# CS229
# Python & Numpy

Angelica Sun

# How does python relate to other languages?

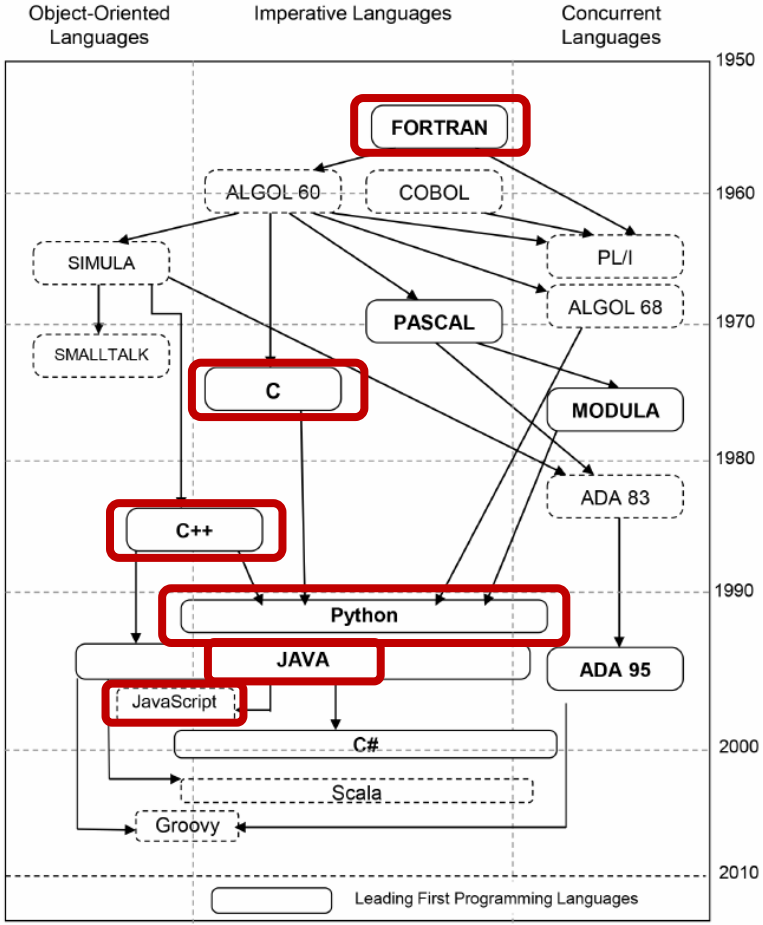
Python 2.0 released in 2000

(Python 2.7 “end-of-life” in 2020)

Python 3.0 released in 2008

(Python 3.6+ for CS 229)

- Interpreted, like MATLAB
- Object-oriented
- Dynamically-typed




https://www.researchgate.net/figure/Genealogy-of-Programming-Languages-doi101371-journalpone0088941g001_fig1_260447599

# Before you start

Always use **conda** for environment management

Create a new environment

```
conda create -n cs229 python=3.9
```

Create an environment (from configuration)

```
conda env create -f environment.yml
```

Activate an environment after creation

```
conda activate cs229
```

List existing environments

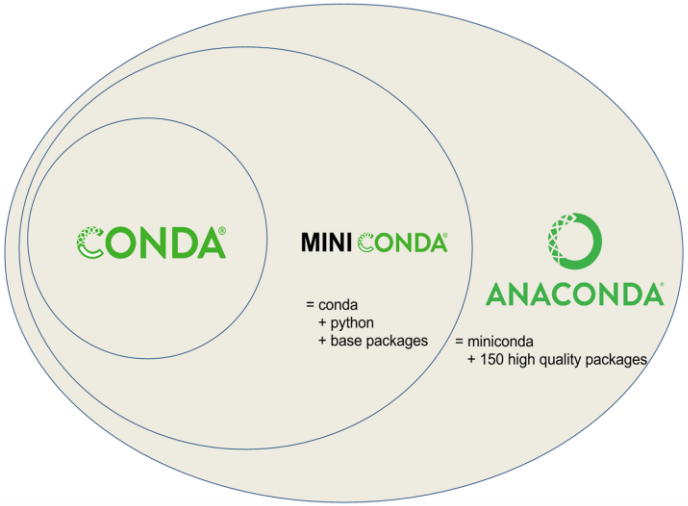
```
conda env list
```

Install a package in current environment

```
conda install PACKAGENAME
```
(or pip)

More commands:

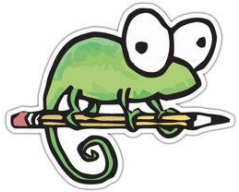
https://conda.io/projects/conda/en/latest/_downloads/843d9e0198f2a193a3484886fa28163c/conda-cheatsheet.pdf



# Notepad is not your friend ...

Get a text editor/IDE

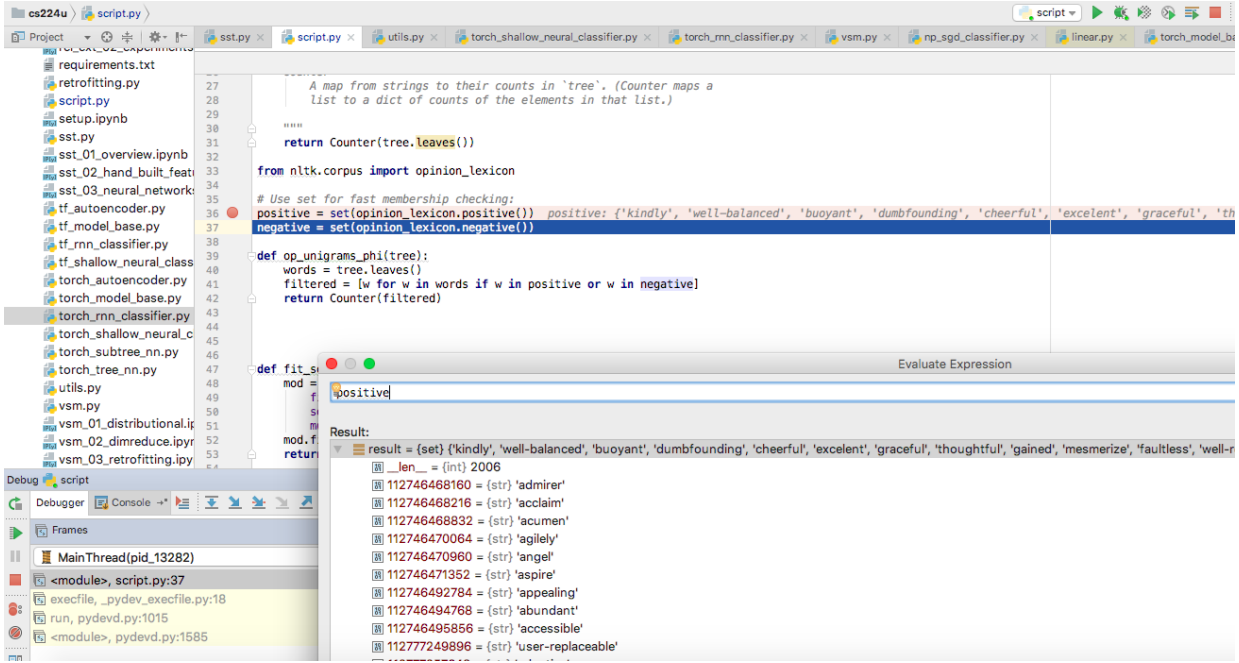
- PyCharm (IDE)
- Visual Studio Code (IDE??)
- Sublime Text (IDE??)
- Notepad ++/gedit
- Vim (for Linux)


# To make you more prepared

## PyCharm

- Great debugger
- Proper project management




FYI, professional version free for students: https://www.jetbrains.com/student/

# To make you more prepared

## Using PyCharm with conda

Settings

Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Project: featuremap.py
Python Interpreter
Project Structure
Build, Execution, Deployment
Languages & Frameworks
Tools

Project: featuremap.py › Python Interpreter  For current project

Python Interpreter: Python 3.8 C:\Users\angel\anaconda3\python.exe

| Package | Version | Latest version |
|---|---|---|
| _anaconda_depends | 2020.07 | |
| _ipyw_jlab_nb_ext_conf | 0.1.0 | |
| alabaster | 0.7.12 | |
| alembic | 1.5.5 | |
| anaconda | custom | |
| anaconda-client | 1.7.2 | |
| anaconda-navigator | 1.9.12 | |
| anaconda-project | 0.8.4 | |
| aniso8601 | 8.1.1 | |
| appdirs | 1.4.4 | |
| argh | 0.26.2 | |
| asn1crypto | 1.3.0 | |
| astroid | 2.4.2 | |
| astropy | 4.0.1.post1 | |
| atomicwrites | 1.4.0 | |
| attrs | 19.3.0 | |
| audioread | 2.1.9 | |
| autopep8 | 1.5.3 | |
| babel | 2.8.0 | |
| backcall | 0.2.0 | |
| backports | 1.0 | |
| backports.functools_lru_cache | 1.6.1 | |
| backports.shutil_get_terminal_size | 1.0.0 | |
| backports.tempfile | 1.0 | |
| backports.weakref | 1.0.post1 | |

OK  Cancel  Apply

Python Interpreters

Python 3.8 C:\Users\angel\anaconda3\python.exe
Python 3.8 (pythonProject) C:\Users\angel\anaconda3\envs\pythonProject\python.exe
Python 2.7 (AA274A_HW2) C:\Users\angel\anaconda3\envs\AA274A_HW2\python.exe
Python 2.7 (python2.7) C:\Users\angel\anaconda3\envs\python2.7\python.exe

OK  Cancel

Add Python Interpreter

Virtualenv Environment
Conda Environment
System Interpreter
Pipenv Environment

New environment
Location: E:\stanford\cs229\cs229-content\Exercises\src\featurem
Base interpreter: C:\Users\angel\AppData\Local\Microsoft\Wind
Inherit global site-packages
Make available to all projects

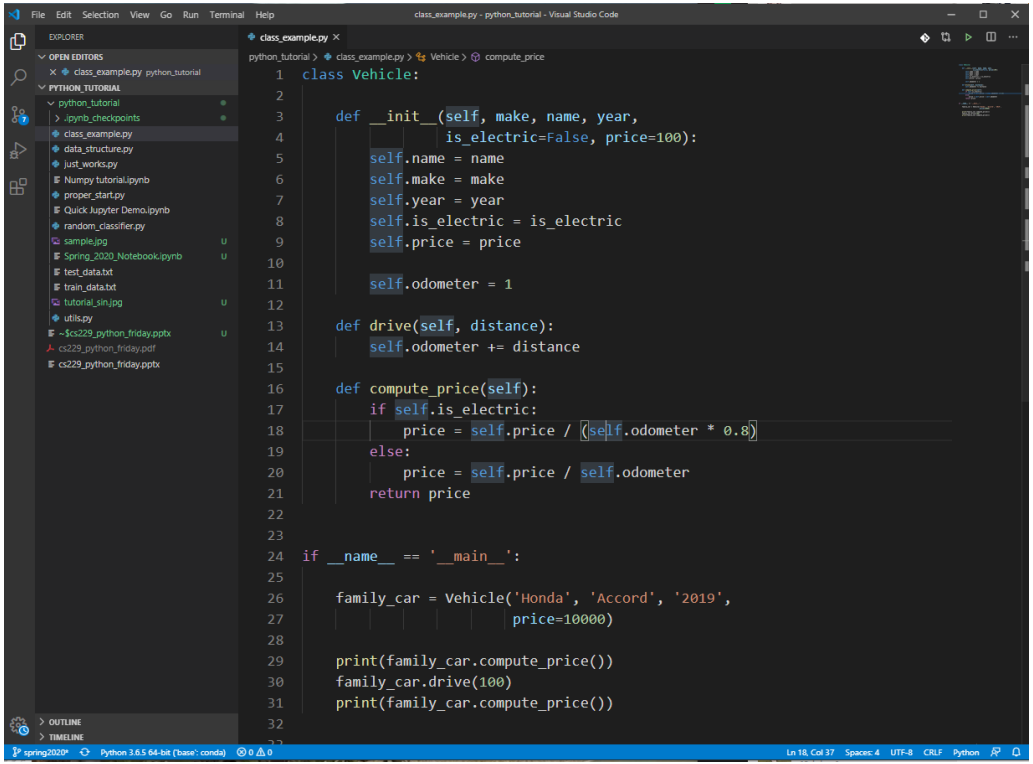
Existing environment
Interpreter: <No interpreter>
Make available to all projects

OK  Cancel

# To make you more prepared

## Visual Studio Code

- Light weight
- Wide variety of plugins to enable support for all languages
- Better UI



# Python

- featuremap.py
- main.py
- utils.py
- External Libraries
- Scratches and Consoles

```python
import numpy                        # import package
from matplotlib import pyplot as plt    # import from package and rename
from utils import *                 # import from local file

var = 10                            # variables don't need type delcaration
anothervar = "Hello"                # codes written at indentation level 0 always get executed
print("I always get executed!")     # even when imported

def print_hi(name):                 # define a function
    print(f'Hi, {name}')

class Foo:                          # define a class
    def __init__(self, x):          # constructor
        self.x = x                  # use first formal parameter (named "self" by convention) for self reference

    def printX(self):               # functions are instance by default
        print(self.x)

    @classmethod                    # use decorator to declare class methods
    def printHello(self):
        print("hello")

class Bar(Foo):                     # inherit a class
    pass

if __name__ == '__main__':          # main function (doesn't execute on import)
    print_hi('CS229')               # call function
    functionInUtils()               # call imported function
    obj = Bar(3)                    # create object
    obj.PrintX()                    # call function using object
```

# Live Demo in Jupyter Notebook

- Links:
- Notebook:
- http://cs229.stanford.edu/notes2021spring/notes2021spring/cs229-python-review-code.ipynb
- PDF Version: http://cs229.stanford.edu/notes2021spring/notes2021spring/python-review-code.pdf

# Your friend for debugging

| Python Command | Description |
|---|---|
| array.shape | Get shape of numpy array |
| array.dtype | Check data type of array (for precision, for weird behavior) |
| type(stuff) | Get type of a variable |
| import pdb; pdb.set_trace() | Set a breakpoint (https://docs.python.org/3/library/pdb.html) |
| print(f'My name is {name}') | Easy way to construct a message |

Good luck on your HW/Project!

Questions?