Data Science & ML Course

Lesson #2 - From Platforms to Python Crash Course

Ivanovitch Silva

September, 2018

Agenda

- Part #01
  - Anaconda distributions (notebook & Jupyterlab & Google colab)
- Part #02
  - Python crash course
- Part #03
  - Dictionaries
  - Functions
  - Debugging errors

Update from repository

```
git clone https://github.com/ivanovitchm/datascience2machinelearning.git
```

Or ....

```
git pull
```

MODERN DATA SCIENTIST

Data Scientist, the sexiest job of 21th century requires a mixture of multidisciplinary skills ranging from an intersection of mathematics, statistics, computer science, communication and business. Finding a data scientist is hard. Finding people who understand who a data scientist is, is equally hard. So here is a little cheat sheet on who the modern data scientist really is.

MATH & STATISTICS

- Machine learning
- Statistical modeling
- Experiment design
- Bayesian inference
- Supervised learning: decision trees, random forests, logistic regression
- Unsupervised learning: clustering, dimensionality reduction
- Optimization: gradient descent and variants

PROGRAMMING & DATABASE

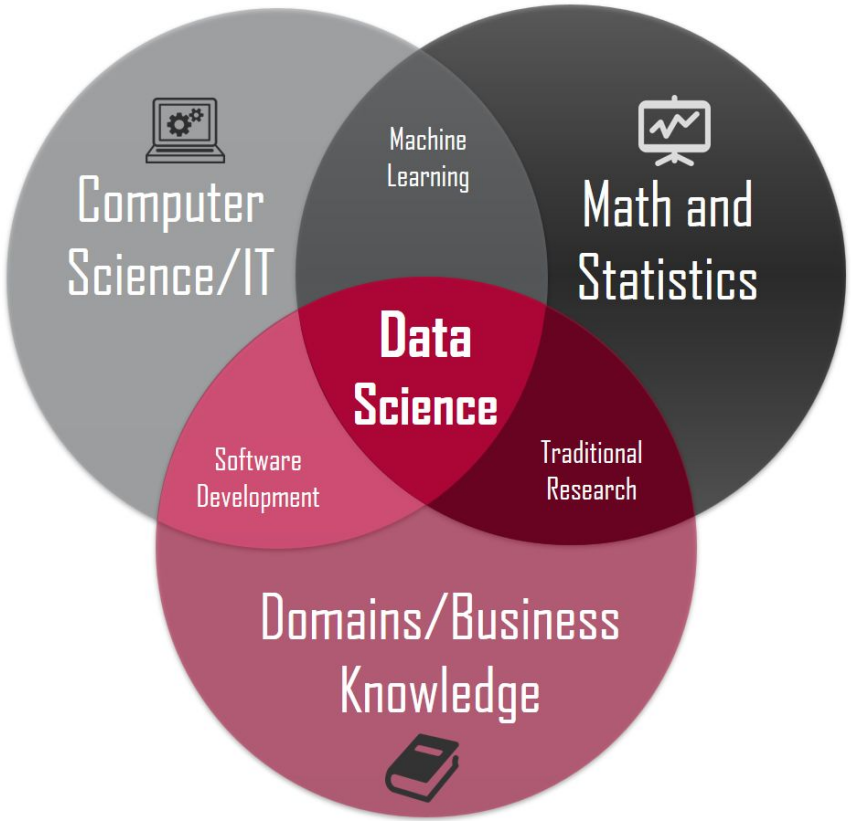
- Computer science fundamentals
- Scripting language e.g. Python
- Statistical computing package e.g. R
- Databases SQL and NoSQL
- Relational algebra
- Parallel databases and parallel query processing
- MapReduce concepts
- Hadoop and Hive/Pig
- Custom reducers
- Experience with xaaS like AWS

DOMAIN KNOWLEDGE & SOFT SKILLS

- Passionate about the business
- Curious about data
- Influence without authority
- Hacker mindset
- Problem solver
- Strategic, proactive, creative, innovative and collaborative

COMMUNICATION & VISUALIZATION

- Able to engage with senior management
- Story telling skills
- Translate data-driven insights into decisions and actions
- Visual art design
- R packages like ggplot or lattice
- Knowledge of any of visualization tools e.g. Flare, D3.js, Tableau

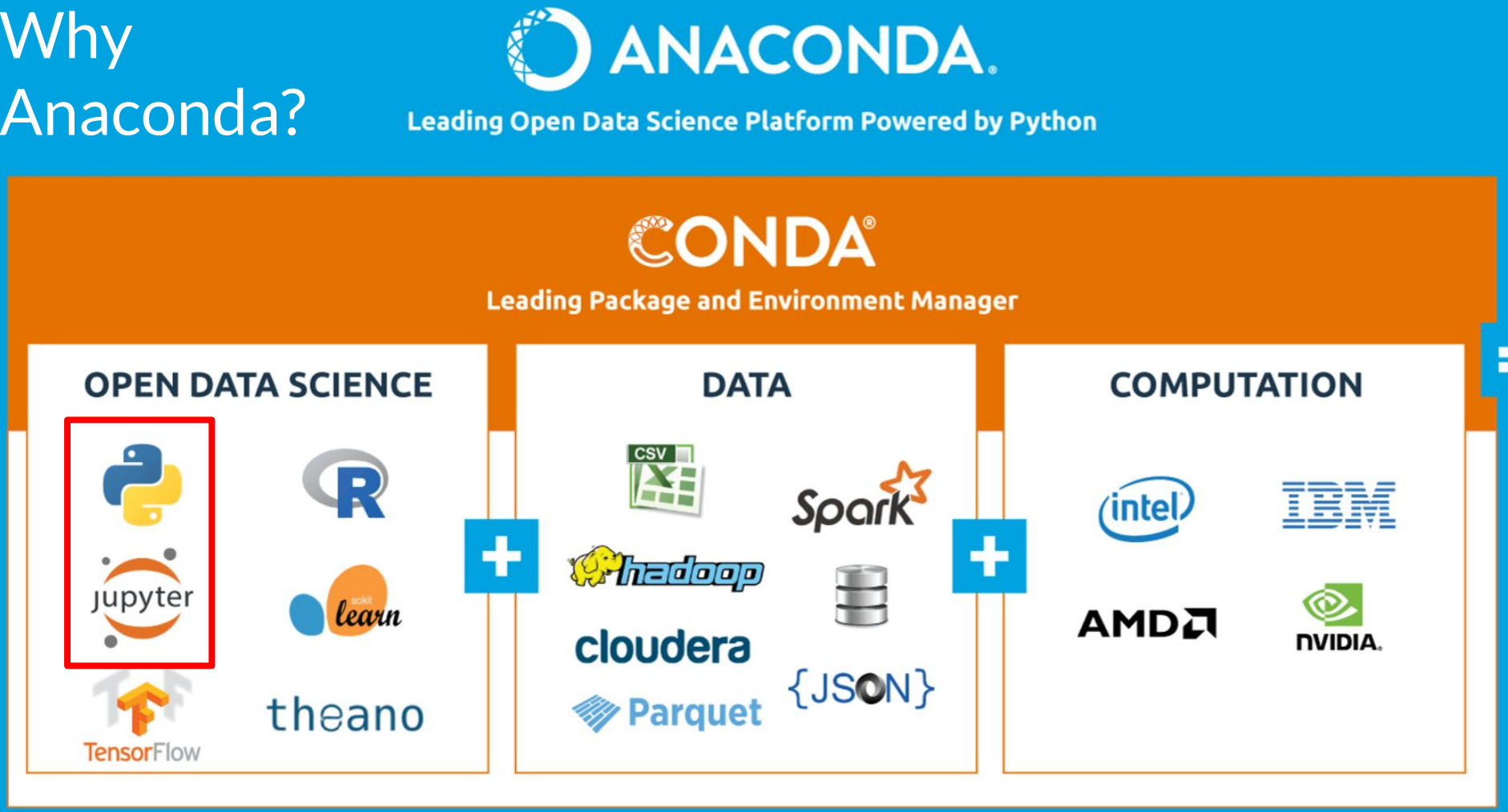


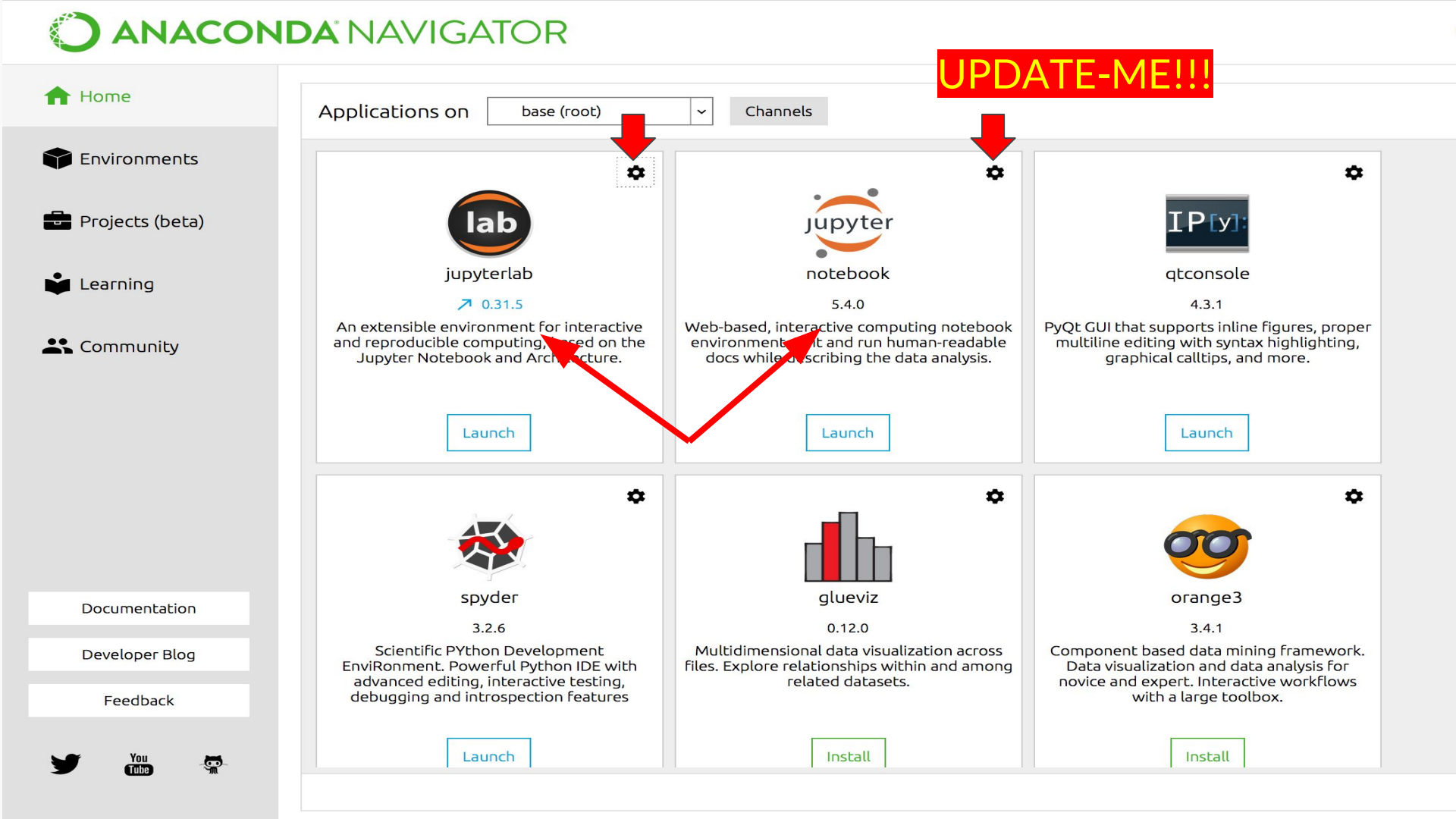
https://goo.gl/VKYfXn

Version 5.2
Release Date: May 30, 2018

https://www.continuum.io/downloads

Why Anaconda?




ANACONDA NAVIGATOR
UPDATE-ME!!!
Home
Environments
Projects (beta)
Learning
Community
Documentation
Developer Blog
Feedback
Applications on
base (root)
Channels
jupyterlab
0.31.5
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
notebook
5.4.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
spyder
3.2.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
glueviz
0.12.0
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
orange3
3.4.1
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install

Extensions

Fundamentally, JupyterLab is designed as an extensible environment. JupyterLab extensions can customize or enhance any part of JupyterLab. They can provide new themes, file viewers and editors, or renderers for rich outputs in notebooks. Extensions can add items to the menu or command palette, keyboard shortcuts, or settings in the settings system. Extensions can provide an API for other extensions to use and can depend on other extensions. In fact, the whole of JupyterLab itself is simply a collection of extensions that are no more powerful or privileged than any custom extension.

JupyterLab extensions are npm packages (the standard package format in Javascript development). There are many community-developed extensions being built on GitHub. You can search for the GitHub topic jupyterlab-extension to find extensions. For information about developing extensions, see the developer documentation.

Find other extensions

Dependence to install extensions in JupyterLab

```
$ conda install -c conda-forge nodejs
```

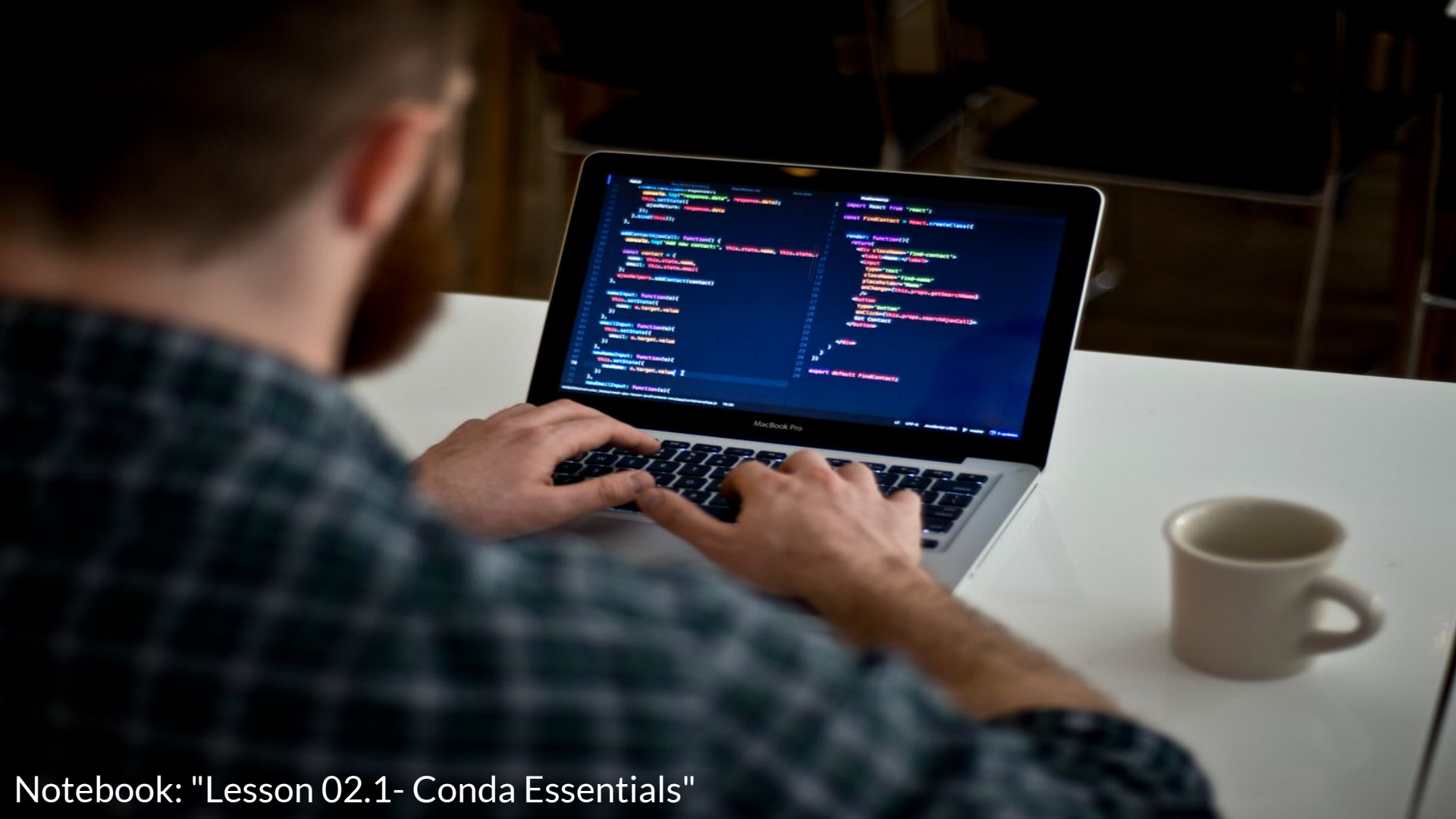
Install extensions

```
$ jupyter labextension install @jupyterlab/toc  @jupyterlab/git  my-extension
```

Uninstall extensions

```
$ jupyter labextension uninstall @jupyterlab/git
```

https://jupyterlab.readthedocs.io/en/stable/user/extensions.html



Notebook: "Lesson 02.1- Conda Essentials"

Python Beginner

- Python basic
- Files and Loops
- Boolean and If statements
- List operations
- Challenges

crimes_rates.csv
dq_unisex_names.csv
la_weather.csv

Python basic

```
import sys
print(sys.version)
```

```
3.6.5 |Anaconda, Inc.| (default, Apr 26 2018, 08:42:37)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
```

Operators	Example expression
Addition +	>>> 4 +5 9
Subtraction -	>>> 4 - 5 - 1
Multiplication *	>>> 4 * 5 20
Division /	>>> 4 / 5 0.8
Exponent **	>>> 4 ** 5 1024
Parentheses ()	>>> (4 ** 5) 1024

Using a list to store multiple values

```
[ ]  cities.append("Albuquerque")
     cities.append("Anaheim")
     print(cities)
     print(type(cities))
```

```
['Albuquerque', 'Anaheim']
<class 'list'>
```

Each time we call **list.append()**, the values in the **list cities** are updated.

Accessing elements in a list

crime_rates

index	0	1	2	3	4
values	749	371	828	503	1379

```
crime_rates = [749, 371, 828, 503, 1379]
first_value = crime_rates[0]
second_value = crime_rates[1]
fifth_value = crime_rates[4]
```

Slicing

```
[ ]  crime_rates = [749, 371, 828, 503, 1379]
     # The following slice selects values at index 2 and 3, but not 4.
     two_four = crime_rates[2:4]
     two_four
```

```
[828, 503]
```

Here's a diagram of the same slice:

crime_rates

index	0	1	2	3	4
values	749	371	828	503	1379

crime_rates[2:4]

2	3
828	503

Handling files

```
# Code from previous cells
f = open('crime_rates.csv', 'r')
data = f.read()
data
```

```
'Albuquerque,749\nAnaheim,371\nAnchorage,828\n
```

"data" is a long string

```
data_list = data.split("\n")
data_list
```

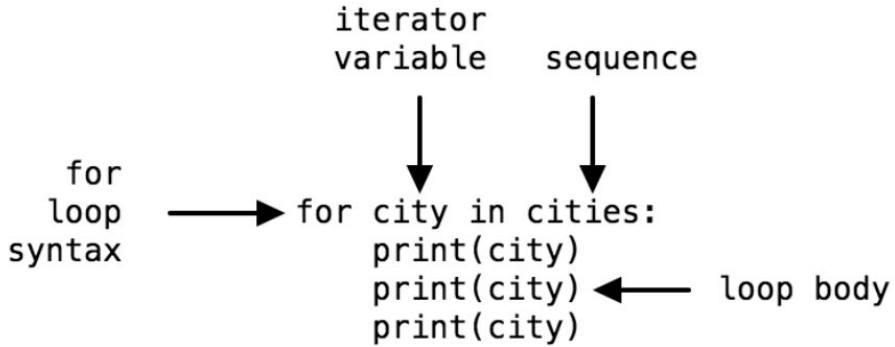
```
['Albuquerque,749',
 'Anaheim,371',
 'Anchorage,828',
 'Arlington,503',
 'Atlanta,1379',
 'Aurora,425',
```

"data_list" list of strings

•••

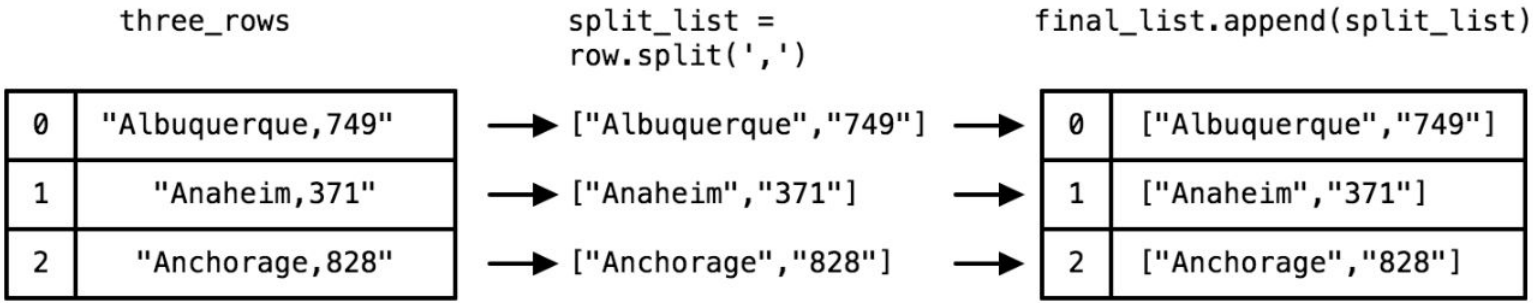
Looping structure

```
for city in cities:
    print(city)
    print(city)
    print(city)
```



List of Lists

```
three_rows = ["Albuquerque,749", "Anaheim,371", "Anchorage,828"]
final_list = []
for row in three_rows:
    split_list = row.split(',')
    final_list.append(split_list)
print(final_list)
```



Accessing elements in a list of lists

```
first_list = final_data[0] # Returns the first list: ['Albuquerque', '749'].
first_list_first_value = first_list[0] # Returns the first list's first element: 'Albuquerque'.
```

final_data
(list of lists)

0	['Albuquerque','749']
1	['Anaheim', '371']
2	['Anchorage', '828']

final_data[0]
(list)

0	'Albuquerque'
1	'749'

final_data[0][0]
(string)

'Albuquerque'

Accessing elements in a list of lists

final_data
(list of lists)

0	['Albuquerque','749']
1	['Anaheim', '371']
2	['Anchorage', '828']

final_data[0]
(list)

0	'Albuquerque'
1	'749'

final_data[0][0]
(string)

'Albuquerque'

```
crime_rates = []
for row in five_elements:
    crime_rate = row[1] # row is a list variable, not a string.
    crime_rates.append(crime_rate)  # crime_rate is a string, the crime rate of the city
```

Boolean variables

```
t = True
f = False
```

```
# True
t = (8 == 8)
# False
u = (8 != 8)
```

```
"8" == "8"
["January", "February"] == ["January", "February"]
5.0 == 5.0
```

```
rates = [10, 15, 20]
rates[0] > rates[1] # False
rates[0] >= rates[0] # True
```

```
rates = [10, 15, 20]
rates[0] < rates[1] # True
rates[0] <= rates[0] # True
```

Conditional structures

```
sample_rate = 749
greater = (sample_rate > 5)
if greater:
    print(sample_rate)
```

```
t = True
f = False
if t:
    print("Now you see me")
if f:
    print("Now you don't")
```

```
sample_rate = 749
greater = (sample_rate > 5)
if greater:
    print(sample_rate)
```

Notebook: "Lesson #02.2 - Python Beginner.ipynb"
Up to section 3.7

Challenge #01

JUN. 10, 2015, AT 9:16 AM

The Most Common Unisex Names In America: Is Yours One Of Them?

By Andrew Flowers

Filed under Names

Get the data on GitHub

FiveThirtyEight

```
Casey,176544.328149
Riley,154860.66517300002
Jessie,136381.830656
Jackie,132928.78874000002
Avery,121797.41951600001
Jaime,109870.18729000002
Peyton,94896.39521599999
```

Estimate number of Americans

Challenge #2

```
Day,Type of Weather
1,Sunny
2,Sunny
3,Sunny
4,Sunny
5,Sunny
6,Rain
7,Sunny
8,Sunny
9,Fog
10,Rain
```

Los Angeles weather data from 2014

DATAQUEST

Notebook: "Lesson #02.2 - Python Beginner.ipynb"
Sections 4 & 5


Break

Agenda

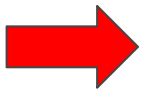
- Dictionaries
- Functions
- Debugging errors
- Challenges

movie_metadata.csv
births.csv
la_weather.csv

Dictionaries - Motivation

Student	Score
Tom	70
Jim	80
Sue	85
Ann	75

```
students = ["Tom", "Jim", "Sue", "Ann"]
scores = [70, 80, 85, 75]
```

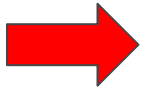


What is the Sue's score?

Dictionaries - Motivation

```
indexes = [0,1,2,3]
name = "Sue"
score = 0
for i in indexes:
    if students[i] == name:
        score = scores[i]
print(score)
```

```
students = ["Tom", "Jim", "Sue", "Ann"]
scores = [70, 80, 85, 75]
```



What is the Sue's score?

Dictionaries

```
students = {
    "Tom": 60,
    "Jim": 70,
    "Sue": 85,
    "Ann": 80
}
```

```
students["Sue"]
```

```
85
```

Using dictionaries as a counter structure

```
# the dataset
pantry = ["apple", "orange", "grape", "apple", "orange",
          "apple", "tomato", "potato", "grape"]

# empty dictionary
pantry_counts = {}

for item in pantry:
  if item in pantry_counts:
    pantry_counts[item] += 1
  else:
    pantry_counts[item] = 1

pantry_counts
```

```
{'apple': 3, 'grape': 2, 'orange': 2, 'potato': 1, 'tomato': 1}
```

Introduction to Functions

movie_title	director_name	color	duration	actor_1_name	language	country	title_year
Avatar	James Cameron	Color	178	CCH Pounder	English	USA	2009
Pirates of the Caribbean: At the World's End	Gore Verbinski	Color	169	Johnny Depp	English	USA	2007
Spectre	Sam Mendes	Color	148	Christoph Waltz	English	UK	2015
The Dark Knight Rises	Christopher Nolan	Color	164	Tom Hardy	English	USA	2012
Star Wars VII: The Force Awakens	JJ Abrams	Color	136	Harrison Ford	English	USA	2015

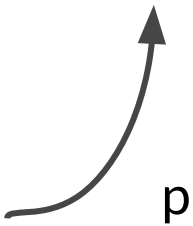
Data about IMDb

Introduction to Functions

```
>> movie_data = parser(movie_metadata)
>> print(movie_data[0:5])
[['movie_title', 'director_name', 'color', 'duration', 'actor_1_name', 'language', 'country',
'title_year'], ['Avatar', 'James Cameron', 'Color', '178', 'CCH Pounder', 'English', 'USA', '2009'],
["Pirates of the Caribbean: At World's End", 'Gore Verbinski', 'Color', '169', 'Johnny Depp', 'English',
'USA', '2007'], ['Spectre', 'Sam Mendes', 'Color', '148', 'Christoph Waltz', 'English', 'UK', '2015'],
['The Dark Knight Rises', 'Christopher Nolan', 'Color', '164', 'Tom Hardy', 'English', 'USA', '2012']]
```

movie_metadata.csv

movie_title	director_name	color	duration	actor_1_name	language	country	title_year
Avatar	James Cameron	Color	178	CCH Pounder	English	USA	2009
Pirates of the Caribbean: At the World's End	Gore Verbinski	Color	169	Johnny Depp	English	USA	2007
Spectre	Sam Mendes	Color	148	Christoph Waltz	English	UK	2015
The Dark Knight Rises	Christopher Nolan	Color	164	Tom Hardy	English	USA	2012
Star Wars VII: The Force Awakens	JJ Abrams	Color	136	Harrison Ford	English	USA	2015



parser()

Functions

```
def counter(input_lst,header_row = False):
    num_elt = 0
    if header_row == True:
        input_lst = input_lst[1:len(input_lst)]
    for each in input_lst:
        num_elt = num_elt + 1
    return num_elt
```

```
>> print(counter(movie_data))
4933
>> print(counter(movie_data, True))
4932
```

```
[['movie_title', 'director_name', 'color', 'duration', 'actor_1_name', 'language', 'country',
'title_year'], ['Avatar', 'James Cameron', 'Color', '178', 'CCH Pounder', 'English', 'USA', '2009'],
["Pirates of the Caribbean: At World's End", 'Gore Verbinski', 'Color', '169', 'Johnny Depp', 'English',
'USA', '2007'], ['Spectre', 'Sam Mendes', 'Color', '148', 'Christoph Waltz', 'English', 'UK', '2015'],
['The Dark Knight Rises', 'Christopher Nolan', 'Color', '164', 'Tom Hardy', 'English', 'USA', '2012']]
```

Calling a functions inside another function

```
>> lists = [["dog","cat","rabbit"],[1,2,3,4],[True]]
>> list_count = (list_counter(lists))
>> print(list_count)
[3,4,1]
```

```
def list_counter(input_lst):
    final_list = []
    for each in input_lst:
        num_elt = counter(each)
        final_list.append(num_elt)
    return final_list
```

Debugging Errors

```
the_answer = "42
```

```
  File "<ipython-input-2-85ffad3b5465>", line 1
    the_answer = "42
                 ^
SyntaxError: EOL while scanning string literal
```

SEARCH STACK OVERFLOW

```
def find():
    print("42")
     print("what, really?")
```

```
  File "<ipython-input-4-dd6a6ca22a8f>", line 3
    print("what, really?")
    ^
IndentationError: unexpected indent
```

SEARCH STACK OVERFLOW

```
# `def` keyword misspelled as `de`.
de find():
    print("42")
```

```
  File "<ipython-input-3-3ae0fdd21c4a>", line 1
    de find():
       ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

Breathe()
Count(5)
Read()

Debugging Errors

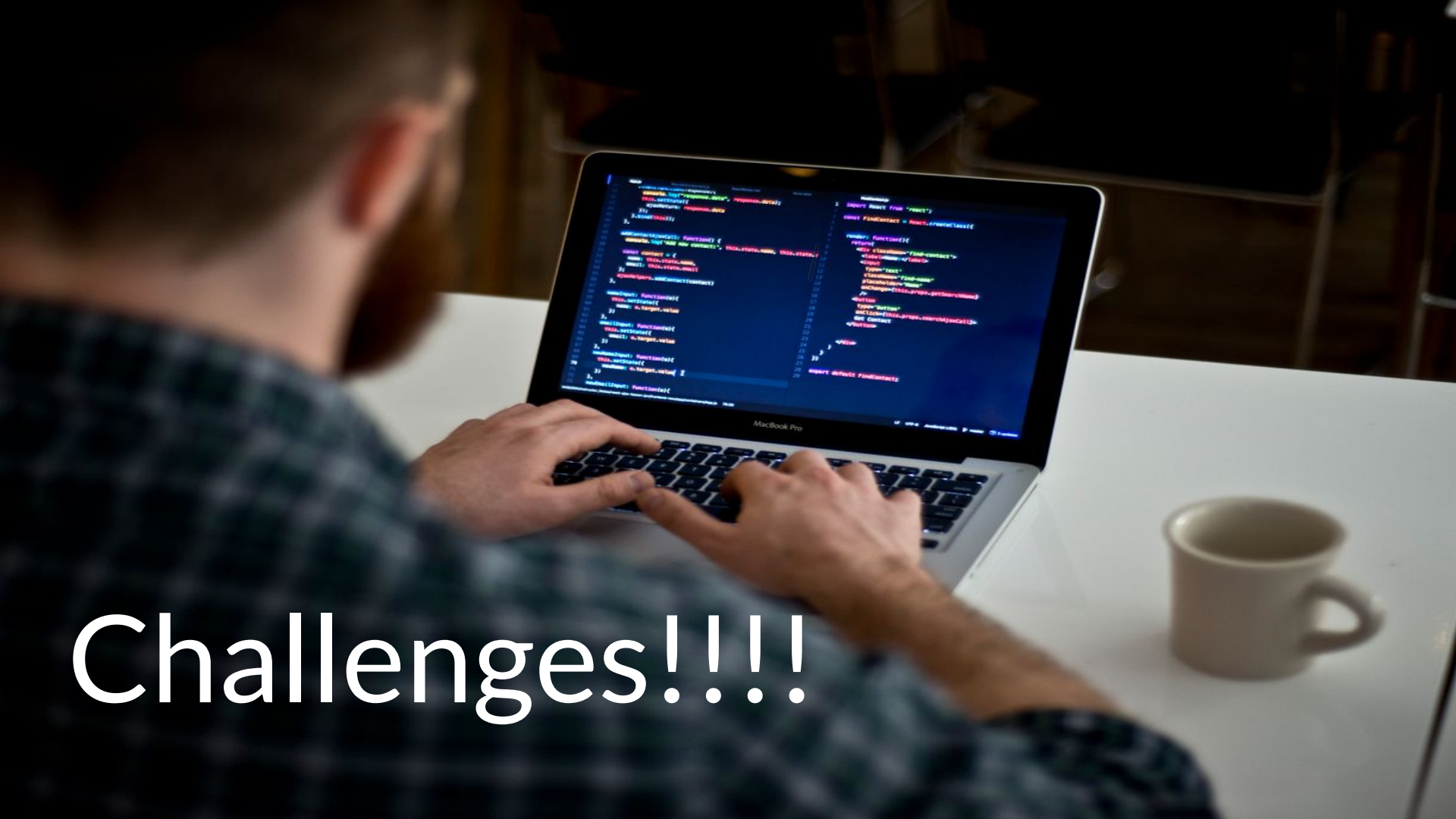
```
# Default code containing errors
lives = [1,2,3]
lives[4]

f = open("story.txt")
f.split(" ")
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-17-f302f35fc49a> in <module>()
      1 lives = [1,2,3]
----> 2 lives[4]
      3
      4 f = open("story.txt")
      5 f.split(" ")

IndexError: list index out of range
```

SEARCH STACK OVERFLOW


Challenges!!!!

Calculating the number of births each day of week, month

MAY 13, 2016, AT 12:21 PM

Some People Are Too Superstitious To Have A Baby On Friday The 13th

By Carl Bialik

Filed under Parenting

Get the data on GitHub

Thousands of babies are born in the U.S. whenever Friday falls on the 13th of the month — but about 800 fewer than you'd expect if parents and the doctors who deliver their newborns treated it like any other day.

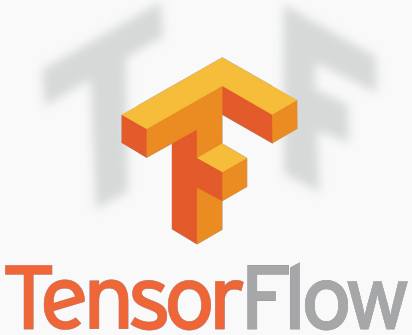
Many births are scheduled, either as induced deliveries or cesarean section. And given the choice, lots of parents would rather not take their chance with a date that delivers a double whammy of superstitious bad luck, tying together longstanding fears about Fridays and the number 13.

Births on the 13th of the month are lower than you'd expect, but especially on Fridays; the effect is smallest when the 13th falls on a weekend, when delivery wards are staffed more thinly and tend to schedule fewer births.[1]

Challenge (...)

https://github.com/tensorflow/tensorflow

```
git log --encoding=latin-1 --pretty="%at,%aN" > log.csv
```



How many people contributed to the project?
Top #10 contributors?