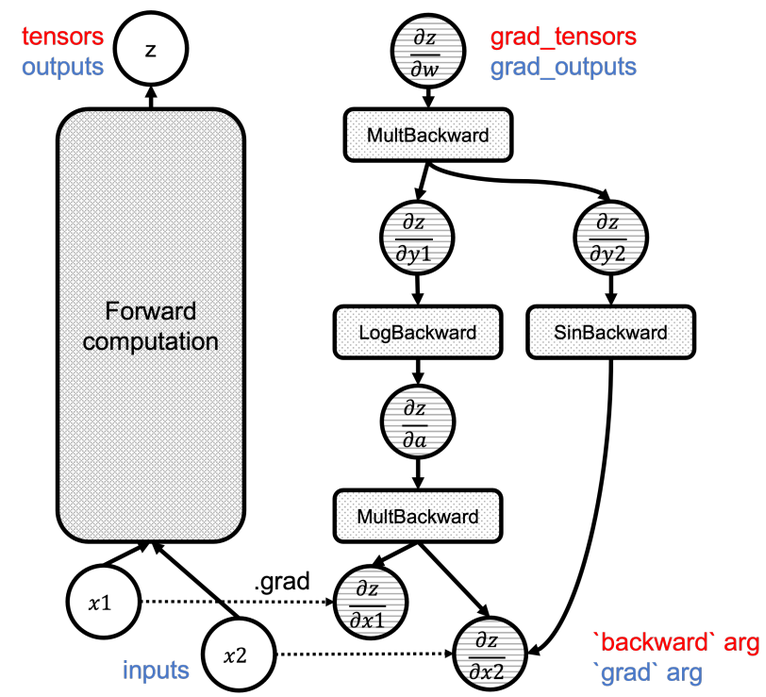


4. Deep Learning Frameworks

DL Platform Comparison

PyTorch

- dynamic computational graph framework
- change 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 concepts

- Tensors: PyTorch's main data structure, similar to numpy's ndarrays
- Autograd: A PyTorch feature that allows for automatic differentiation of tensors. It is used to compute gradients.
- Neural networks: PyTorch provides a built-in module for building and training neural networks in torch.nn.
- Optimizers: SGD, Adam, etc. in torch.optim
- Data loading and preprocessing: torch.utils.data

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

5. MLOps Platform

MLOps Platform

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

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 & Biases

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

```
# 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