# STIA 210 : Intro to Data Science

Prof. Venkatesh Krishnamoorthy
Prof. Kyle Meyer
(usually) Mondays 6:30 - 9ish PM
Room ICC 219A (for now…)

## Course Logistics Up Front

- Most important logistic throughout the course:
    - Ask questions whenever you feel like you need to!

## Agenda

- Introductions - Professors, MITRE, and Students
- Motivation for the course
- Course logistics
    - Syllabus
    - Assignments
    - Policies
    - Office Hours
    - Final Project
- Introduction to Section 1: Fundamentals
    - Intro to programming - Python
    - Pandas, numpy, matplotlib - the big 3

# Introductions: Prof. Venkatesh Krishnamoorthy

Hewlett Packard Enterprise

APACHE nifi

MISSOURI S&T

MICRO FOCUS

Hybrid Cloud Management

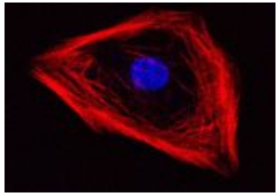
MITRE

## Introductions: Prof. Venkatesh Krishnamoorthy

- Masters in Computer Applications - India
- Data Semantic Analysis Engineer at MITRE
- Professional Services- Cloud Orchestration and Automation @ Hewlett Packard Software ~ 15 years

    Mentored and trained Technology Teams, Customers
- Presently enrolled in Computational Intelligence Graduate Program @ Missouri University of Science & Technology

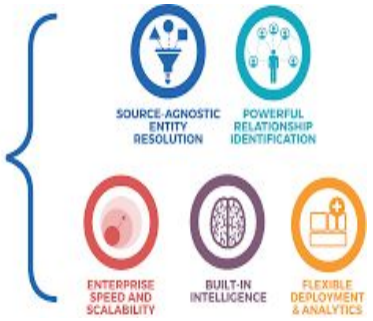
## Introductions: Prof. Kyle Meyer



hadoop

BROWN

NOVETTA


SOURCE-AGNOSTIC ENTITY RESOLUTION
POWERFUL RELATIONSHIP IDENTIFICATION
ENTERPRISE SPEED AND SCALABILITY
BUILT-IN INTELLIGENCE
FLEXIBLE DEPLOYMENT & ANALYTICS

MITRE

aws

## Introductions: Prof. Kyle Meyer

- Undergrad & Masters in Biomedical Engineering at Brown University
- Head Instructor - Android Application Development Summer Course 2016
- Co-Instructor - Disease Control - 2016 - 2019

- Software Engineer in Hadoop & AWS @ Novetta in Entity Resolution (more to come)
- Senior Software Engineer at MITRE

- Certs for days: AWS Machine Learning Certified, AWS Cloud Practitioner, AWS Certified Associate Developer, Certified Scaled Agile Framework for Government

# Entity Resolution

- Entity Resolution
  - Linking and grouping records between different data sources that describe the same "entity"
- Nightmare for Data Scientists
- Goal: a new column (entity ID), that can be used for better queries, joins

| Name | Address | Phone |
|---|---|---|
| John Smith | 12 Penn. Ave | 7036281190 |
| Johnny S. | | 7036281190 |
| J. Smith | 45 Mass. Ave | 1 (703) 628 - 1190 |
| Sally Smith | 45 Mass. Ave | 7031298871 |

| Full Name | Username | Phone |
|---|---|---|
| John Smith | jsmith | 628 - 1190 |
| John | jsmith2 | |
| Johnny S. | itsssjohnny | 7036281190 |

# Entity Resolution

- Entity Resolution Issues
  - Name (or other string) disambiguation
    - Punctuation (easy to fix), nicknames (often lookup tables, can get tedious), common names, uncommon spellings (again lookup tables), different formats (First Middle Last, or Last, First M.I., or F.I. Last, or all initials - settled by better programming often)

| Name | Address | Phone |
|---|---|---|
| John Smith | 12 Penn. Ave | 7036281190 |
| Johnny S. |  | 7036281190 |
| J. Smith | 45 Mass. Ave | 1 (703) 628 - 1190 |
| Sally Smith | 45 Mass. Ave | 7031298871 |

| Full Name | Username | Phone |
|---|---|---|
| John Smith | jsmith | 628 - 1190 |
| John | jsmith2 |  |
| Johnny S. | itsssjohnny | 7036281190 |

# Entity Resolution

- Entity Resolution Issues
  - Addresses
    - USPS releases Address Format Standardization software (proprietary, bulky, slow)
    - Mix of data types, abbreviations, formats
  - Phone numbers have many of the same problems
    - Area code, country code, punctuation... many can be fixed by better programming!

| Name | Address | Phone |
|---|---|---|
| John Smith | 12 Penn. Ave | 7036281190 |
| Johnny S. |  | 7036281190 |
| J. Smith | 45 Mass. Ave | 1 (703) 628 - 1190 |
| Sally Smith | 45 Mass. Ave | 7031298871 |

| Full Name | Username | Phone |
|---|---|---|
| John Smith | jsmith | 628 - 1190 |
| John | jsmith2 |  |
| Johnny S. | itsssjohnny | 7036281190 |

## Entity Resolution

- Entity Resolution Issues
    - Different data sets have different schemas

| Name | Address | Phone |
|---|---|---|
| John Smith | 12 Penn. Ave | 7036281190 |
| Johnny S. |  | 7036281190 |
| J. Smith | 45 Mass. Ave | 1 (703) 628 - 1190 |
| Sally Smith | 45 Mass. Ave | 7031298871 |

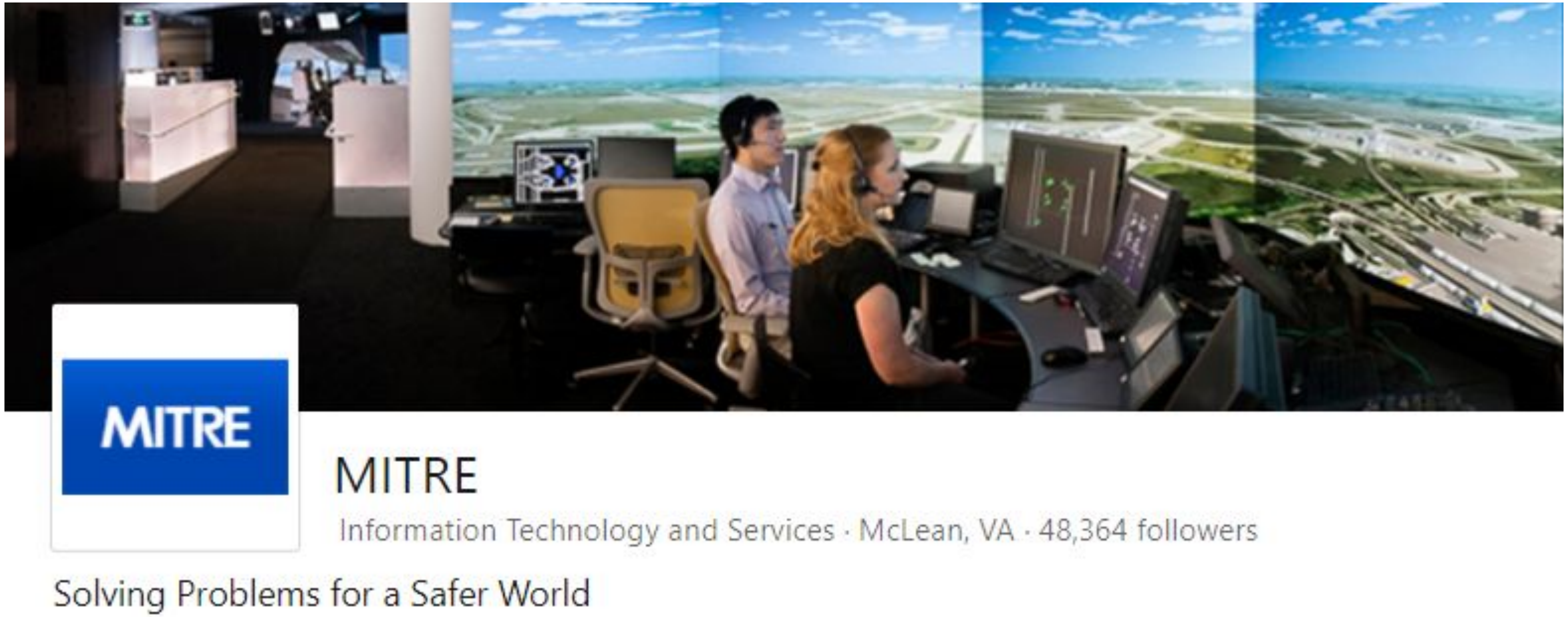
| Full Name | Username | Phone |
|---|---|---|
| John Smith | jsmith | 628 - 1190 |
| John | jsmith2 |  |
| Johnny S. | itsssjohnny | 7036281190 |

## Entity Resolution

- Grouping
  - Which records between the multiple data sets belong to John Smith (creates an entity called John Smith)
- Linking
  - Which entities are related (John Smith and Sally Smith)

- Use Cases:
  - Fraud detection, homeland security, family trees, health records and family histories, related web searches, social networks

## About both the Professors: MITRE


MITRE

Information Technology and Services · McLean, VA · 48,364 followers

Solving Problems for a Safer World

FFRDC = Federally Funded Research and Development Center

## Who are you?

- Name
- Year
- Major
- Let us know your background in Python, coding in general, and/or data science + math (course and research positions if applicable)
- What you want to get out of this course (any ideas of projects in your head?)

## Motivation

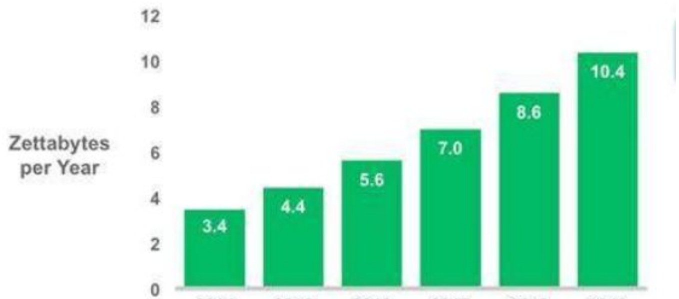
- **Data** = factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation
- **Science** = the state of knowing : knowledge as distinguished from ignorance or misunderstanding

(Merriam-Webster dictionary)

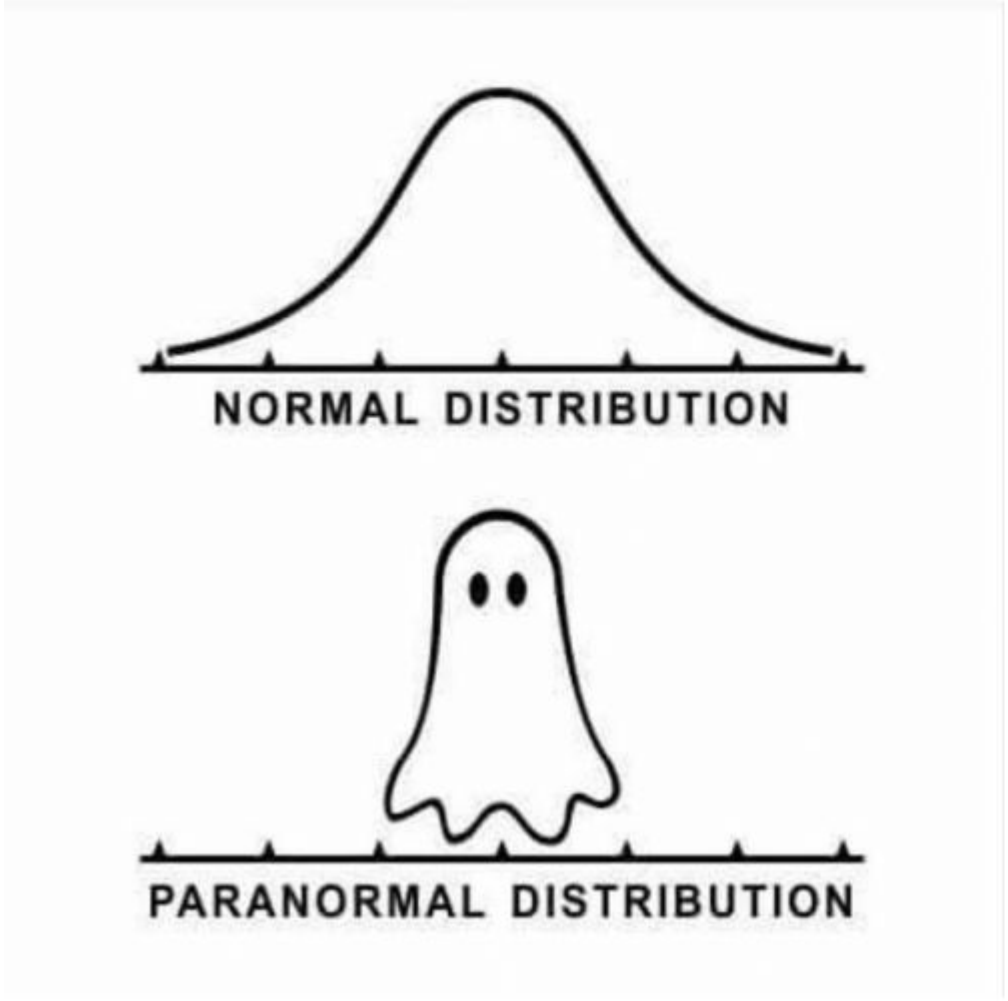
...also pet peeve → the word “data” is the plural form of datum...

## Motivation

Built on statistics, data science will empower every decision made by businesses, government, individuals, adversaries, etc.



Annual data growth rate Source: Cisco Global Cloud Index, 2014-2019



Zettabyte = 10^21… or 1 billion TB

## Motivation

If you do well in this course, you'll understand all the jokes in this video!

https://www.youtube.com/watch?v=UXd0EDy7aTY

## Course Logistics - Section 1: Fundamentals

- Week 1: Today! (Happy Labor Day Weekend) **Intro to Course**
    - Introduction to the course, policies, final project)
    - Introduction to Python, Google Colaboratory, Pandas
- Week 2: Sept. 9th - **Data Manipulation**
    - Go over first problem set, reading discussion
    - Get our hands on data, ETL, string methods/regex, reshaping, joining/merging data
- Week 3: Sept. 16th - **Visualization Fundamentals**

## Course Logistics - Section 2: Intro to Statistical Methods

- Week 4: Sept. 23 **- Statistical Foundation for the course**

- Week 5: Sept. 30th - **Intro to Supervised Learning**
  - Regression

- Week 6: Oct. 7th - **Intro to Supervised Learning 2**
  - Classification

## Course Logistics - Section 2: Intro to Statistical Methods

- Week 7: Oct. 21st **- Unsupervised Learning**

- Week 8: Oct. 28th - **Unsupervised Learning continued & Intro to Reinforcement Learning**

- Week 9: Nov. 4th - **Reinforcement Learning**

## Course Logistics - Section 3: Special Topics

- Week 10: Nov. 11th **- Intro to Neural Networks**
- Week 11: Nov. 18th - **Natural Language Processing**
- Week 12: Nov. 25th - **Focus on Geospatial Data**

## Course Logistics - Section 4: Sprint to the finish

- Week 13: Dec 2nd **- Final Exam Prep**
- Week 14: Dec. 9th **- Final Project Presentations**

## Course Logistics

- Grades
  - 25 % Weekly Assignments
    - Will include in class participation for discussions on reading assignments when applicable
    - 8 Problem Sets
  - 60 % Final Project
  - 15 % Final Exam

## Course Logistics

- Grading policies:
  - The problem sets will be graded and returned before the following class, 8 Problem Sets total (may vary in weight, but they will mostly be the same weight)
  - Participation in discussions will make up remaining grade for “Weekly Assignments”
  - Problem Sets will be graded for accuracy (did you get the correct answer), deductions will be made for bad code, additional points usually available if you went above and beyond the question

## Final Project

- The problem sets will force you to start your final project early by slowly ramping up the deliverables

- The Final Project will be done individually, though components of the project may overlap between students
    - More on this later

## Final Project

- Requirements Document Approach to the Final Project
- A Final Project will:
  - Result in a professional looking paper, written in LateX using an Academic Journal format (**15%**)
  - Demonstrate mastery of the concepts of the course
  - Utilize structured and unstructured data
    - Open source as well as real data - collected by you!
  - Result in a tangible, functional piece of software (**30%**)
  - Derive <u>real insight</u> from <u>real data</u> to answer a <u>hypothesis</u>
  - End in a presentation to the class that demonstrates your ability to explain your work, field questions, and demo live software (**15%**)

## Final Project

- Open ended, but heading in a known direction
  - Scope will be important
    - *Over scope* → you bit off a project that was too hard or impossible, you don't get real insights, you don't get functional software, you don't get the A
    - *Under scope* → your project was too easy, you pointed out the obvious, your software wasn't much more than the internet already has, you don't get the A
    - *Goldilocks scope* - you worked hard to get a piece of software you're proud of, your data was unique and the methods applied led to non-obvious insights …. You get the A

## Course Logistics

- In class:
    - Laptops = cool, phones = less cool, phone calls = very uncool
    - Google, Stack Overflow = cool, Instagram = less cool
    - Questions = super cool

## Office Hours

- Official Hours:
    - Weekly in person for Prof. Meyer: Before class on Mondays or by request
    - Weekly in person for Prof. Krishnamoorthy: By appointment
    - Weekly availability via Zoom:
        - Kyle: TBD
        - Venkatesh: TBD
- If you want to reach out with questions (within an hour or so up to midnight-ish):
    - Email kyle_meyer@alumni.brown.edu
    - Email vkrishnam23@gmail.com ( will switch to Georgetown email soon!)

## Course logistics

- We understand the obstacles that you face through the semester may impact your course attendance or performance
    - **Religious or Cultural Holidays and Traditions** will be respected, and we will make arrangements for you to catch up on material if needed
    - **Learning accommodations** will also be made if asked for - as well as resources via the Academic Research Center http://academicsupport.georgetown.edu/disability
    - **Title IX**: The University treats sexual misconduct very seriously, and its impact on a student is immeasurable. If you or someone you know faces a situation in this category, we will absolutely fulfill our duty as Faculty members to support you confidentially and respect your needs

## Course Logistics

- How the honor code applies to this course:
  - There's a lot of great resources for code + data science on the internet
  - Copying code verbatim is a clear honor code violation
  - Using code snippets and using comments to cite the source (even if it's StackOverflow) is an acceptable approach
  - Your comments in your code will let us know you understand it, and that it's really your work

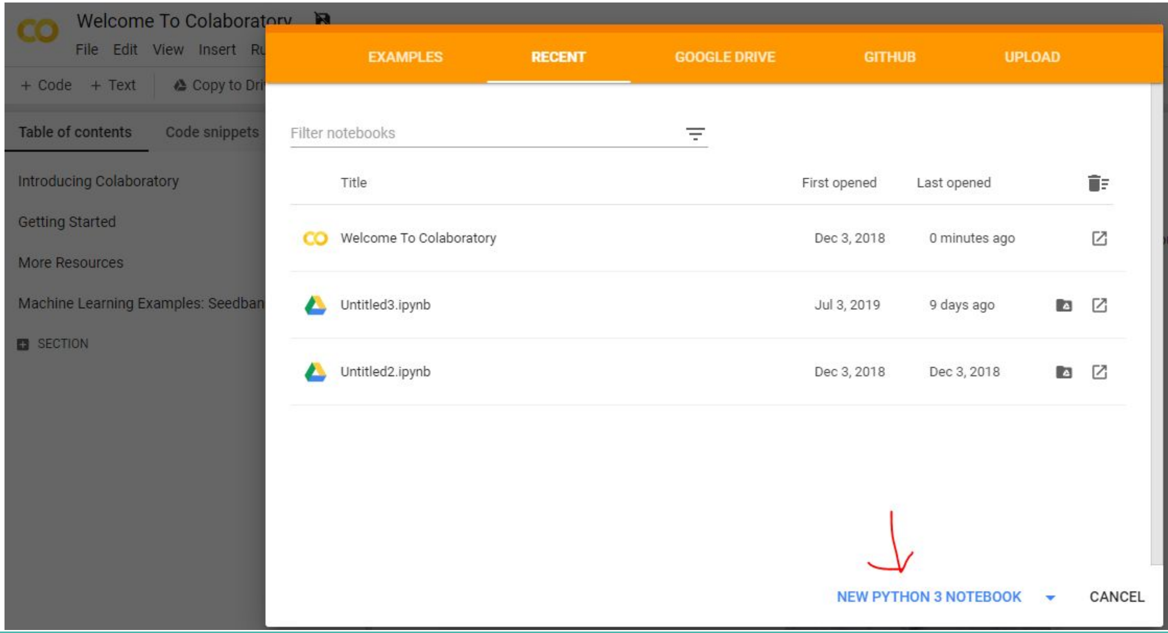
## Course Logistics

- Coding languages
    - This class will use Python, we will teach Python… you can choose not to if you feel strongly
    - To avoid the hassle of setting up a development environment, we will use **Google Colaboratory**

# Google Colaboratory

- Navigate to http://colab.research.google.com
- Log in with preferred gmail credentials, start a Python 3 Notebook
- All Notebooks have Internet access
  !pip install <something>
- Share Notebooks with Professors and partner for improved experience

- Enable Corgi Mode

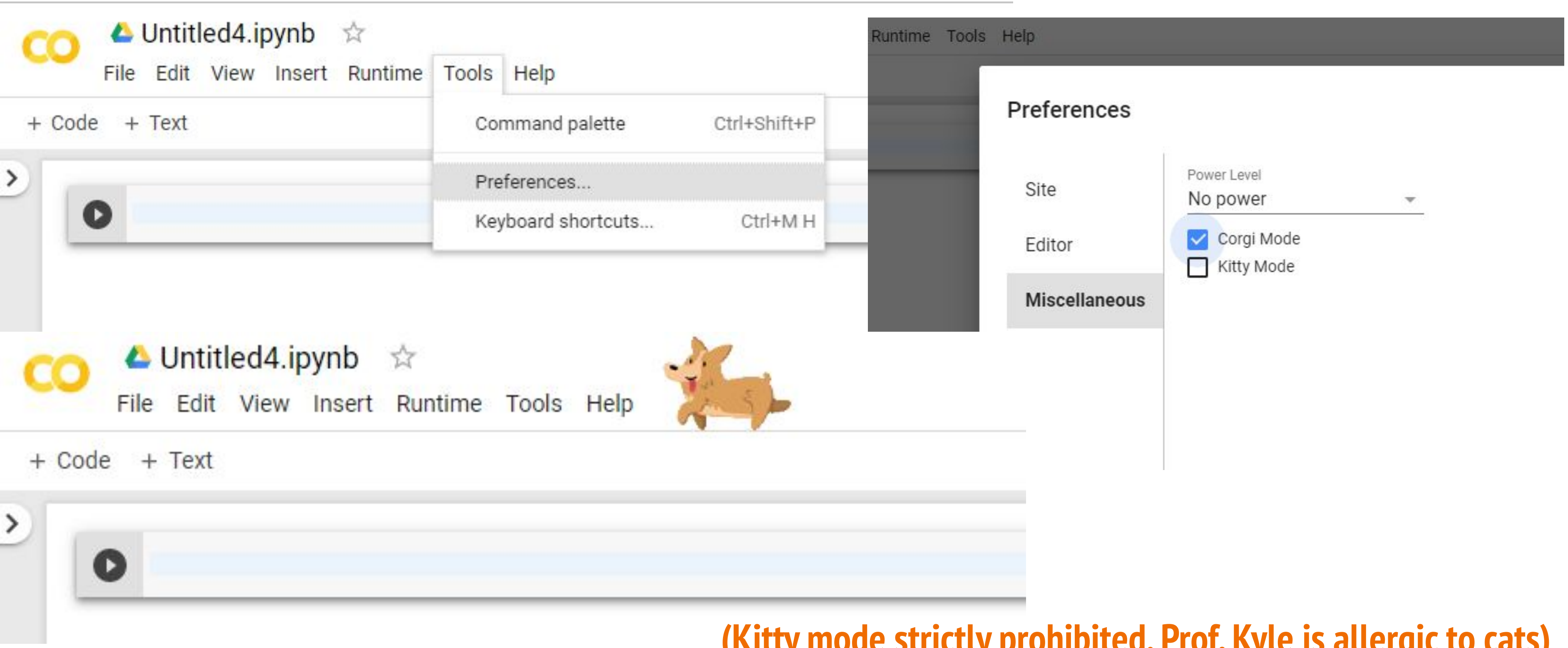

Welcome To Colaboratory
File Edit View Insert Ru
+ Code + Text Copy to Dri

| Table of contents | Code snippets |
|---|---|

Introducing Colaboratory
Getting Started
More Resources
Machine Learning Examples: Seedban
SECTION

| Title | First opened | Last opened |
|---|---|---|
| Welcome To Colaboratory | Dec 3, 2018 | 0 minutes ago |
| Untitled3.ipynb | Jul 3, 2019 | 9 days ago |
| Untitled2.ipynb | Dec 3, 2018 | Dec 3, 2018 |

# Assignment 1: Corgi Mode


Untitled4.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
Command palette Ctrl+Shift+P
Preferences...
Keyboard shortcuts... Ctrl+M H
Runtime Tools Help
Preferences
Site
Editor
Miscellaneous
Untitled4.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text

Power Level: No power

[✓] Corgi Mode

[ ] Kitty Mode

(Kitty mode strictly prohibited, Prof. Kyle is allergic to cats)

## Section 1 - Fundamentals

## Intro to Python Programming

- Syntax overview, basic commands
- Intro to libraries for data scientists

Piece of advice, write the code first, understand it second

## Python basics

- Strong focus on readability
  - Spacing matters, lines starting with # don't
  - Strong recommendation that lines should have fewer than 80 characters (**Style guide = Python Enhancement Proposals - or PEP**)

- Powerfully wraps high level system functions into readable code
  - As a result, Python is allegedly "less performance oriented"

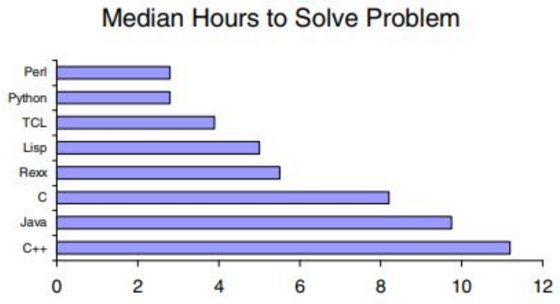
# Python performance hit is always worth it

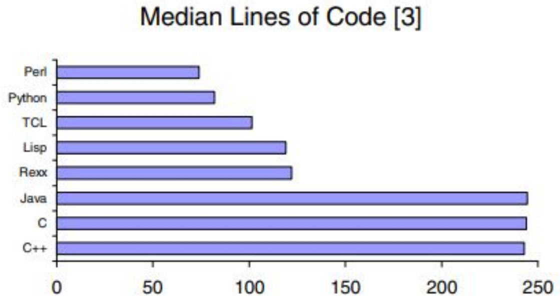
- Python is easier to write
  - Optimize your most expensive resource
    - → your developer's time

- By writing fewer lines of code, code is easier to maintain - reducing bugs over time and preventing single point of failure personnel

[https://medium.com/pyslackers/yes-python-is-slow-and-i-dont-care-13763980b5a1](https://medium.com/pyslackers/yes-python-is-slow-and-i-dont-care-13763980b5a1)

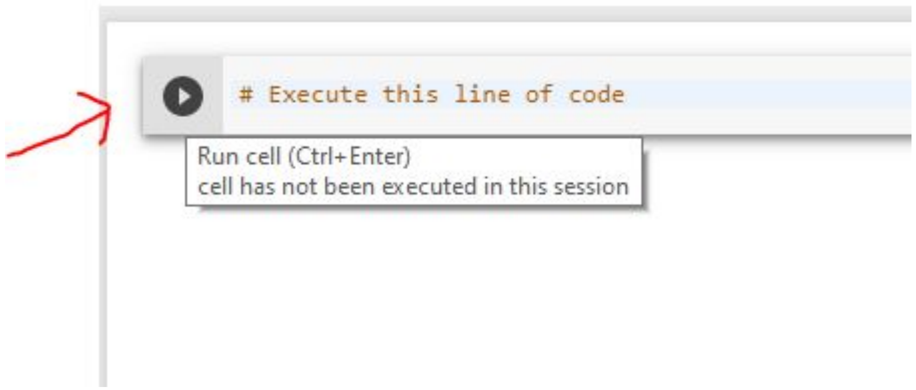
Data compiled from studies by Prechelt [1] and Garret [2] of a particular string processing problem.
Connelly Barnes, public domain 2006.





## Colab Notebooks

- Use Python 3 Notebooks only!
  - Python 2 will no longer be supported in 2020 - expect legacy systems to break
- When typing code, hit <shift> + <enter> to execute the block, or the play button:

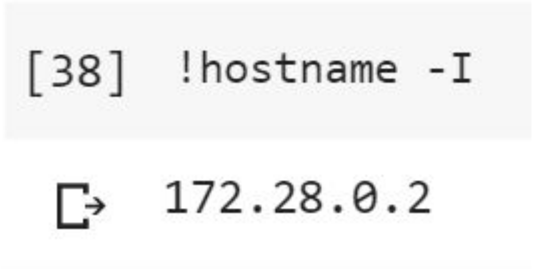


```
# Execute this line of code
```

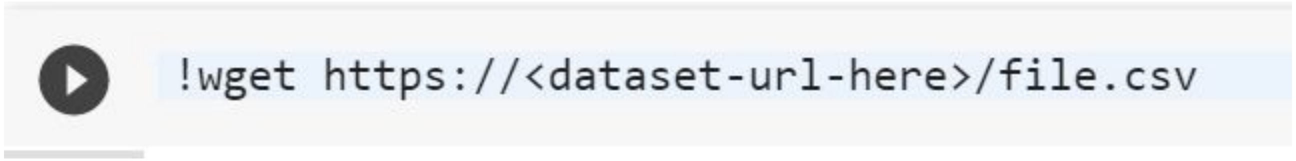
Run cell (Ctrl+Enter)
cell has not been executed in this session

## Colab Notebooks

- Execute bash linux commands by starting a line with !
  - This will help greatly with debugging and local finals, installing libraries too
  - Bash / Linux / Unix will help you in life, take big breaths and small steps



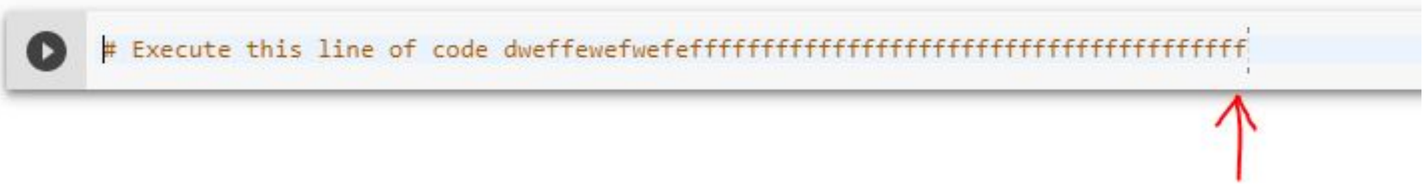
```
[38]  !hostname -I

      172.28.0.2
```



```
!wget https://<dataset-url-here>/file.csv
```

# Colab Notebooks

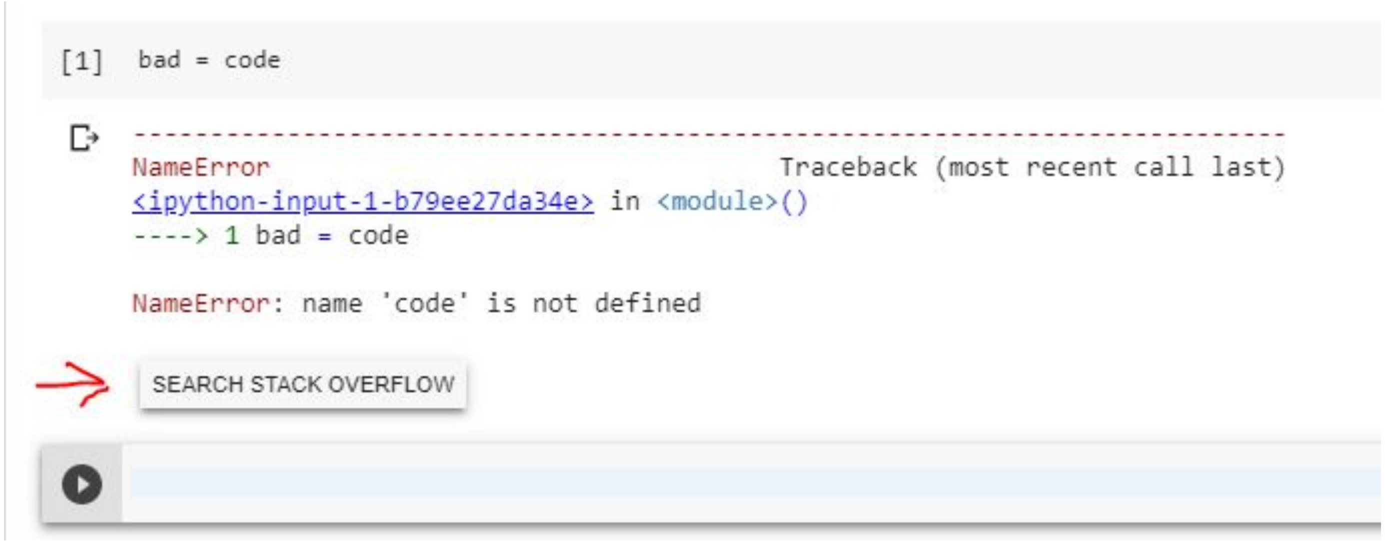
- Will make PEP-like recommendations, like to avoid more than 80 characters per line (with a dashed vertical line to deter you from typing further):



```
# Execute this line of code dweffewefwefeffffffffffffffffffffffffffffffffffffff
```

# Colab Notebooks

- Will raise errors when code fails, also offering to automagically search Stack Overflow for the issue (here, the variable code was not defined):

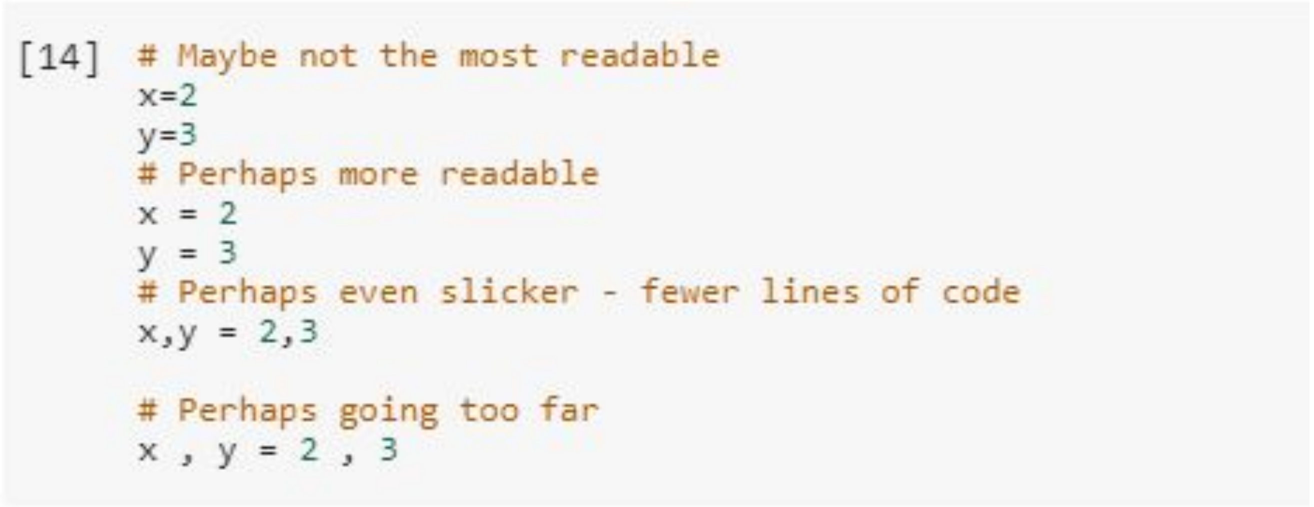


```
[1]  bad = code
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-b79ee27da34e> in <module>()
----> 1 bad = code

NameError: name 'code' is not defined
```

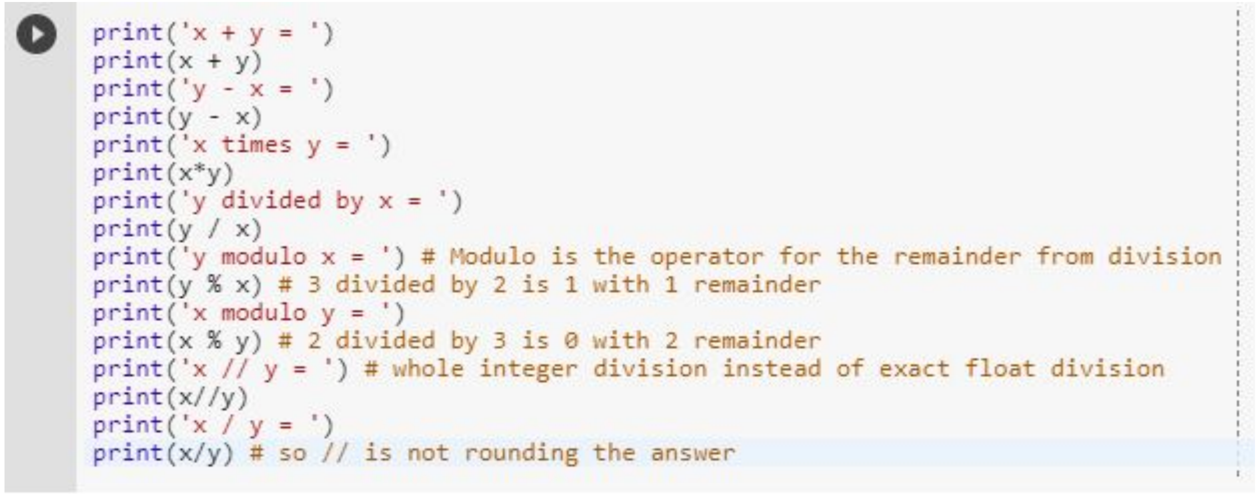
## Python basics

- Define variables

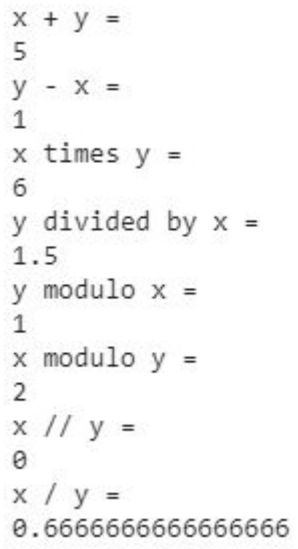


```
[14] # Maybe not the most readable
     x=2
     y=3
     # Perhaps more readable
     x = 2
     y = 3
     # Perhaps even slicker - fewer lines of code
     x,y = 2,3

     # Perhaps going too far
     x , y = 2 , 3
```

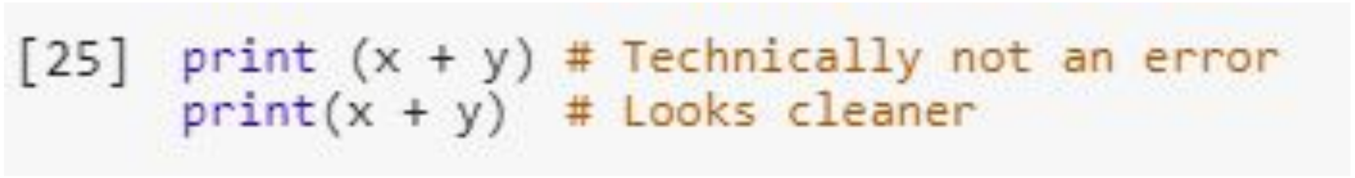
# Python basics - simple arithmetic



```
print('x + y = ')
print(x + y)
print('y - x = ')
print(y - x)
print('x times y = ')
print(x*y)
print('y divided by x = ')
print(y / x)
print('y modulo x = ') # Modulo is the operator for the remainder from division
print(y % x) # 3 divided by 2 is 1 with 1 remainder
print('x modulo y = ')
print(x % y) # 2 divided by 3 is 0 with 2 remainder
print('x // y = ') # whole integer division instead of exact float division
print(x//y)
print('x / y = ')
print(x/y) # so // is not rounding the answer
```



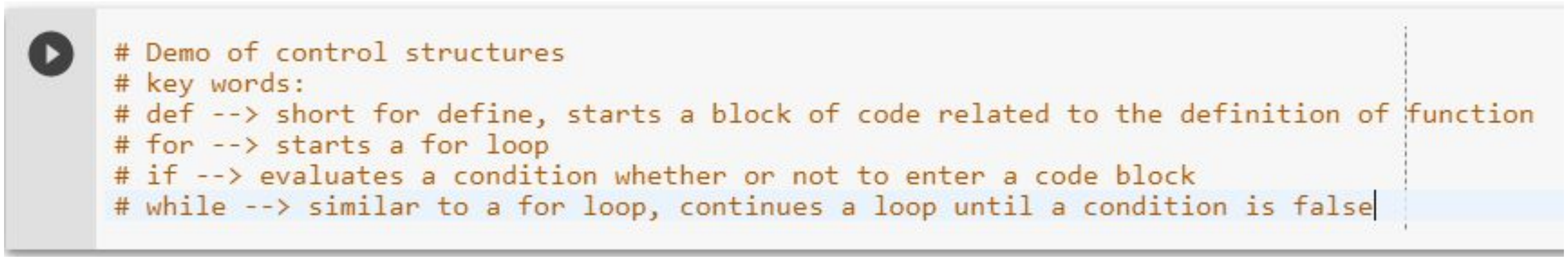
```
x + y =
5
y - x =
1
x times y =
6
y divided by x =
1.5
y modulo x =
1
x modulo y =
2
x // y =
0
x / y =
0.6666666666666666
```



```
[25] print (x + y) # Technically not an error
     print(x + y)  # Looks cleaner
```

## Python basics, control structures

- In Python, indentation indicates a block of related code that follows a colon :

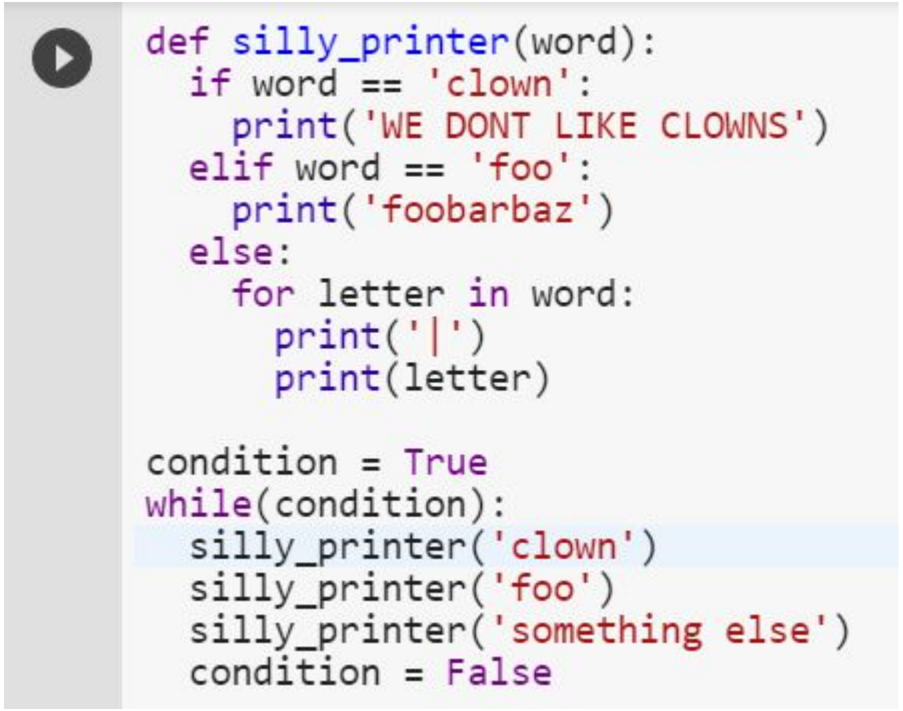


```
# Demo of control structures
# key words:
# def --> short for define, starts a block of code related to the definition of function
# for --> starts a for loop
# if --> evaluates a condition whether or not to enter a code block
# while --> similar to a for loop, continues a loop until a condition is false
```

## Python basics, control structures

- Most of the time, Google Colab will automagically indent correctly for your control blocks

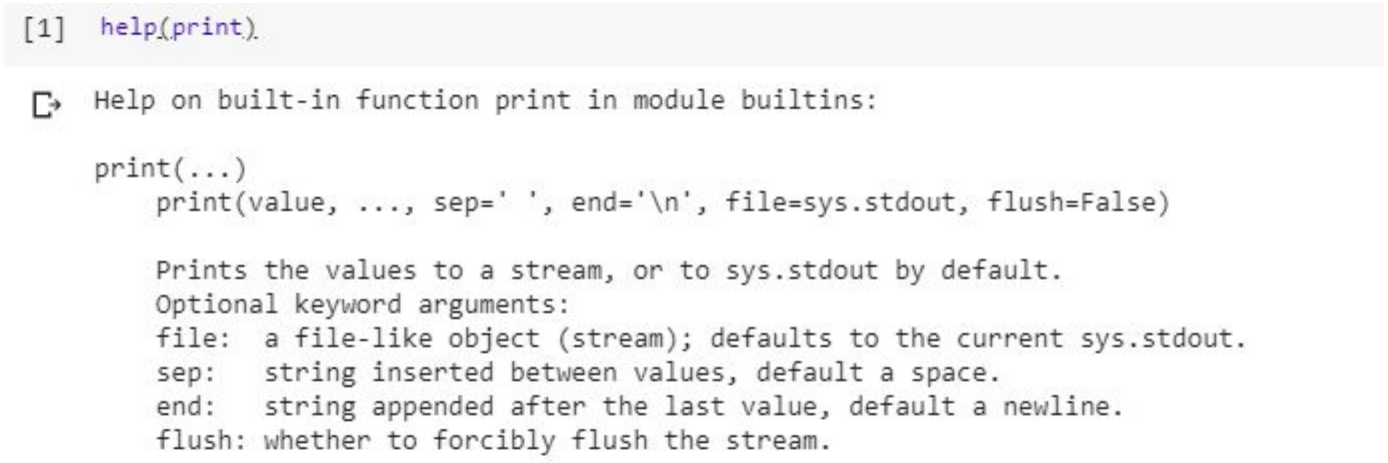
- The following code snippet uses most of what you'll need



```python
def silly_printer(word):
  if word == 'clown':
    print('WE DONT LIKE CLOWNS')
  elif word == 'foo':
    print('foobarbaz')
  else:
    for letter in word:
      print('|')
      print(letter)

condition = True
while(condition):
  silly_printer('clown')
  silly_printer('foo')
  silly_printer('something else')
  condition = False
```

## help

help(<function-name>) is a function that returns the help info on a function

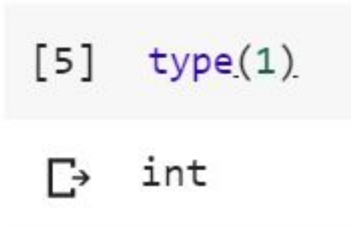


```
[1]  help(print)
```

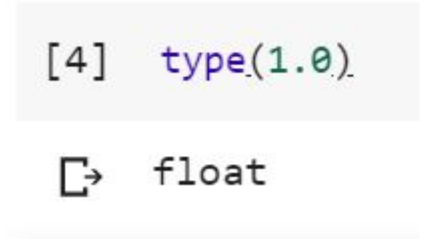
```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

## Data types

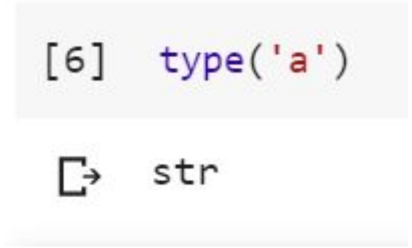
- Similar to other programming languages: integers, floating point numbers, strings/characters, and collections of primary data types



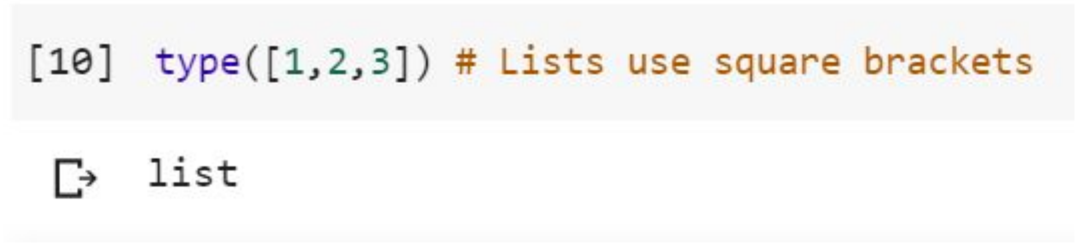
```
[5]  type(1)

     int
```



```
[4]  type(1.0)

     float
```



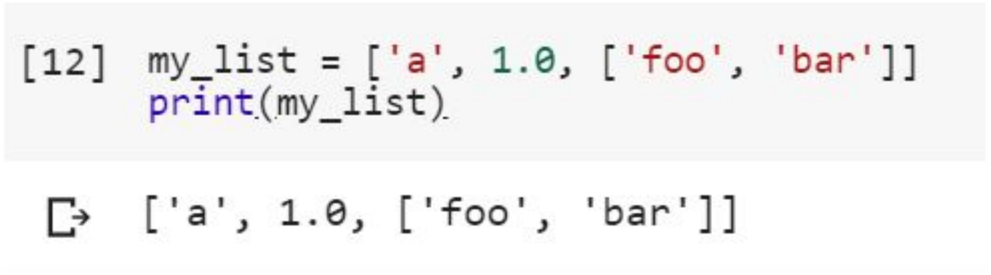
```
[6]  type('a')

     str
```



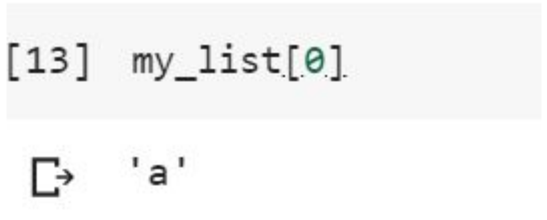
```
[10]  type([1,2,3]) # Lists use square brackets

      list
```

## Lists

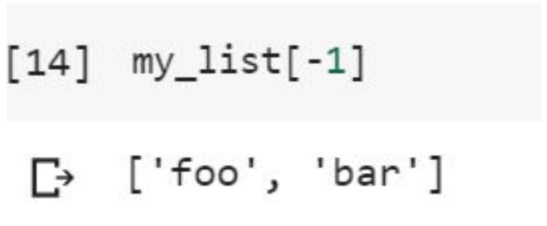
- Lists:
  - Mutable - can be changed
  - Ordered collection of any variety of data types
  - Can access list elements by their index
  - Can splice list elements by index (result is a new list)
  - Lists start at index 0 in Python



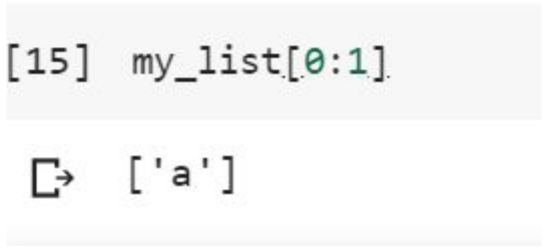
```
[12]  my_list = ['a', 1.0, ['foo', 'bar']]
      print(my_list)

      ['a', 1.0, ['foo', 'bar']]
```



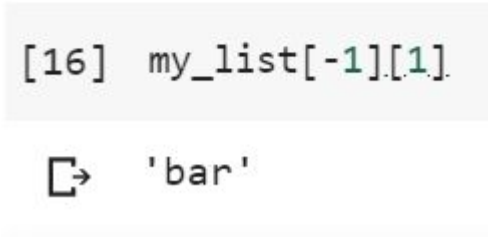
```
[13]  my_list[0]

      'a'
```



```
[14]  my_list[-1]

      ['foo', 'bar']
```

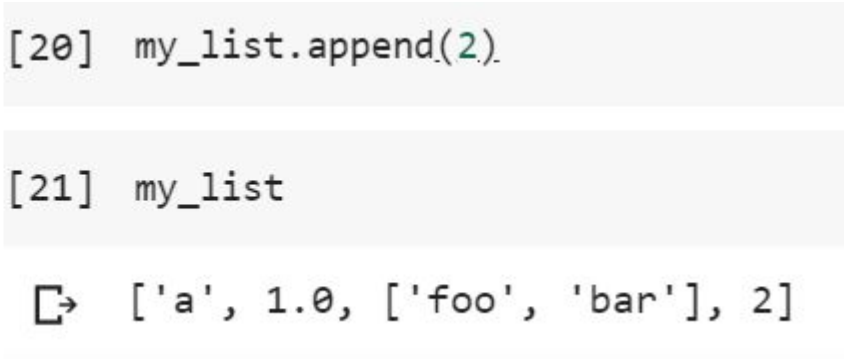


```
[15]  my_list[0:1]

      ['a']
```



```
[16]  my_list[-1][1]

      'bar'
```

## Lists have many built-in methods



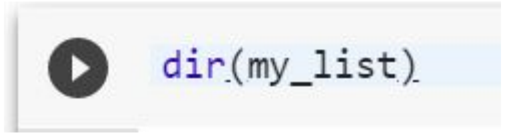
```
[20] my_list.append(2)
```

```
[21] my_list
```

```
['a', 1.0, ['foo', 'bar'], 2]
```

Need to refresh yourself on the list methods without using the internet?

The dir() method will list the applicable methods for an object



```
dir(my_list)
```

## dir('string')

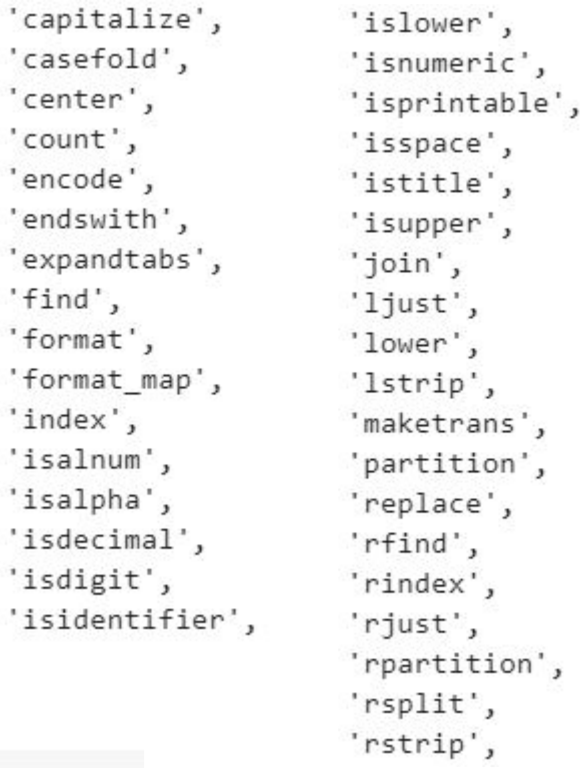
Many built in string methods

to make your life easier

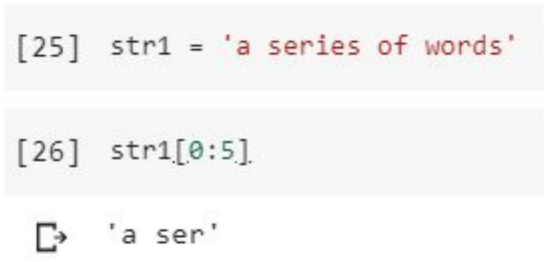
… at the end of the day

strings are just special lists of

characters



```
'capitalize',
'casefold',
'center',
'count',
'encode',
'endswith',
'expandtabs',
'find',
'format',
'format_map',
'index',
'isalnum',
'isalpha',
'isdecimal',
'isdigit',
'isidentifier',
'islower',
'isnumeric',
'isprintable',
'isspace',
'istitle',
'isupper',
'join',
'ljust',
'lower',
'lstrip',
'maketrans',
'partition',
'replace',
'rfind',
'rindex',
'rjust',
'rpartition',
'rsplit',
'rstrip',
```

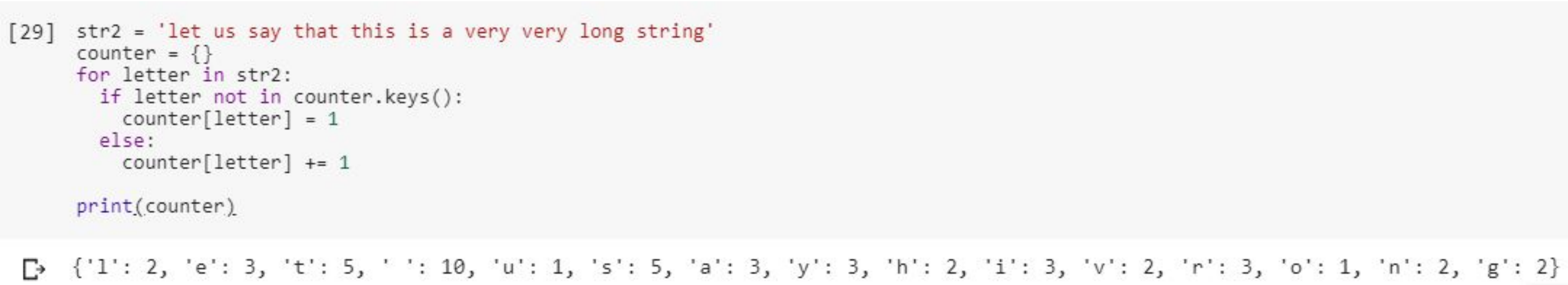


```
[25]  str1 = 'a series of words'

[26]  str1[0:5]

      'a ser'
```

## dictionaries

- Aside from lists, arguably the most important collection in Python
- Dictionaries are mappings of keys to values (a set of key : value pairs)
  - Keys are the index of a dictionary, they must be an immutable type
  - Values can be nearly any data type

# Utility of dictionaries

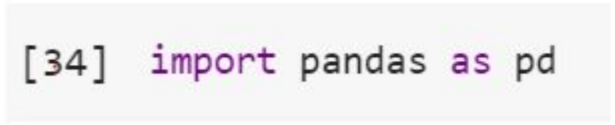


```
[29] str2 = 'let us say that this is a very very long string'
     counter = {}
     for letter in str2:
       if letter not in counter.keys():
         counter[letter] = 1
       else:
         counter[letter] += 1

     print(counter)
```

```
{'l': 2, 'e': 3, 't': 5, ' ': 10, 'u': 1, 's': 5, 'a': 3, 'y': 3, 'h': 2, 'i': 3, 'v': 2, 'r': 3, 'o': 1, 'n': 2, 'g': 2}
```

- counter.keys() returns the current set of dictionary keys, if we haven’t seen a letter yet, create an entry in the dictionary. Otherwise, increment the entry. We have a mechanism to count the letters in a list! ….hint for later

## Pandas - our first library



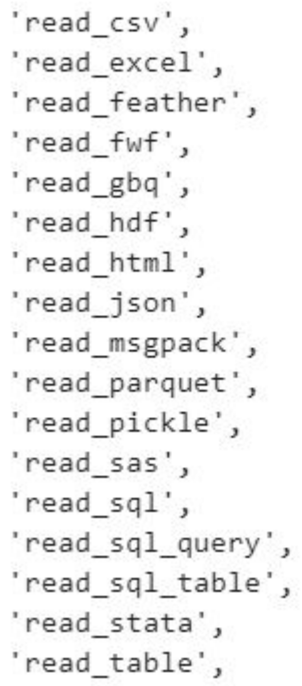
```
[34]  import pandas as pd
```

- We use libraries in programming languages so we don’t have to re-write code
- Pandas is a library for strong data manipulation
- Syntax: import <library>
- To give the library a “nickname” in your code,
  - import <something> as <nickname>
  - Do follow conventions, import pandas as pd
  - Don’t make silly nicknames
    - import pandas as p
    - p could stand for a lot of different libraries

## First thing, dir(pd)

- Some of the results are capitalized, those are Classes within the library
- Other results are lower case, those are methods that are accessible outside of objects

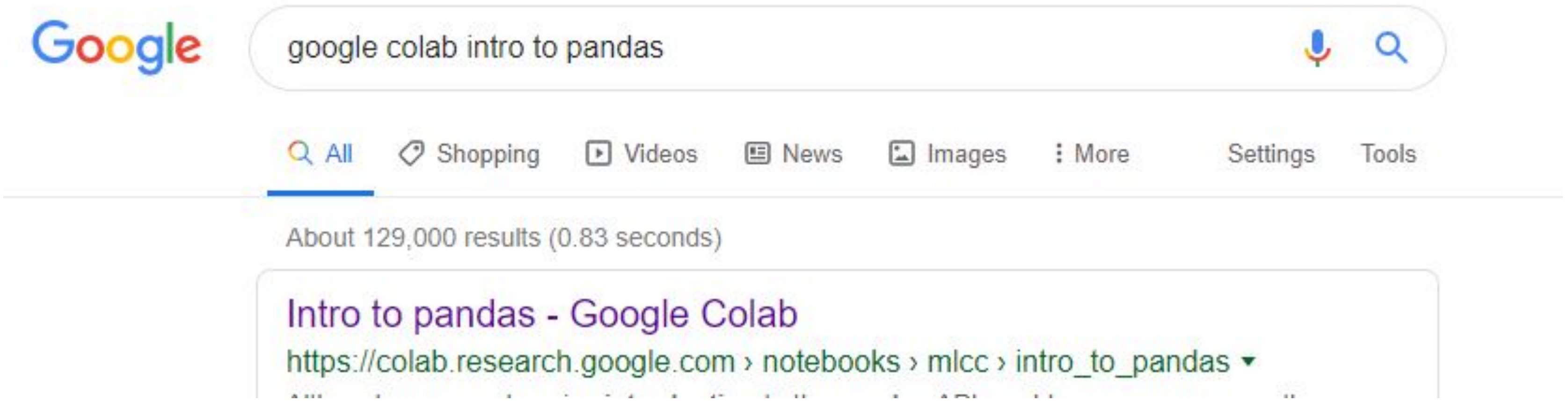
- Second thing, let’s get & read data



```
'read_csv',
'read_excel',
'read_feather',
'read_fwf',
'read_gbq',
'read_hdf',
'read_html',
'read_json',
'read_msgpack',
'read_parquet',
'read_pickle',
'read_sas',
'read_sql',
'read_sql_query',
'read_sql_table',
'read_stata',
'read_table',
```

# Intro to Pandas

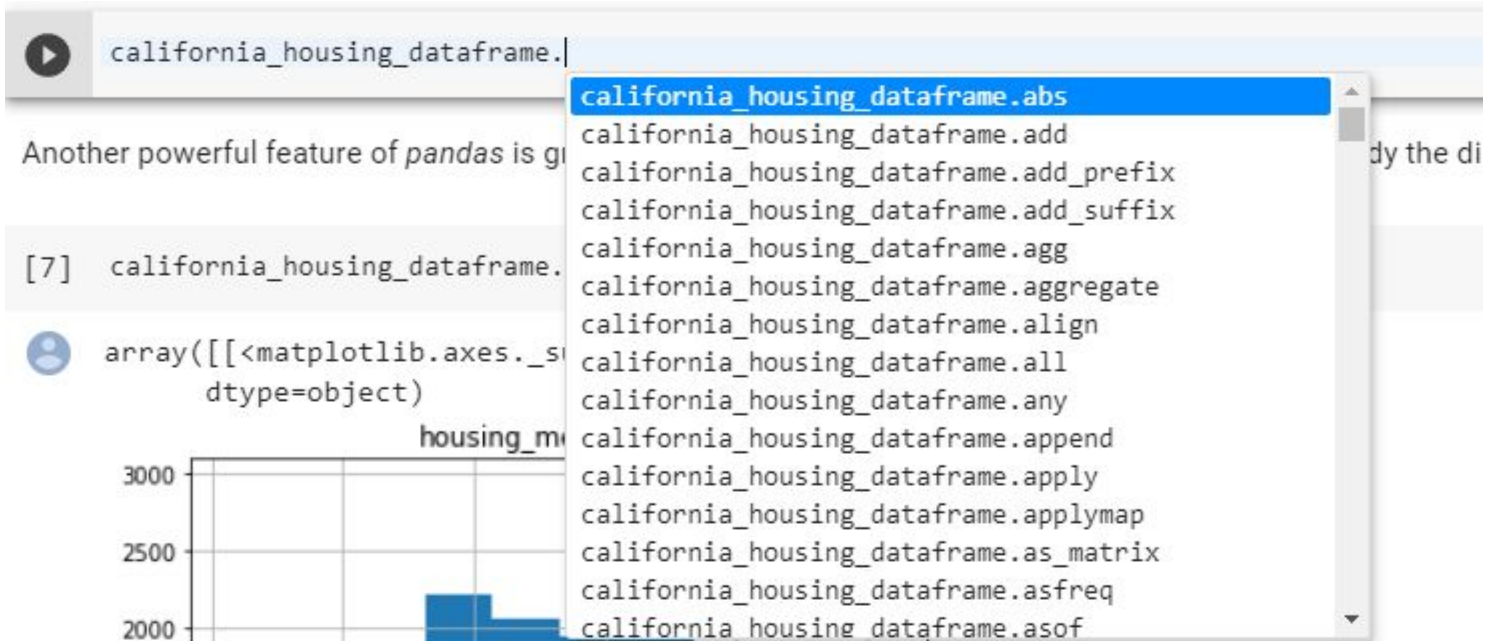
- Intro_to_pandas.ipynb
- Google it, open in Colab, save a copy, move to preferred location
- We'll go through it for a bit, feel free to move ahead



About 129,000 results (0.83 seconds)

Intro to pandas - Google Colab
https://colab.research.google.com › notebooks › mlcc › intro_to_pandas

# tab after period in Colab gives options



```
california_housing_dataframe.
```

```
california_housing_dataframe.abs
california_housing_dataframe.add
california_housing_dataframe.add_prefix
california_housing_dataframe.add_suffix
california_housing_dataframe.agg
california_housing_dataframe.aggregate
california_housing_dataframe.align
california_housing_dataframe.all
california_housing_dataframe.any
california_housing_dataframe.append
california_housing_dataframe.apply
california_housing_dataframe.applymap
california_housing_dataframe.as_matrix
california_housing_dataframe.asfreq
california_housing_dataframe.asof
```

Another powerful feature of *pandas* is gı … dy the di

```
[7]  california_housing_dataframe.
```

```
array([[<matplotlib.axes._su
       dtype=object)
```

## Typical flow of pandas

- Download a dataset
- Read the dataset into a dataframe
- df.head()
- df.describe()
- Think about what you need to do
  - Filtering, normalization, joining, manipulation, exploration ...

## Assignment 1

- Problem set, due 9/4 by 9PM
- Please record how long it takes you to finish it, will be helpful for designing further assignments
- Questions? If you have them now, ask away. Otherwise, email. We may cc the entire class if you ask a question that’s applicable.