Tools & Python

Nicholas Mattei, *Tulane University*

CMPS3660 – Introduction to Data Science – Fall 2019

https://rebrand.ly/TUDataScience

Many Thanks
Slides based off Introduction to Data Science from John P. Dickerson - https://cmsc320.github.io/
Some examples taken from *Data Science* by John D. Kelleher and Brendan Tierney, MIT Press.

Announcements

- Dr. Mattei's Office Hours will be:
  - Tuesday 1400 – 1500 (going possibly later)
  - Thursday 1600 – 1700 (going possibly later)
- Arie is here!

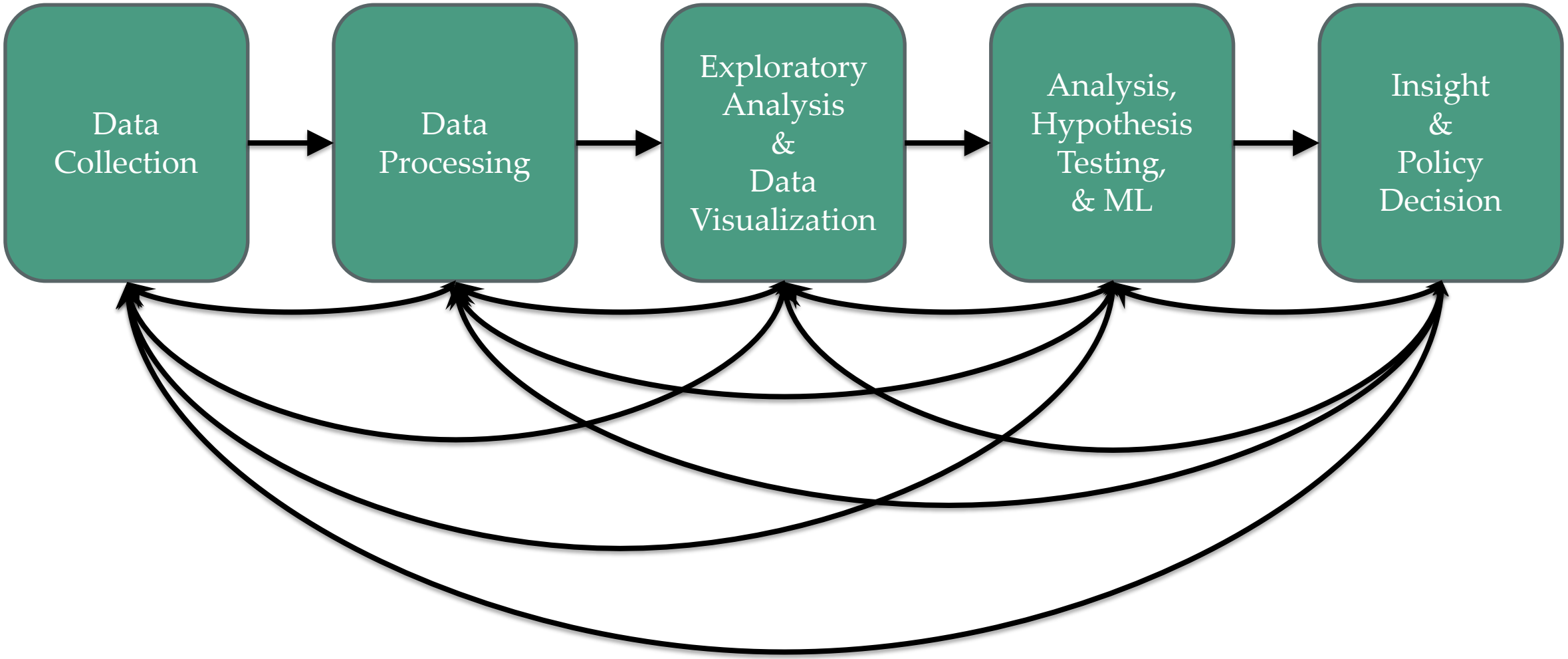
- We now have 34 People in the course.
  - If you are not formally enrolled and want to be come see me after class.

- Both Project0 and Questions1 are posted.
  - https://github.com/TulaneIntroDataScience/fall2019/tree/master/project0
  - Quick overview on how to notebook…

- Please complete Project0 before class on 9/5 – want to do in class lab work that day!

Today

The Data LifeCycle



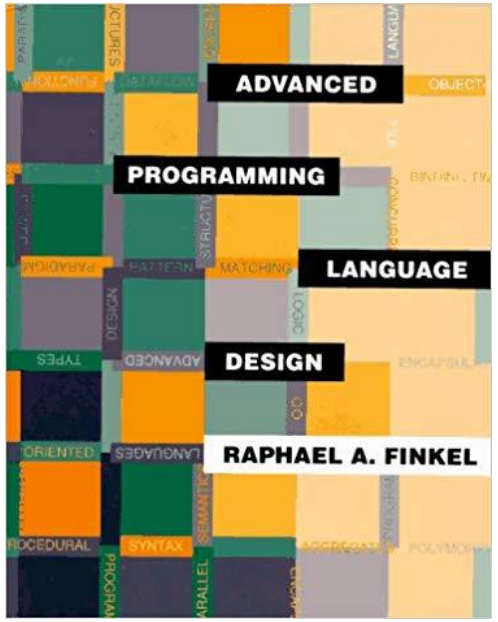
But first, snakes!

- 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
    - OO = Object Oriented (you can make objects)
    - F = Functional (everything is a function, stateless, like LISP)
    - I = Imperative (i.e., procedural)
- 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

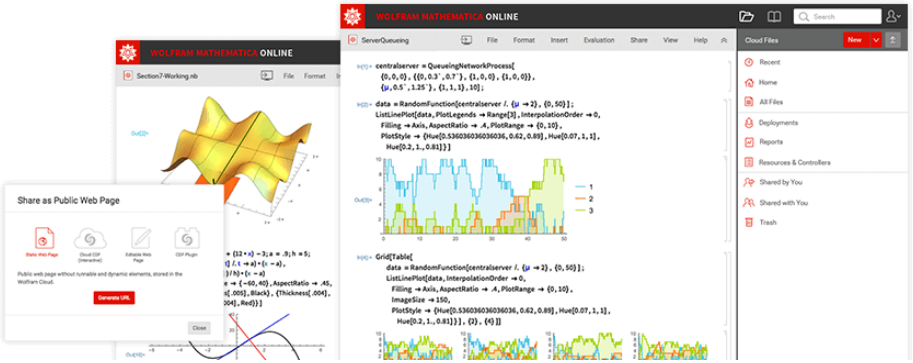
- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules …
  - … although practicality 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

10-Minute Python primer

- Define a function:

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

- 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 Build-In Functions

- `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 Fucntions: Map and Filter

Note!

There is a problem here for Python3!

- `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 as the syntax is a little different.

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")

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)  # ????????`
  - `Prints "4" in Python 2 and "1" in Python 3 (narrow scope)`

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` statement)
  - `_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 braces`, `from _future_ import must_use_python_3`

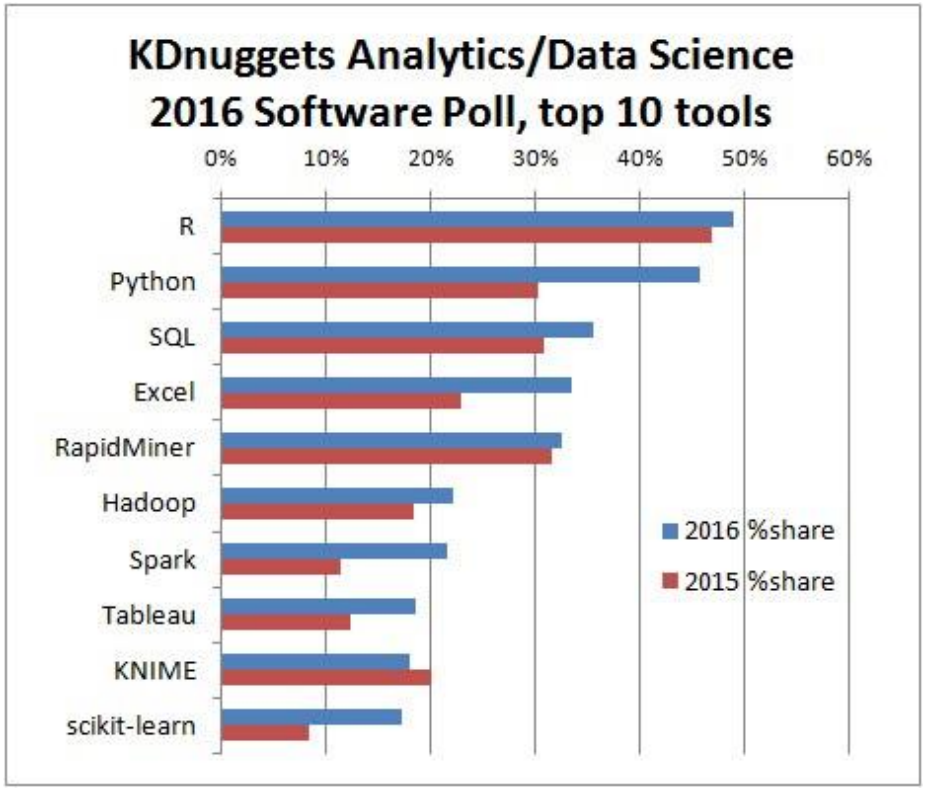
In CMPS 3660 this is wrong.

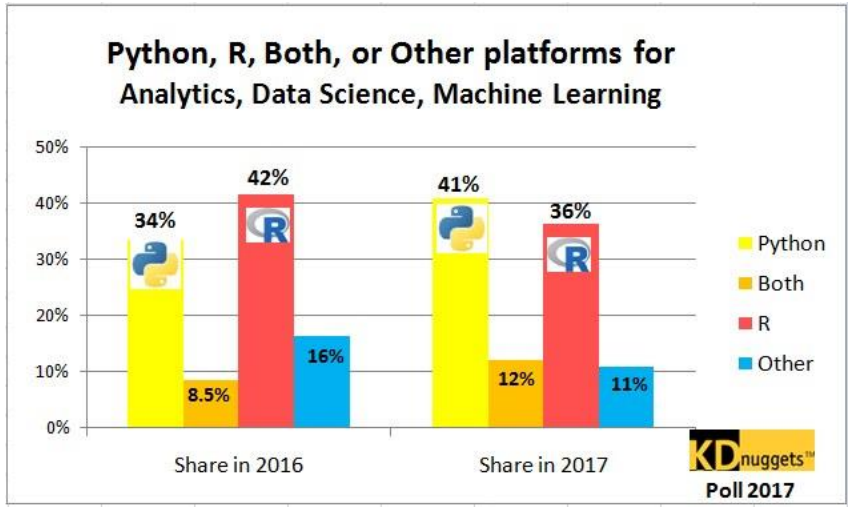
If your code does not run in Python 3, it is wrong.

I'm in charge!

Python v. 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/

- Go look at the Notebook that we made for class today!
  - Link TBD.

- Work through Project 0, which will take you through some baby steps with Python and Docker.
  - https://github.com/TulaneIntroDataScience/fall2019/tree/master/project0