# INTRODUCTION TO DATA SCIENCE

JOSÉ MANUEL CALDERÓN TRILLA

(SLIDES BY JOHN P. DICKERSON)

**Lecture #2 – 06/02/2021**

**CMSC320**
**Weekdays**
**2:00pm – 3:25pm**
**(… or anytime on the Internet)**

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

## ANNOUNCEMENTS

**Register on Discord:**

- some have registered already ♡
- The rest have not 💔

**If you were on Discord, you'd know …**

- Project 0 is out! It is "due" Friday evening.
- Link: **https://github.com/cmsc320/summer202/tree/main/project0**

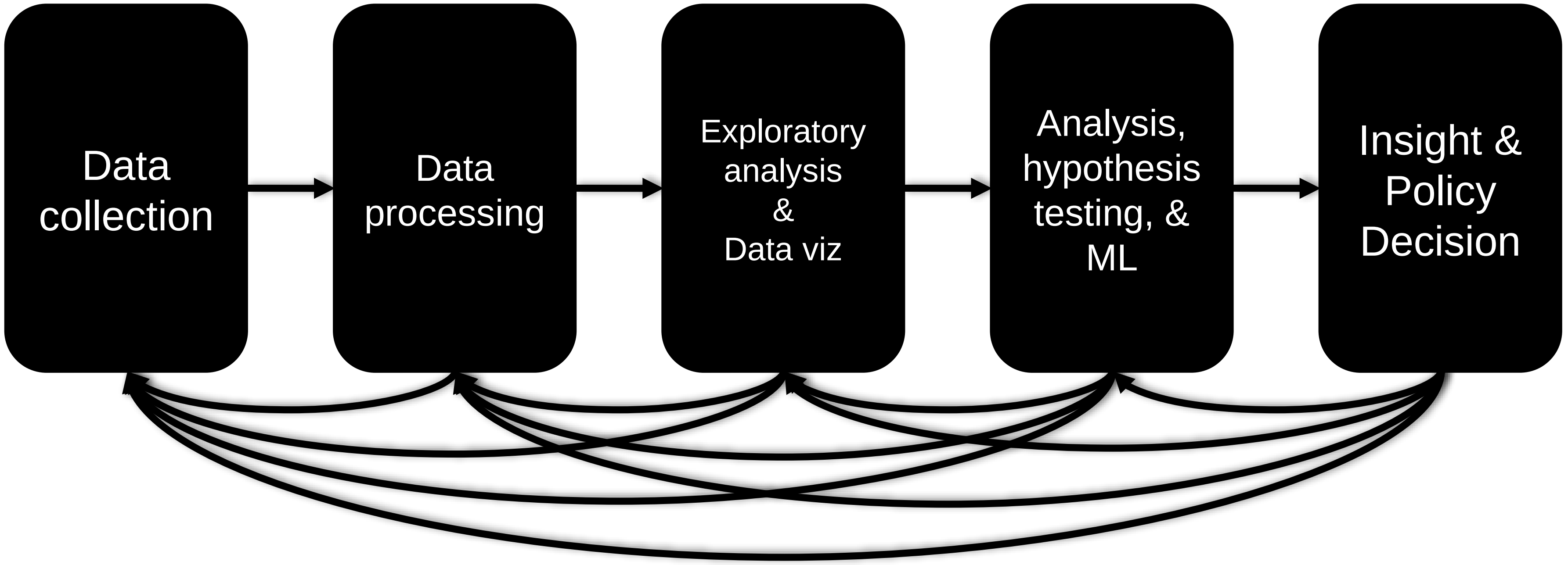
**We've also linked some reading for the week!**

- First quiz will be due Monday at noon.
- Quiz will go up Friday

*UP NEXT …*

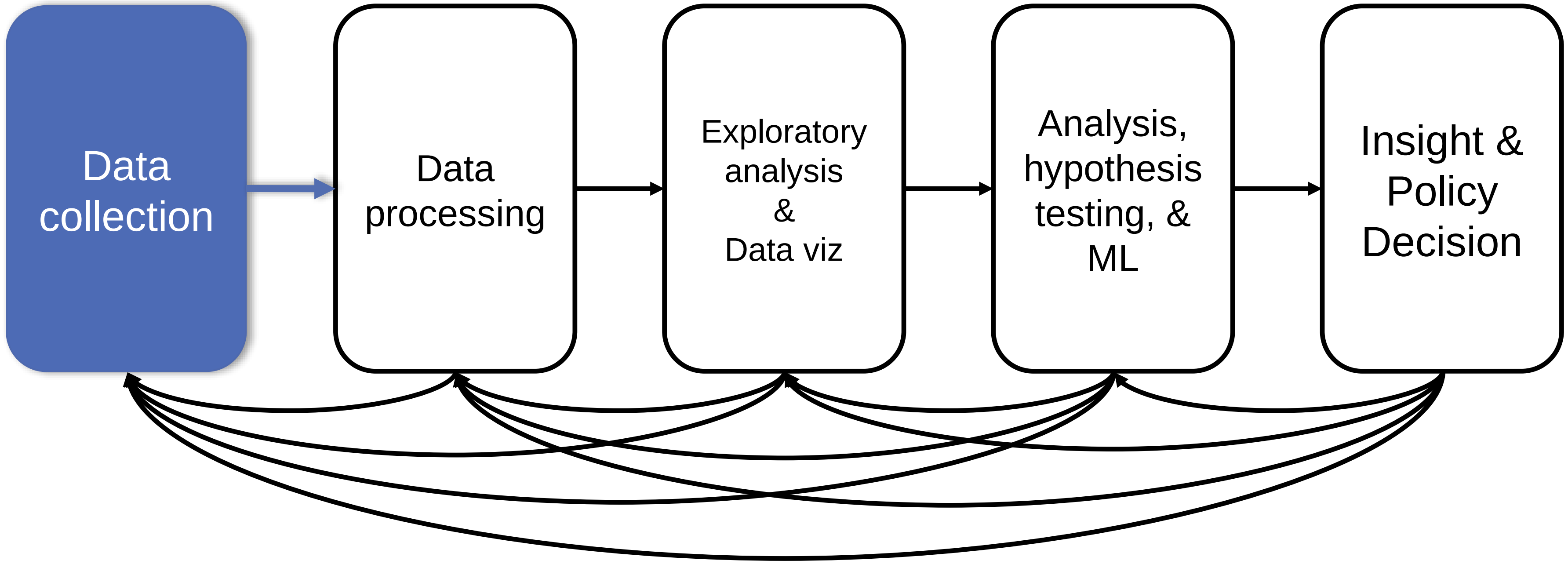
## SCRAPING DATA WITH PYTHON

# THE DATA LIFECYCLE



Data collection
Data processing
Exploratory analysis & Data viz
Analysis, hypothesis testing, & ML
Insight & Policy Decision

## (THE REST OF) TODAY'S LECTURE


Data collection
Data processing
Exploratory analysis & Data viz
Analysis, hypothesis testing, & ML
Insight & Policy Decision

## BUT FIRST, SNAKES!

**Python is an interpreted, dynamically-typed, high-level, garbage-collected, object-oriented-functional-imperative, and widely used scripting language.**

- **Interpreted:** instructions executed without being compiled into (virtual) machine instructions*
- **Dynamically-typed:** verifies type safety at runtime
- **High-level:** abstracted away from the raw metal and kernel
- **Garbage-collected:** memory management is automated
- **OOFI:** you can do bits of OO, F, and I programming

**Not the point of this class!**

- Python is fast (developer time), intuitive, and used in industry!

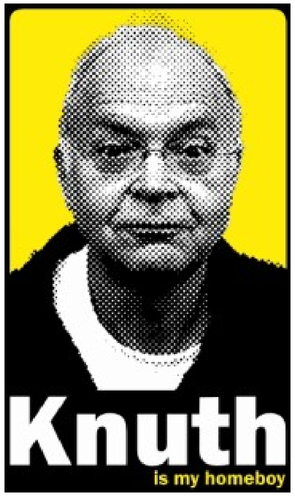
*you can compile Python source, but it's not required

## THE ZEN OF PYTHON

- **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.**

Thanks: SDSMT ACM/LUG

# LITERATE PROGRAMMING



**Literate code contains in one document:**

- the source code;
- text explanation of the code; and
- the end result of running the code.

**Basic idea: present code in the order that logic and flow of human thoughts demand, not the machine-needed ordering**

- Necessary for data science!
- Many choices made need textual explanation, ditto results.

**Stuff you'll be using in Project 0 (and beyond)!**

IP[y]: IPython Interactive Computing

jupyter

## JUPYTER PROJECT

**Started as iPython Notebooks, a web-based frontend to the iPython Shell**

- Notebook functionality separated out a few years ago
- Now supports over 40 languages/kernels
- Notebooks can be shared easily
- Can leverage big data tools like Spark

**Apache Zeppelin:**

- https://www.linkedin.com/pulse/comprehensive-comparison-jupyter-vs-zeppelin-hoc-q-phan-mba-

Several others including RStudio (specific to R)

# 10-MINUTE PYTHON PRIMER

**Define a function:**

```
def my_func(x, y):
    if x > y:
        return x
    else:
        return y
```

**Python is whitespace-delimited**

**Define a function that returns a tuple:**

```
def my_func(x, y):
    return (x-1, y+2)

(a, b) = my_func(1, 2)
```

```
a = 0; b = 4
```

## USEFUL BUILT-IN FUNCTIONS: COUNTING AND ITERATING

**`len`: returns the number of items of an enumerable object**

```
len( [‘c’, ‘m’, ‘s’, ‘c’, 3, 2, 0] )
```

```
7
```

**`range`: returns an iterable object**

```
list( range(10) )
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

**`enumerate`: returns iterable tuple (index, element) of a list**

```
enumerate( [“311”, “320”, “330”] )
```

```
[(0, “311”), (1, “320”), (2, “330”)]
```

**https://docs.python.org/3/library/functions.html**

# USEFUL BUILT-IN FUNCTIONS: MAP AND FILTER

**`map`: apply a function to a sequence or iterable**

```
arr = [1, 2, 3, 4, 5]
map(lambda x: x**2, arr)
```

```
[1, 4, 9, 16, 25]
```

**`filter`: returns a list* of elements for which a predicate is true**

```
arr = [1, 2, 3, 4, 5, 6, 7]
filter(lambda x: x % 2 == 0, arr)
```

```
[2, 4, 6]
```

**We'll go over in much greater depth with pandas/numpy.**

**in Python 3, returns Iterable*

## PYTHONIC PROGRAMMING

**Basic iteration over an array in Java:**

```
int[] arr = new int[10];
for(int idx=0; idx<arr.length; ++idx) {
   System.out.println( arr[idx] );
}
```

**Direct translation into Python:**

```
idx = 0
while idx < len(arr):
   print( arr[idx] ); idx += 1
```

**A more "Pythonic" way of iterating:**

```
for element in arr:
   print( element )
```

## LIST COMPREHENSIONS

**Construct sets like a mathematician!**

- $P = \{ 1, 2, 4, 8, 16, \ldots, 2^{16} \}$
- $E = \{ x \mid x \text{ in } \mathbb{N} \text{ and } x \text{ is odd and } x < 1000 \}$

**Construct lists like a mathematician who codes!**

```
P = [ 2**x for x in range(17) ]
```

```
E = [ x for x in range(1000) if x % 2 != 0 ]
```

**Very similar to `map`, but:**

- **You'll see these way more than `map` in the wild**
- **Many people consider `map`/`filter` not "pythonic"**
- **They can perform differently (`map` is "lazier")**

## EXCEPTIONS

**Syntactically correct statement throws an exception:**

- `tweepy` (Python Twitter API) returns “Rate limit exceeded”
- sqlite (a file-based database) returns `IntegrityError`

```
print('Python', python_version())

try:
   cause_a_NameError
except NameError as err:
   print(err, '-> some extra text')
```

# PYTHON 2 VS 3

**Python 3 is intentionally backwards incompatible**

- (But not *that* incompatible)

**Biggest changes that matter for us:**

- `print “statement”` ✉ `print(“function”)`
- `1/2 = 0` ✉ `1/2 = 0.5` and `1//2 = 0`
- ASCII `str` default ✉ default Unicode

**Namespace ambiguity fixed:**

```
i = 1
[i for i in range(5)]
print(i)   # ????????
```

## TO ANY CURMUDGEONS …

**If you're going to use Python 2 anyway, use the `_future_` module:**

- Python 3 introduces features that will throw runtime errors in Python 2 (e.g., `with` statements)
- `_future_` module incrementally brings 3 functionality into 2
- https://docs.python.org/2/library/__future__.html

```
from _future_ import division
from _future_ import print_function
from _future_ import please_just_use_python_3
```

## SO, HOW DOES IMPORT WORK?

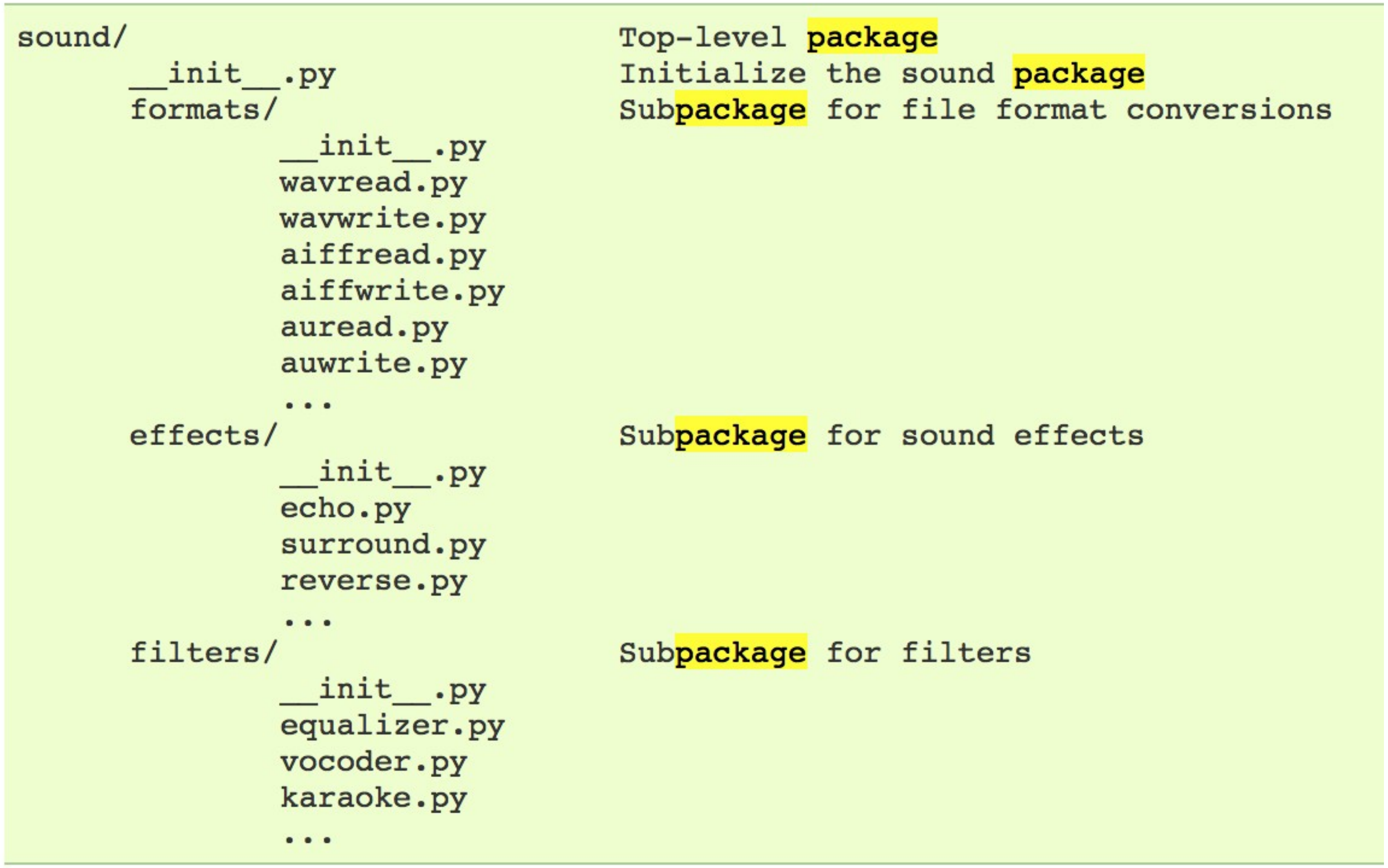
**Python code is stored in module – simply put, a file full of Python code**

**A package is a directory (tree) full of modules that also contains a file called `__init.py__`**

- Packages let you structure Python's module namespace
- E.g., `X.Y` is a submodule Y in a package named X

**For one module to gain access to code in another module, it must import it**

## EXAMPLE



```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

```
# Load (sub)module sound.effects.echo
import sound.effects.echo
# Must use full name to reference echo functions
sound.effects.echo.echofilter(input, output, delay=0.7)
```

https://docs.python.org/2/tutorial/modules.html

# EXAMPLE

```python
# Load (sub)module sound.effects.echo
import sound.effects.echo
# Must use full name to reference echo functions
sound.effects.echo.echofilter(input, output, delay=0.7)
```

```python
# Load (sub)module sound.effects.echo
from sound.effects import echo
# No longer need the package prefix for functions in echo
echo.echofilter(input, output, delay=0.7)
```

```python
# Load a specific function directly
from sound.effects.echo import echofilter
# Can now use that function with no prefix
echofilter(input, output, delay=0.7)
```

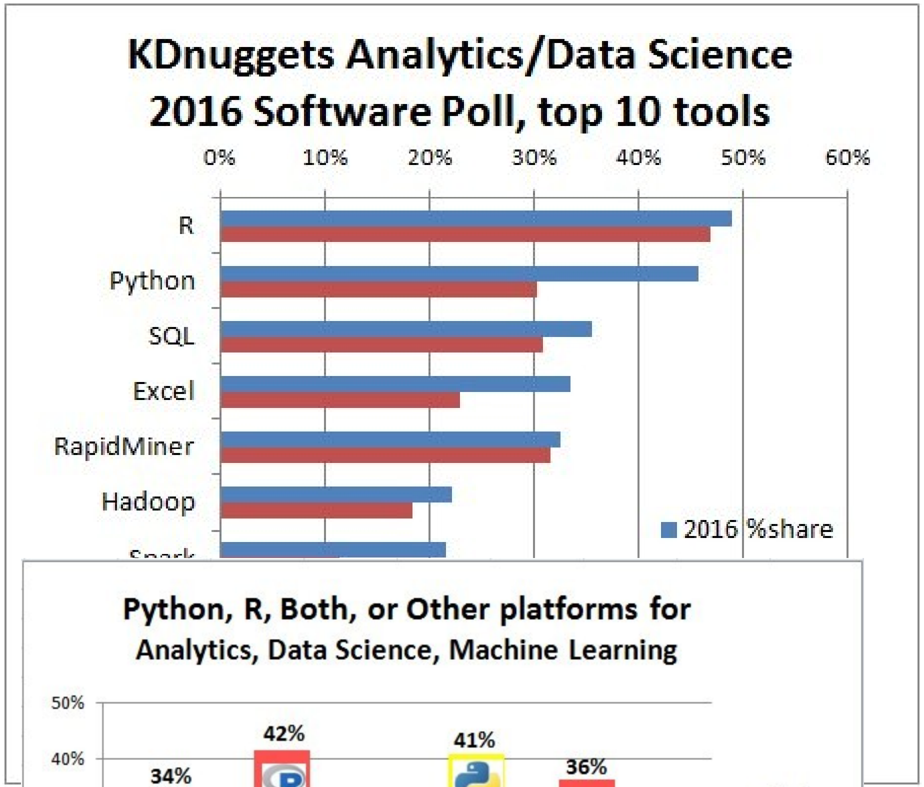
*https://docs.python.org/2/tutorial/modules.html*

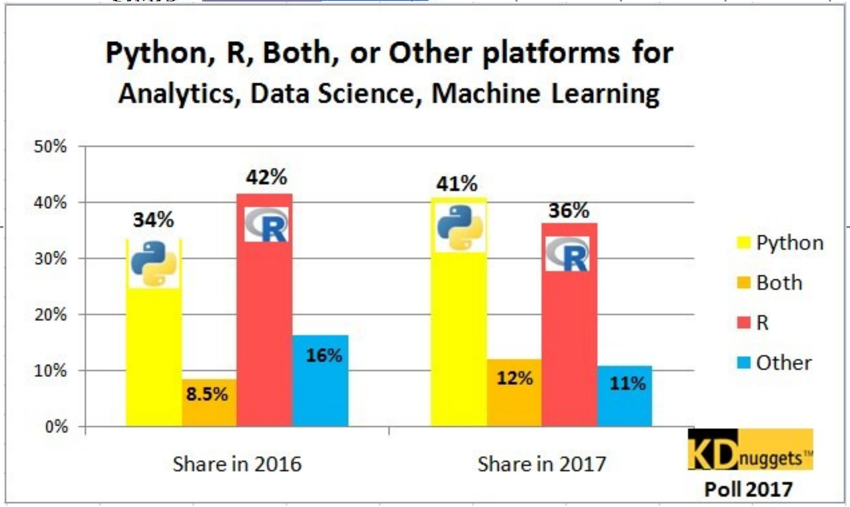
# PYTHON VS R (FOR DATA SCIENTISTS)

**There is no right answer here!**

- **Python is a "full" programming language – easier to integrate with systems in the field**
- **R has a more mature set of pure stats libraries …**
- **… but Python is catching up quickly …**
- **… and is already ahead specifically for ML.**

**You will see Python more in the tech industry.**





## EXTRA RESOURCES

**Plenty of tutorials on the web:**

- https://www.learnpython.org/

**Work through Project 0, which will take you through some baby steps with Python and the Pandas library:**

- (We'll also post some more readings soon.)

**Come (virtually!) hang out at office hours:**

- All office hours will be on the website/Piazza by early next week.
- Will have coverage MTWThF.