INTRODUCTION TO DATA SCIENCE

JOHN P DICKERSON

Lecture #2 – 08/29/2019

CMSC320
Tuesdays & Thursdays
5:00pm – 6:15pm

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

ANNOUNCEMENTS

Register on Piazza: piazza.com/umd/fall2019/cmsc320

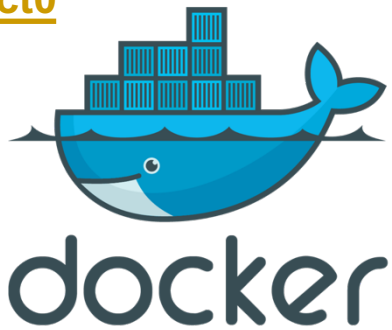
- 210 have registered already ♡
- 88 have not registered yet 💔

If you were on Piazza, you'd know …

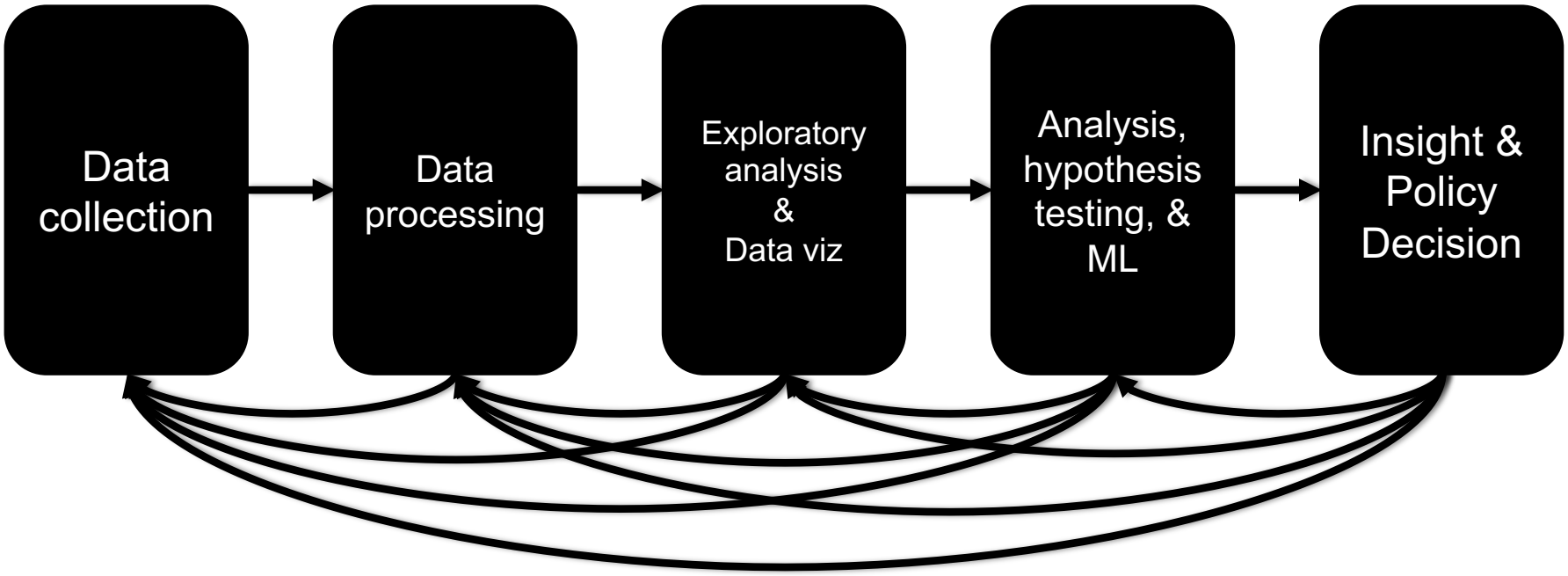
- Project 0 is out! It is "due" next Wednesday evening.
- Link: https://github.com/cmsc320/fall2019/tree/master/project0

We've also linked some reading for the week!

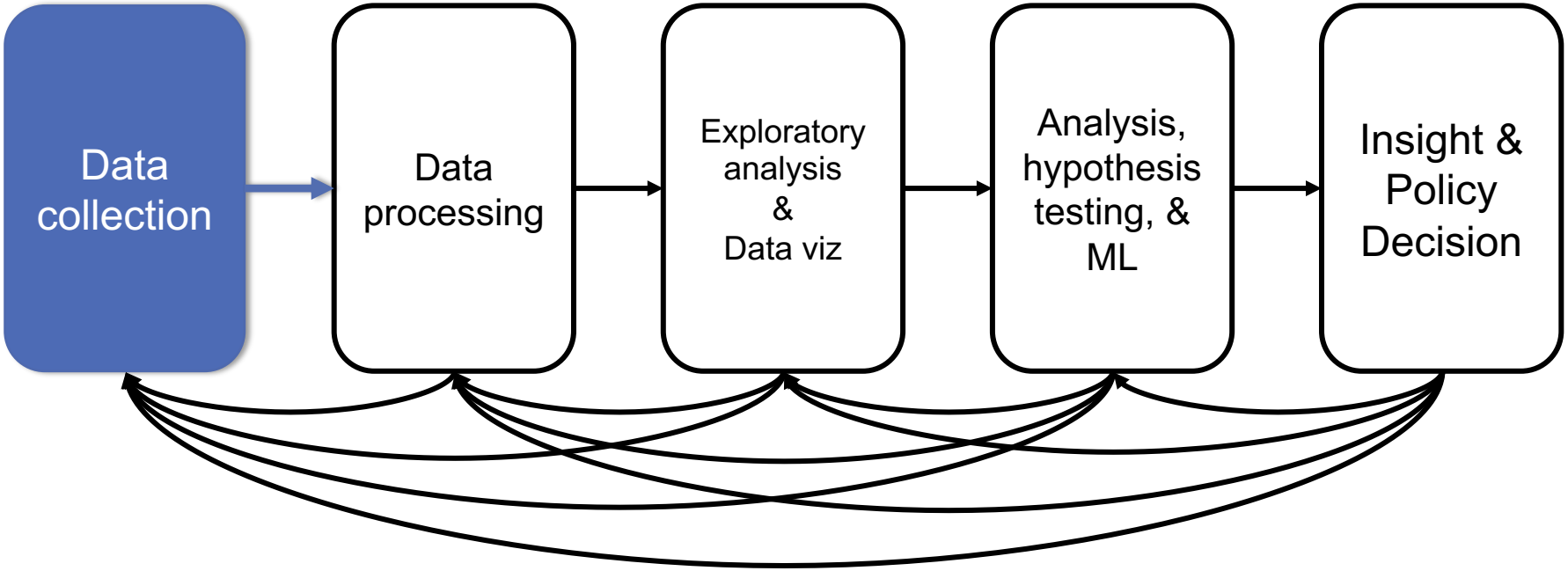
- First quiz will be due Thursday at noon.
- (Quiz should be up on ELMS now.)



THE DATA LIFECYCLE



TODAY'S LECTURE



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

```
# 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)
```

```
# 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)
```

```
# 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)
```

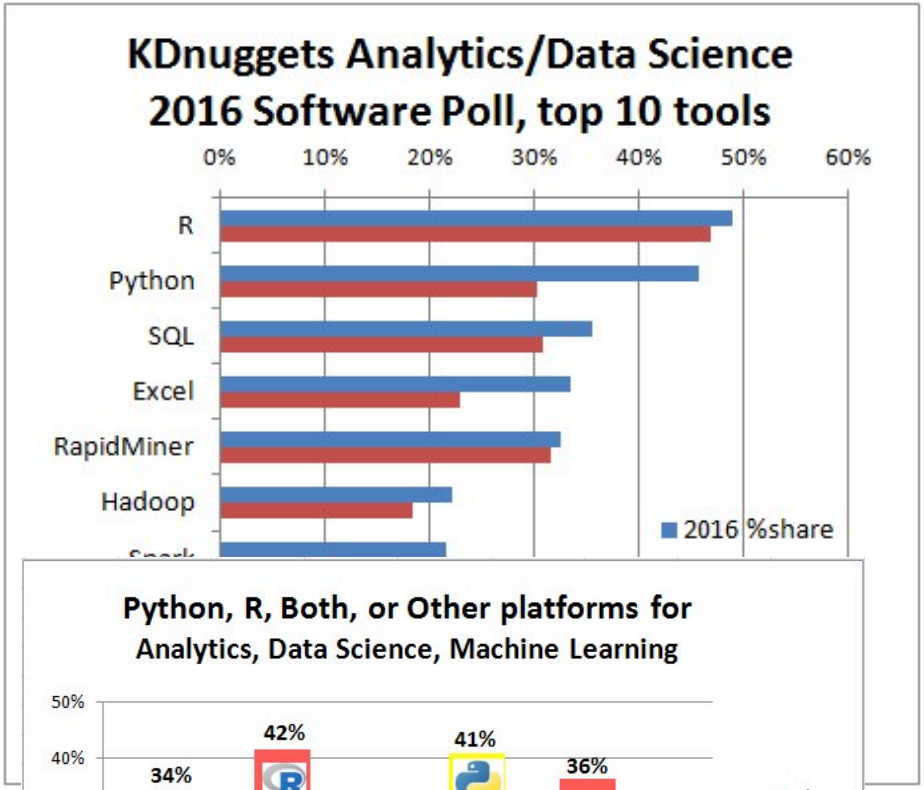
https://docs.python.org/2/tutorial/modules.html

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/

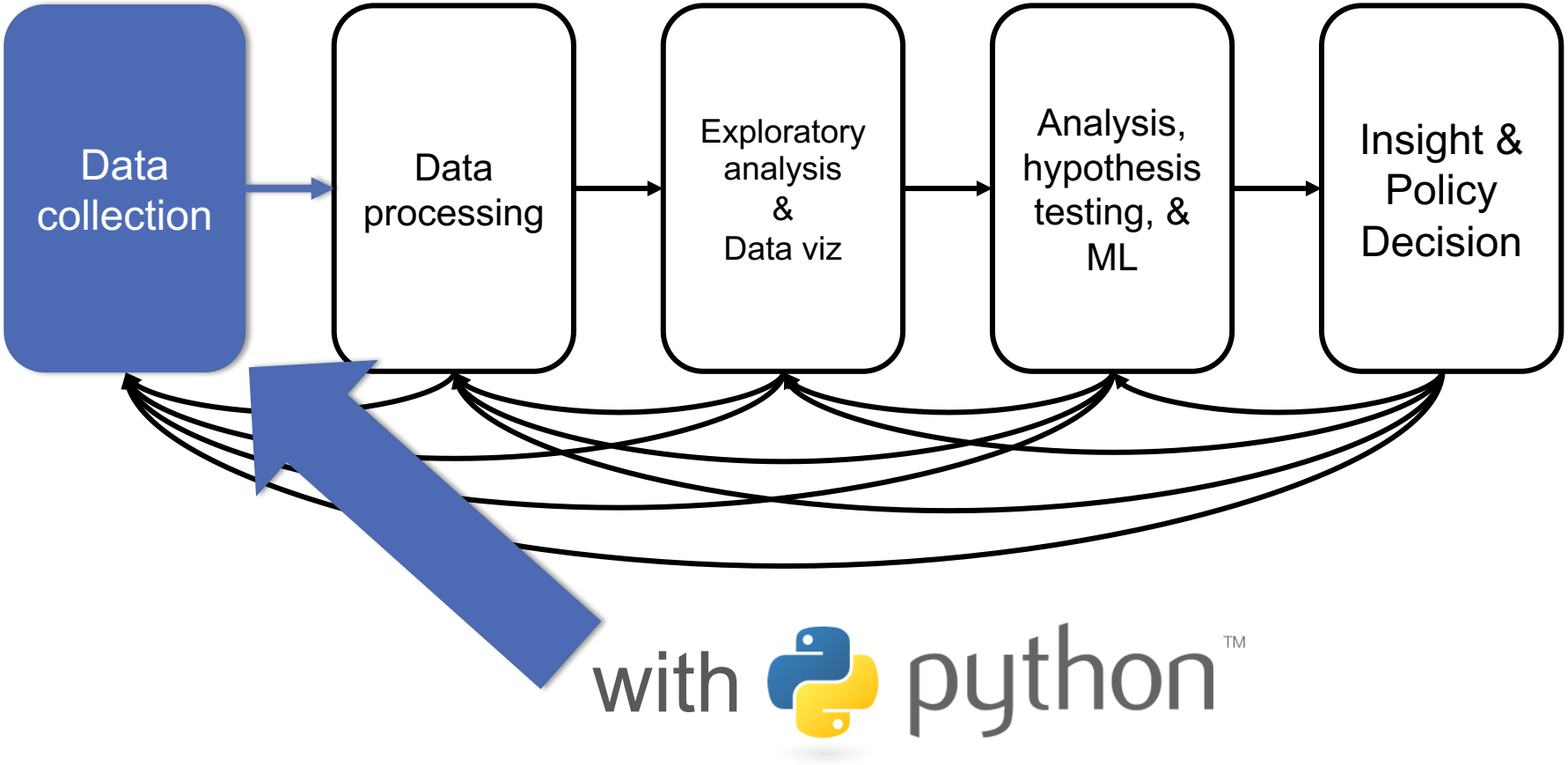
Work through Project 0, which will take you through some baby steps with Python and the Pandas library:

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

Come hang out at office hours (or chat with me privately)

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

TODAY'S LECTURE



Thanks: Zico Kolter's 15-388, Amol Deshpande, Nick Mattei

Google

data

All | News | Images | Books | Videos | More | Settings | Tools

analysis | science | icon | visualization | infographic | cloud

Data - Wikipedia
en.wikipedia.org

Data Digest: Predictive Analytics and ...
tdwi.org

What Is Big Data? | SAS US
sas.com

Data - Wikipedia
en.wikipedia.org

W3C Data Activity - Building the Web of ...
w3.org

Helping MIT faculty ...

Data-
rathe

WHAT IS THIS "DATA"?

TABULAR DATA

Quick teaser. We'll go into greater depth when discussing **tidy data.**

Data is an abstraction of some real world entity.

- Also called: instance, example, record, object, case, individual.

Each of these entities is described by a set of features.

- Sometimes called variables, features, attributes, …

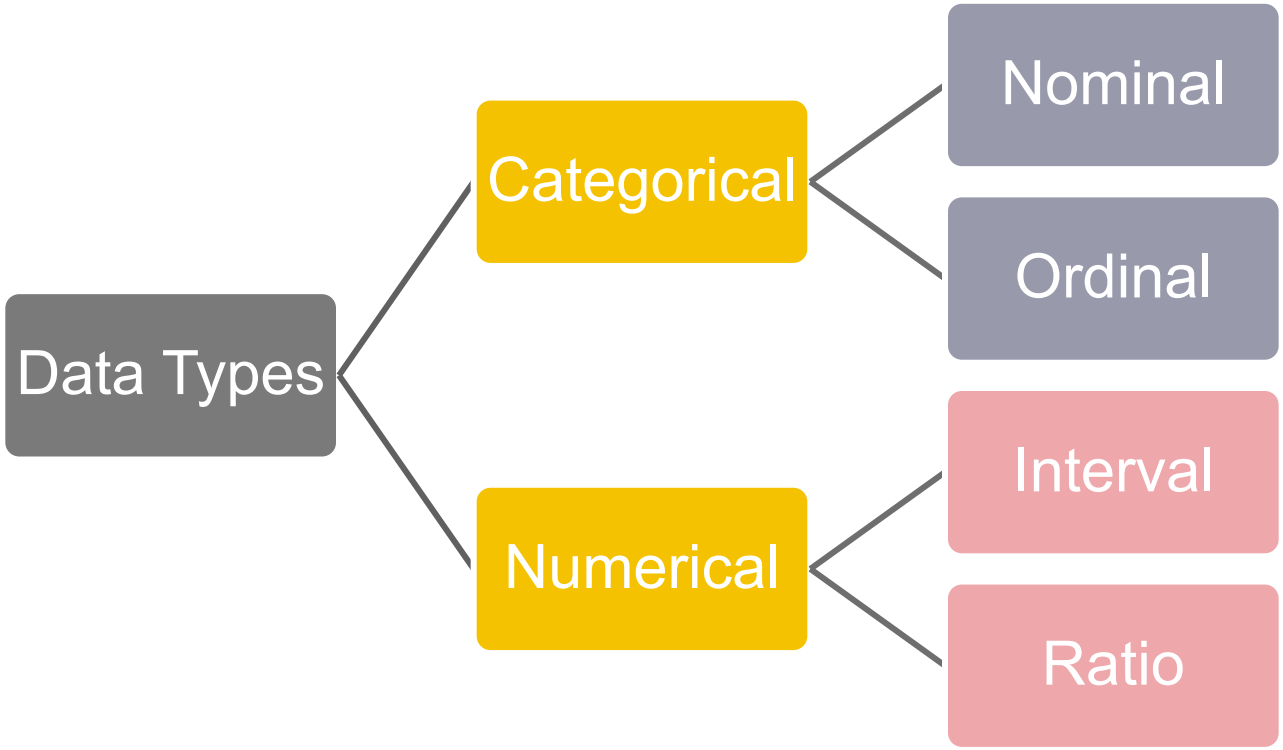
Can be processed into an *n* (number of entities) by *m* (number of attributes) matrix.

- Result of merging & processing different records!
- Picking the data that goes into this table has both technical and ethical concerns (recall: Target, Netflix, AOL examples)

ID	Title	Author	Year	Cover	Edition	Price
1	Emma	Austen	1815	Paper	20th	$5.75
2	Dracula	Stoker	1897	Hard	15th	$12.00
3	Ivanhoe	Scott	1820	Hard	8th	$25.00
4	Kidnapped	Stevenson	1886	Paper	11th	$5.00

CLASSICAL STATISTICAL VIEW OF DATA

There are four classical types of data



CATEGORICAL DATA: TAKES A VALUE FROM A FINITE SET

Nominal (aka Categorical) Data:

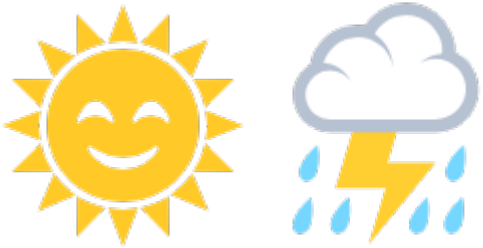
- Values have names: describe the categories, classes, or states of things
- Marital status, drink type, or some binary attribute
- Cannot compare easily, thus cannot naturally order them

Ordinal Data:

- Values have names: describe the categories, classes, or states of things
- However, there is an *ordering* over the values:
  - Strongly like, like, neutral, strongly dislike
- Lacks a mathematical notion of *distance* between the values

This distinction can be blurry…

- Is there an ordering over: sunny, overcast, rainy?



NUMERICAL DATA: MEASURED USING INTEGERS OR REALS

Interval Scale:

- Scale with fixed but arbitrary interval (e.g., dates)
- The difference between two values is *meaningful*:
  - Difference between 9/1/2019 and 10/1/2019 is the same as the difference between 9/1/2018 and 10/1/2018
- Can't compute ratios or scales: e.g., what unit is 9/1/2019 * 8/2/2020?

Ratio Scale:

- All the same properties as interval scale data, but the scale of measurement also possesses a **true-zero origin**
- Can look at the *ratio* of two quantities (unlike interval)
- E.g., zero money is an absolute, one money is half as much as two money, and so on

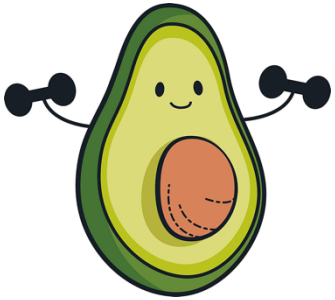
NUMERICAL DATA: EXAMPLES

Temperatures:

- Celsius / Fahrenheit: interval or ratio scale ???????????
  - **Interval:** 0C is not 0 heat, but is an arbitrary fixed point
  - Hence, we can't say that 30F is twice as warm as 15F.
- Kelvin (K): interval or ratio scale ???????????
  - **Ratio:** 0K is assumed to mean zero heat, a true fixed point

Weight:

- Grams: interval or ratio scale ??????????
- **Ratio:** 0g served as fixed point, 4g is twice 2g, …



GENERAL RULES

OK to compute....	Nominal	Ordinal	Interval	Ratio
frequency distribution	?	?	?	?

Thanks to GraphPad

GENERAL RULES

OK to compute....	Nominal	Ordinal	Interval	Ratio
frequency distribution	Yes	Yes	Yes	Yes
median and percentiles	?	?	?	?

GENERAL RULES

OK to compute....	Nominal	Ordinal	Interval	Ratio
frequency distribution	Yes	Yes	Yes	Yes
median and percentiles	No	Yes	Yes	Yes
addition or subtraction	?	?	?	?

GENERAL RULES

OK to compute....	Nominal	Ordinal	Interval	Ratio
frequency distribution	Yes	Yes	Yes	Yes
median and percentiles	No	Yes	Yes	Yes
addition or subtraction	No	No	Yes	Yes
mean or standard deviation	?	?	?	?

GENERAL RULES

OK to compute....	Nominal	Ordinal	Interval	Ratio
frequency distribution	Yes	Yes	Yes	Yes
median and percentiles	No	Yes	Yes	Yes
addition or subtraction	No	No	Yes	Yes
mean or standard deviation	No	No	Yes	Yes
ratio, or coefficient of variation	?	?	?	?

GENERAL RULES

OK to compute....	Nominal	Ordinal	Interval	Ratio
frequency distribution	Yes	Yes	Yes	Yes
median and percentiles	No	Yes	Yes	Yes
addition or subtraction	No	No	Yes	Yes
mean or standard deviation	No	No	Yes	Yes
ratio, or coefficient of variation	No	No	No	Yes

DATA MANIPULATION AND COMPUTATION

Data Science == manipulating and computing on data

Large to very large, but somewhat "structured" data

We will see several tools for doing that this semester

Thousands more out there that we won't cover

Need to learn to shift thinking from:

Imperative code to manipulate data structures

to:

Sequences/pipelines of operations on data

Should still know how to implement the operations themselves, especially for debugging performance (covered in classes like 420, 424), but we won't cover that much

DATA MANIPULATION AND COMPUTATION

1. **Data Representation**, i.e., what is the natural way to think about given data

One-dimensional Arrays, Vectors

0.1	2	3.2	6.5	3.4	4.1

“data”	”representation”	”i.e.”

Indexing
Slicing/subsetting
Filter
‘map’ → apply a function to every element
’reduce/aggregate’ → combine values to get a single scalar (e.g., sum, median)

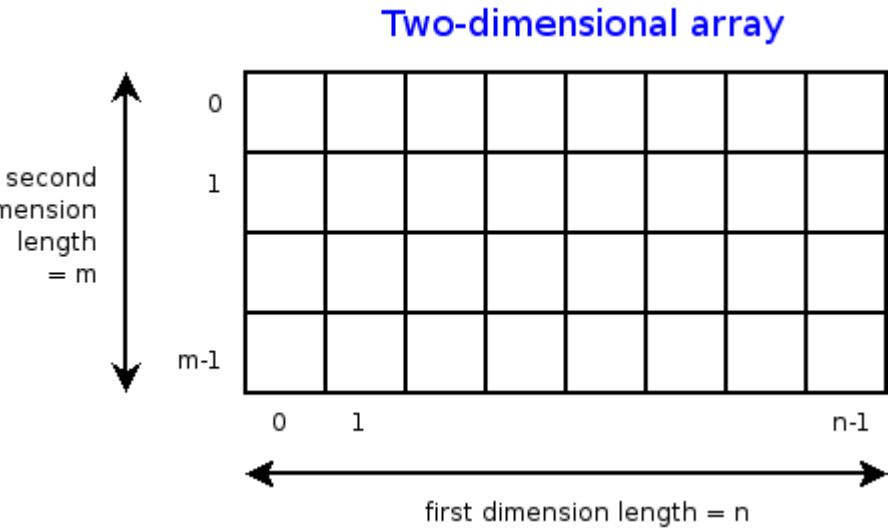
Given two vectors: **Dot and cross products**

2. **Data Processing Operations**, which take one or more datasets as input and produce one or more datasets as output

DATA MANIPULATION AND COMPUTATION

1. **Data Representation**, i.e., what is the natural way to think about given data

n-dimensional arrays



Indexing
Slicing/subsetting
Filter
'map' → apply a function to every element
'reduce/aggregate' → combine values across a row or a column (e.g., sum, average, median etc..)

2. **Data Processing Operations**, which take one or more datasets as input and produce one or more datasets as output

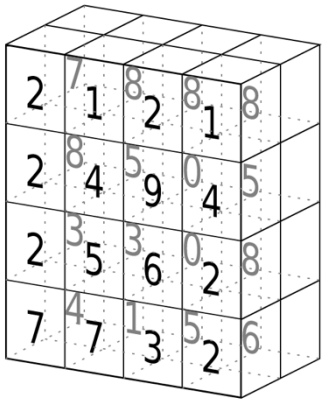
DATA MANIPULATION AND COMPUTATION

1. **Data Representation**, i.e., what is the natural way to think about given data

Matrices, Tensors

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

tensor of dimensions [6,4]
(matrix 6 by 4)



tensor of dimensions [4,4,2]

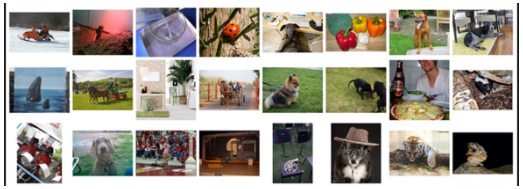
n-dimensional array operations
+
Linear Algebra
- **Matrix/tensor multiplication**
- **Transpose**
- **Matrix-vector multiplication**
- **Matrix factorization**

2. **Data Processing Operations**, which take one or more datasets as input and produce one or more datasets as output

DATA MANIPULATION AND COMPUTATION

1. **Data Representation**, i.e., what is the natural way to think about given data

Sets: of Objects



Filter
Map
Union

Reduce/Aggregate

Sets: of (Key, Value Pairs)

(amol@cs.umd.edu,(email1, email2,…))

(john@cs.umd.edu,(email3, email4,…))

Given two sets, **Combine/Join** using "keys"

Group and then aggregate

2. **Data Processing Operations**, which take one or more datasets as input and produce one or more datasets as output

DATA MANIPULATION AND COMPUTATION

1. **Data Representation**, i.e., what is the natural way to think about given data

Tables/Relations == Sets of Tuples

company	division	sector	tryint
00nil_Combined_Company	00nil_Combined_Division	00nil_Combined_Sector	14625
apple	00nil_Combined_Division	00nil_Combined_Sector	10125
apple	hardware	00nil_Combined_Sector	4500
apple	hardware	business	1350
apple	hardware	consumer	3150
apple	software	00nil_Combined_Sector	5625
apple	software	business	4950
apple	software	consumer	675
microsoft	00nil_Combined_Division	00nil_Combined_Sector	4500
microsoft	hardware	00nil_Combined_Sector	1890
microsoft	hardware	business	855
microsoft	hardware	consumer	1035
microsoft	software	00nil_Combined_Sector	2610
microsoft	software	business	1215
microsoft	software	consumer	1395

Filter rows or columns

”Join” two or more relations

”Group” and “aggregate” them

Relational Algebra formalizes some of them

Structured Query Language (SQL)

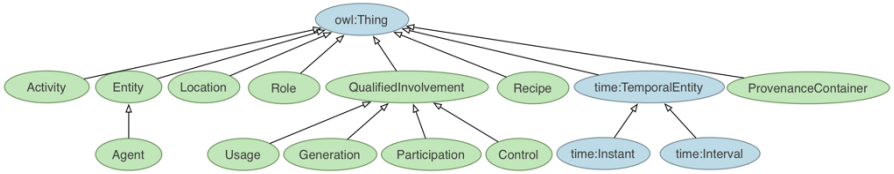
Many other languages and constructs, that look very similar

2. **Data Processing Operations**, which take one or more datasets as input and produce one or more datasets as output

DATA MANIPULATION AND COMPUTATION

1. **Data Representation**, i.e., what is the natural way to think about given data

Hierarchies/Trees/Graphs



"Path" queries

Graph Algorithms and Transformations

Network Science

Somewhat more ad hoc and special-purpose

Changing in recent years

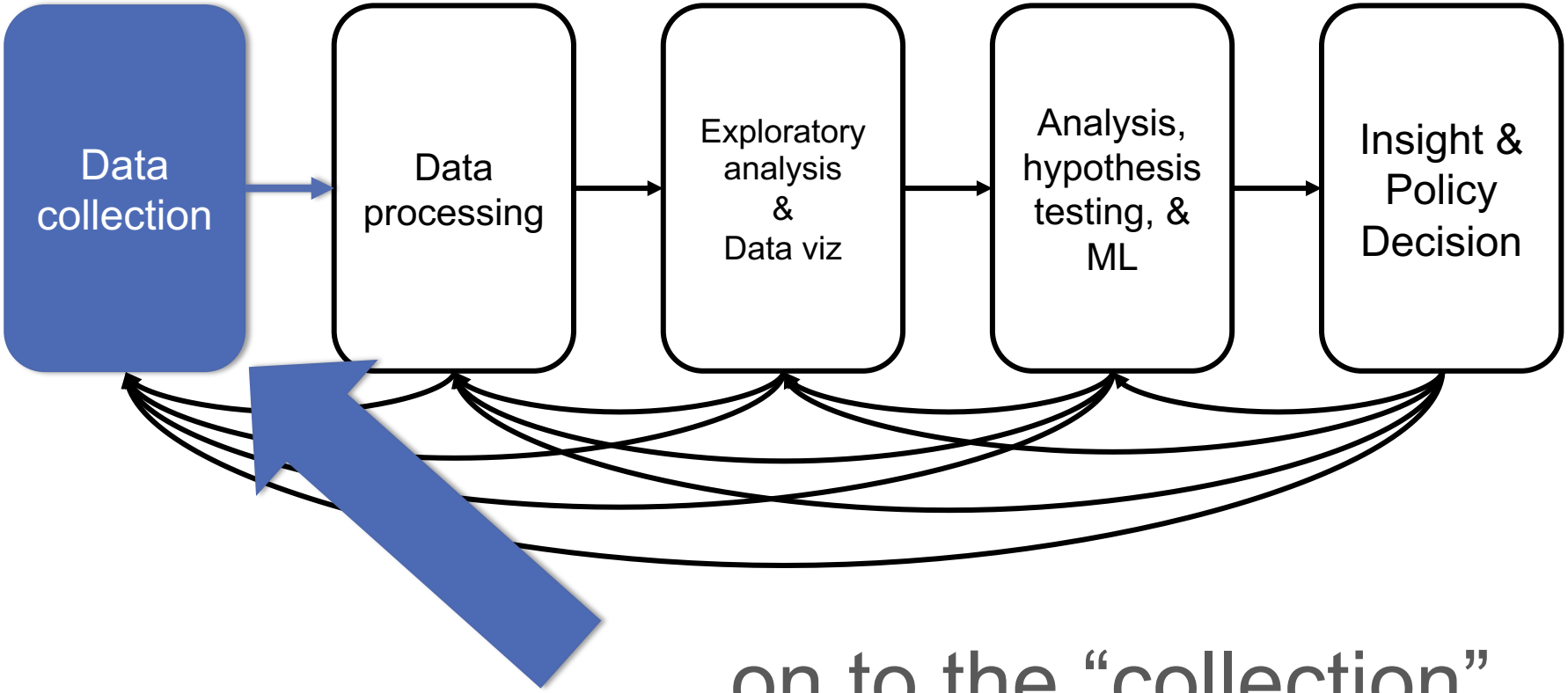
2. **Data Processing Operations**, which take one or more datasets as input and produce one or more datasets as output

DATA MANIPULATION AND COMPUTATION

1. **Data Representation**, i.e., what is the natural way to think about given data
2. **Data Processing Operations**, which take one or more datasets as input and produce

- **Why?**
  - Allows one to think at a higher level of abstraction, leading to simpler and easier-to-understand scripts
  - Provides ”independence” between the abstract operations and concrete implementation
  - Can switch from one implementation to another easily
- **For performance debugging, useful to know how they are implemented and rough characteristics**

NEXT LECTURE



… on to the “collection” part of things …