Define a function:

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

Define a function that returns a tuple:

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

```python
(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

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

7

range: returns an iterable object

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

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

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

```python
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

```python
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

```python
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.
Basic iteration over an array in Java:

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

Direct translation into Python:

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

A more “Pythonic” way of iterating:

```python
for element in arr:
    print( element )
```
LIST COMPREHENSIONS

Construct sets like a mathematician!
• \( P = \{ 1, 2, 4, 8, 16, \ldots, 2^{16} \} \)
• \( E = \{ x \mid x \in \mathbb{N} \text{ and } x \text{ is odd and } x < 1000 \} \)

Construct lists like a mathematician who codes!

\[
P = [ 2**x \text{ for } x \text{ in range}(17) ]
\]

\[
E = [ x \text{ for } x \text{ in range}(1000) \text{ if } x \% 2 != 0 ]
\]

• You’ll see these way more than \texttt{map} in the wild
• Many people consider \texttt{map/filter} not “pythonic”
• They can perform differently (\texttt{map} is “lazier”)

\[
E = [ x \text{ for } x \text{ in range}(1000) \text{ if } x \% 2 != 0 ]
\]
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:**

```python
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
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 readings soon.)

Come hang out at office hours
• Office hours are on the website.
• Also, email me – if it doesn’t fit your schedules.
TODAY’S LECTURE

Data collection

Data processing

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

with Python

Thanks: Zico Kolter’s 15-388
GOTTA CATCH ‘EM ALL

Five ways to get data:
• Direct download and load from local storage
• Generate locally via downloaded code (e.g., simulation)
• Query data from a database (covered in a few lectures)
• Query an API from the intra/internet
• Scrape data from a webpage

Covered today.
WHEREFORE ART THOU, API?

A web-based Application Programming Interface (API) like we’ll be using in this class is a contract between a server and a user stating:

“If you send me a specific request, I will return some information in a structured and documented format.”

(More generally, APIs can also perform actions, may not be web-based, be a set of protocols for communicating between processes, between an application and an OS, etc.)
“SEND ME A SPECIFIC REQUEST”

Most web API queries we’ll be doing will use HTTP requests:

• conda install –c anaconda requests=2.24.0

```python
r = requests.get( 'https://api.github.com/user', auth=('user', 'pass') )

r.status_code
200

r.headers[‘content-type’]
‘application/json; charset=utf8’

r.json()
{u'private_gists': 419, u'total_private_repos': 77, ...}
```
HTTP REQUESTS

https://www.google.com/?q=cmsc320&tbs=qdr:m

Google

HTTP GET Request:
GET /?q=cmsc320&tbs=qdr:m HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:10.0.1) Gecko/20100101 Firefox/10.0.1

```python
def main():
    params = { "q": "cmsc320", "tbs": "qdr:m" }
    r = requests.get( "https://www.google.com", params = params )

if __name__ == '__main__':
    main()
```

*be careful with https:// calls; requests will not verify SSL by default*
RESTFUL APIS

This class will just query web APIs, but full web APIs typically allow more.

Representational State Transfer (RESTful) APIs:

- **GET**: perform query, return data
- **POST**: create a new entry or object
- **PUT**: update an existing entry or object
- **DELETE**: delete an existing entry or object

Can be more intricate, but verbs (“put”) align with actions
QUERYING A RESTFUL API

Stateless: with every request, you send along a token/authentication of who you are

token = "super_secret_token"
r = requests.get("https://github.com/user",
params={"access_token": token})
print( r.content )

{"login":"Mohammad Nayeem Teli","id":10536112,"avatar_url":"ht..."}

• PUT/POST/DELETE can edit your repositories, etc.
• Try it out: https://github.com/settings/tokens/new
AUTHENTICATION AND OAUTH

Old and busted:

```python
r = requests.get("https://api.github.com/user", auth=("nayeemmm", "cmsc320fall2020"))
```

New hotness:

- What if I wanted to grant an app access to, e.g., my Facebook account without giving that app my password?
- OAuth: grants access tokens that give (possibly incomplete) access to a user or app without exposing a password
“... I WILL RETURN INFORMATION IN A STRUCTURED FORMAT.”

So we’ve queried a server using a well-formed GET request via the requests Python module. What comes back?

General structured data:
• Comma-Separated Value (CSV) files & strings
• Javascript Object Notation (JSON) files & strings
• HTML, XHTML, XML files & strings

Domain-specific structured data:
• Shapefiles: geospatial vector data (OpenStreetMap)
• RVT files: architectural planning (Autodesk Revit)
• You can make up your own! Always document it.
Any CSV reader worth anything can parse files with any delimiter, not just a comma (e.g., “TSV” for tab-separated)

1,28-Sep,Introduction,—,"pdf, pptx",Teli,
2,29-Sep,Scraping Data with Python,Anaconda's Test Drive.,,Teli,
3,30-Sep,"Vectors, Matrices, and Dataframes",Introduction to pandas.,,Teli,
4,31-Sep,Jupyter notebook lab,,,"Hanyu, Jue",

Don’t write your own CSV or JSON parser

```python
import csv
with open("schedule.csv", "rb") as f:
    reader = csv.reader(f, delimiter="","", quotechar=""""")
    for row in reader:
        print(row)
```
JSON FILES & STRINGS

JSON is a method for serializing objects:
- Convert an object into a string (done in Java in 131/132?)
- Deserialization converts a string back to an object

Easy for humans to read (and sanity check, edit)
Defined by three universal data structures

Images from: http://www.json.org/
JSON IN PYTHON

Some built-in types: "Strings", 1.0, True, False, None

Lists: ["Goodbye", "Cruel", "World"]

Dictionaries: {"hello": "bonjour", "goodbye", "au revoir"}

Dictionaries within lists within dictionaries within lists:

[1, 2, {"Help": [
    "I’m", {"trapped": "in"},
    "CMSC641"
]}
]}
GET https://api.twitter.com/1.1/friends/list.json?cursor=-1&screen_name=twitterapi&skip_status=true&include_user_entities=false

{
    "previous_cursor": 0,
    "previous_cursor_str": "0",
    "next_cursor": 1333504313713126852,
    "users": [ {
        "profile_sidebar_fill_color": "252429",
        "profile_sidebar_border_color": "181A1E",
        "profile_background_tile": false,
        "name": "Sylvain Carle",
        "profile_image_url": "http://a0.twimg.com/profile_images/2838630046/4b82e286a659fae310012520f4f756bb_normal.png",
        "created_at": "Thu Jan 18 00:10:45 +0000 2007",
        ...} ...]
Repeat: don’t write your own CSV or JSON parser

- https://news.ycombinator.com/item?id=7796268
- rsdy.github.io/posts/dont_write_your_json_parser_plz.html

Python comes with a fine JSON parser

```
import json

r = requests.get( "https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name=JohnDoe", auth=auth )

data = json.loads(r.content)

json.load(some_file)  # loads JSON from a file
json.dump(json_obj, some_file)  # writes JSON to file
json.dumps(json_obj)  # returns JSON string
```
Still hugely popular online, but JSON has essentially replaced XML for:

- Asynchronous browser ←→ server calls
- Many (most?) newer web APIs

**XML is a hierarchical markup language:**

```xml
<tag attribute="value1">
  <subtag>
  Some cool words or values go here!
  </subtag>
  <openclosetag attribute="value2" />
</tag>
```

You probably won’t see much XML, but you will see plenty of HTML, its substantially less well-behaved cousin ...
DOCUMENT OBJECT MODEL (DOM)
SCRAPING HTML IN PYTHON

HTML – the specification – is fairly pure
HTML – what you find on the web – is horrifying
We’ll use Beautiful Soup:
• conda install -c asmeurer beautiful-soup=4.3.2

```python
import requests
from bs4 import BeautifulSoup

r = requests.get( "https://cs.umd.edu/class/fall2020/cmsc320-0201/" )

root = BeautifulSoup( r.content )
root.findAll("a")  # links for CMSC320
```
Totally not hypothetical situation:
• You really want to learn about data science, so you choose to download all of last semester’s CMSC320 lecture slides to wallpaper your room …
• … but you now have carpal tunnel syndrome from clicking refresh on Piazza last night, and can no longer click on the PDF and PPTX links.

Hopeless? No! Earlier, you built a scraper to do this!

```python
lnks = root.findAll("a")  # links for CMSC320
```

Sort of. You only want PDF and PPTX files, not links to other websites or files.
REGULAR
EXPRESSIONS

Given a list of URLs (strings), how do I find only those strings that end in *.pdf or *.pptx?

• Regular expressions!
• (Actually Python strings come with a built-in endswith function.)

"this_is_a_filename.pdf".endswith((".pdf", ".pptx"))

What about .pDf or .pPTx, still legal extensions for PDF/PPTX?

• Regular expressions!
• (Or cheat the system again: built-in string lower function.)

"tHiS_IS_a_FiLeNAme.pDF".lower().endswith(
    (".pdf", ".pptx"))
IF YOU'RE HAVIN' PERL PROBLEMS I FEEL BAD FOR YOU, SON—

I GOT 99 PROBLEMS,

SO I USED REGULAR EXPRESSIONS.

NOW I HAVE 100 PROBLEMS.
REGULAR EXPRESSIONS

Used to search for specific elements, or groups of elements, that match a pattern

```python
import re

# Find the index of the 1st occurrence of "cmsc320"
match = re.search(r"cmsc320", text)
print( match.start() )

# Does start of text match "cmsc320"?
match = re.match(r"cmsc320", text)

# Iterate over all matches for "cmsc320" in text
for match in re.finditer(r"cmsc320", text):
    print( match.start() )

# Return all matches of "cmsc320" in the text
match = re.findall(r"cmsc320", text)
```
MATCHING MULTIPLE CHARACTERS

Can match sets of characters, or multiple and more elaborate sets and sequences of characters:

- Match the character ‘a’: a
- Match the character ‘a’, ‘b’, or ‘c’: [abc]
- Match any character except ‘a’, ‘b’, or ‘c’: [^abc]
- Match any digit: \d (= [0123456789] or [0-9])
- Match any alphanumeric: \w (= [a-zA-Z0-9_])
- Match any whitespace: \s (= [\t\n\r\f\v])
- Match any character: .

Special characters must be escaped: .^$*+?{[]}( )
MATCHING SEQUENCES AND REPEATED CHARACTERS

A few common modifiers (available in Python and most other high-level languages; +, {n}, {n,} may not):

- Match character ‘a’ exactly once: a
- Match character ‘a’ zero or once: a?
- Match character ‘a’ zero or more times: a*
- Match character ‘a’ one or more times: a+
- Match character ‘a’ exactly \( n \) times: \( a^n \)
- Match character ‘a’ at least \( n \) times: \( a^{n,} \)

Example: match all instances of “University of <somewhere>” where <somewhere> is an alphanumeric string with at least 3 characters:

- \s*University\sof\s\w{3,}
COMPILED REGEXES

If you’re going to reuse the same regex many times, or if you aren’t but things are going slowly for some reason, try compiling the regular expression.

* https://blog.codinghorror.com/to-compile-or-not-to-compile/

```python
# Compile the regular expression “cmsc320”
regex = re.compile(r“cmsc320”)  

# Use it repeatedly to search for matches in text
regex.match( text )  # does start of text match?
regex.search( text ) # find the first match or None
regex.findall( text ) # find all matches
```
import re
import requests
from bs4 import BeautifulSoup

try:
    from urllib.parse import urlparse
except ImportError:
    from urlparse import urlparse

# HTTP GET request sent to the URL url
r = requests.get( url )

# Use BeautifulSoup to parse the GET response
root = BeautifulSoup( r.content )
lnks = root.find("div", id="schedule")
    .find("table")
    .find("tbody").findAll("a")
# Cycle through the href for each anchor, checking
# to see if it's a PDF/PPTX link or not
import os
outbase='~/Users/nayeem/
for lnk in lnks:
    href = lnk['href']
    # If it's a PDF/PPTX link, queue a download
    if href.lower().endswith(('pdf', 'pptx')):
        urld = urljoin(url, href)
        rd = requests.get(urld, stream=True)
        filename = urld.split("/")[-1]
        # Write the downloaded PDF to a file
        outfile = os.path.join(outbase, filename)
        with open(outfile, 'wb') as f:
            f.write(rd.content)

Get some more data?!
NEXT LECTURE

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

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Data viz

Analysis, hypothesis testing, & ML

Insight & Policy Decision

Exploratory analysis & Dataviz
NEXT CLASS:

NUMPY, SCIPY, AND DATAFRAMES

\[ y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it} \]