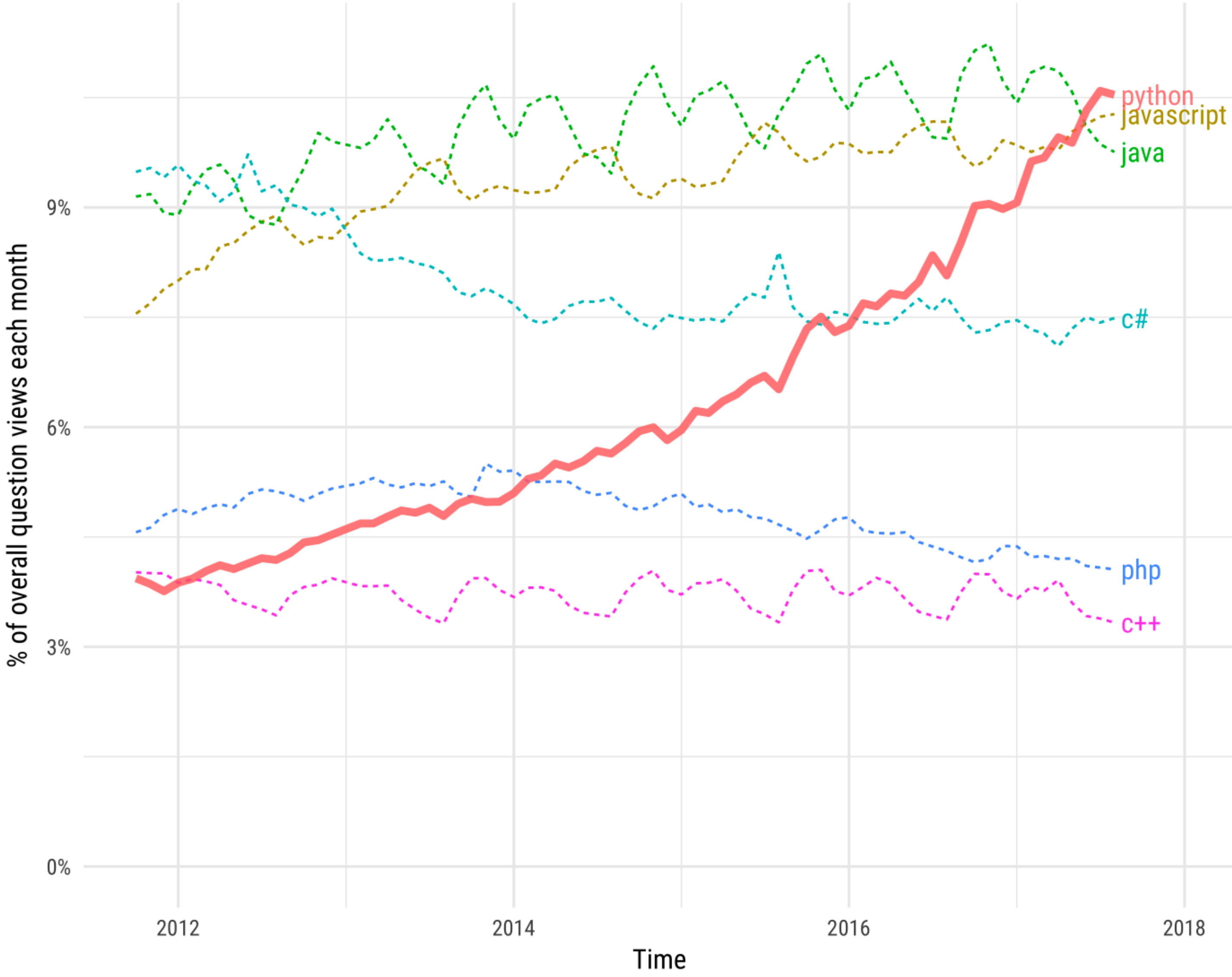
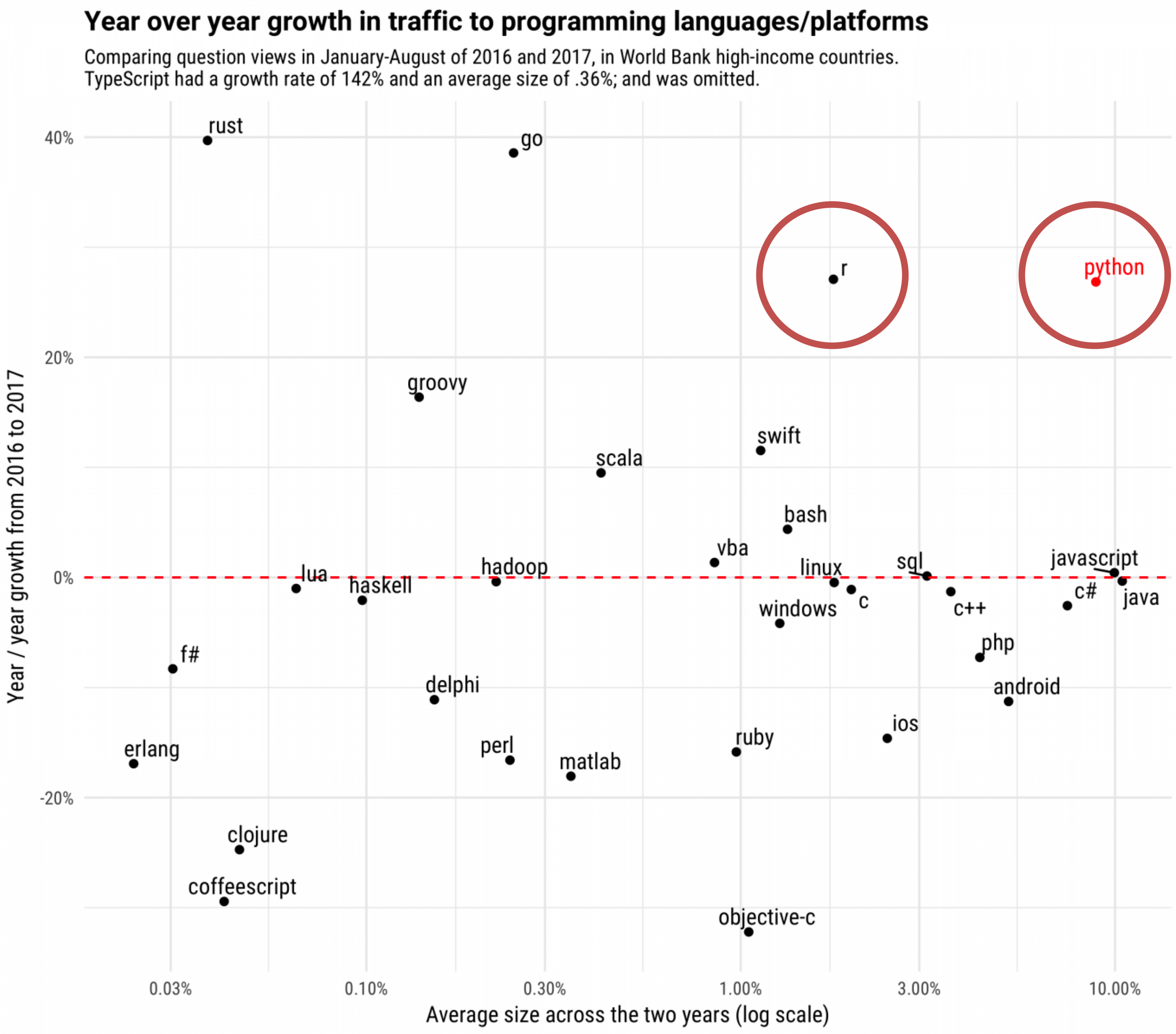
# Part II: Python

**Growth of major programming languages**

Based on Stack Overflow question views in World Bank high-income countries



Stackoverflow blog, September 2017



Stackoverflow blog, September 2017

# Python has many applications

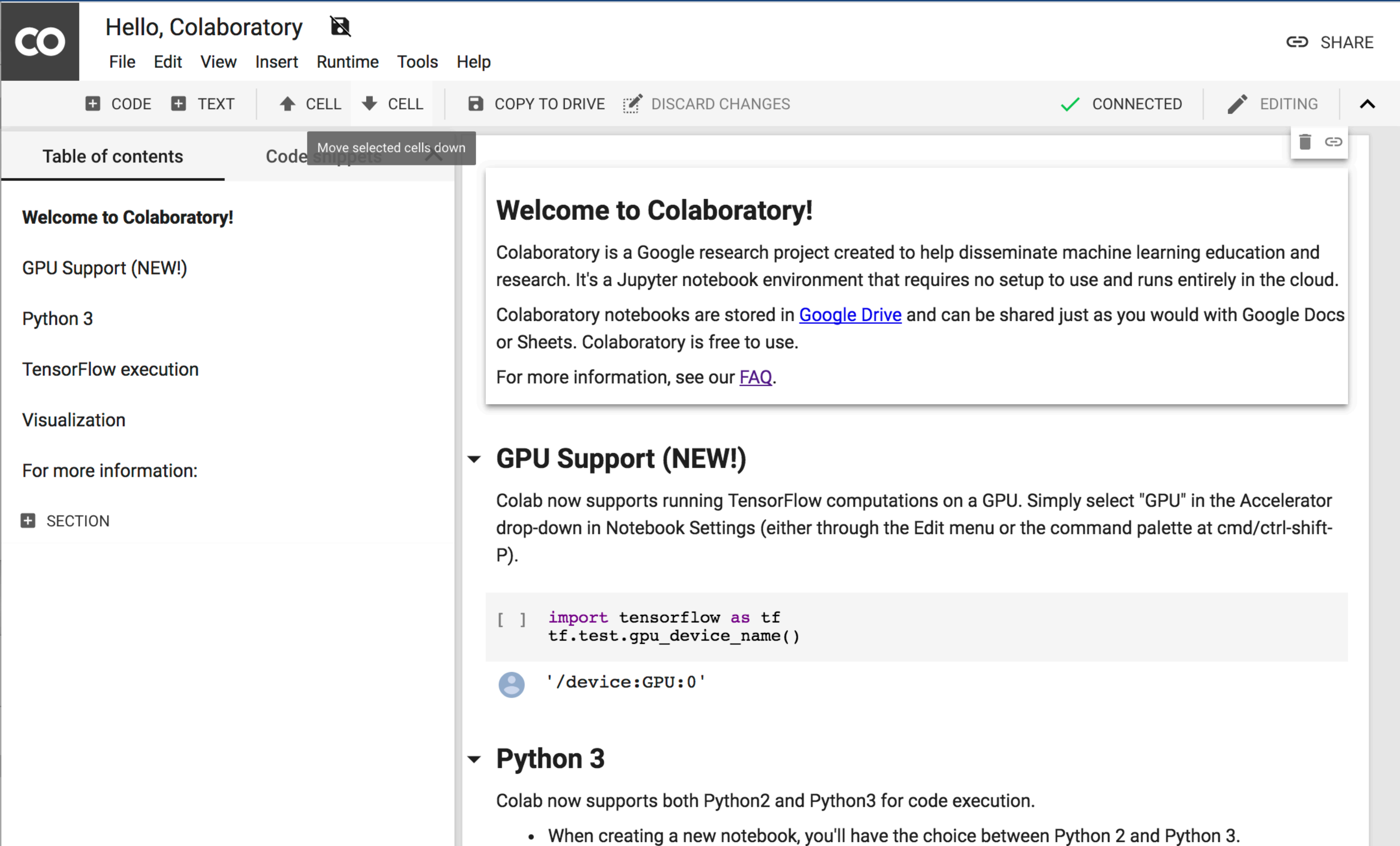
- Web development
- Application development
- Computer graphics
- Scientific computing
  - Bioinformatics
  - Machine learning
  - Simulations

https://www.python.org/about/quotes/

# Three alternatives to get Python

- Google Colaboratory (free) https://colab.research.google.com/
- Anaconda (free, ~1.5GB of space required)
- CoCalc ($14/month, can try for free)

# Google Colaboratory



## Welcome to Colaboratory!

Colaboratory is a Google research project created to help disseminate machine learning education and research. It's a Jupyter notebook environment that requires no setup to use and runs entirely in the cloud.

Colaboratory notebooks are stored in Google Drive and can be shared just as you would with Google Docs or Sheets. Colaboratory is free to use.

For more information, see our FAQ.

## GPU Support (NEW!)

Colab now supports running TensorFlow computations on a GPU. Simply select "GPU" in the Accelerator drop-down in Notebook Settings (either through the Edit menu or the command palette at cmd/ctrl-shift-P).

```
[ ]  import tensorflow as tf
     tf.test.gpu_device_name()
```

```
'/device:GPU:0'
```

## Python 3

Colab now supports both Python2 and Python3 for code execution.

- When creating a new notebook, you'll have the choice between Python 2 and Python 3.

# CoCalc

Collaborative Calculation in the Cloud

or sign in with your account

Online computing environment

# Anaconda

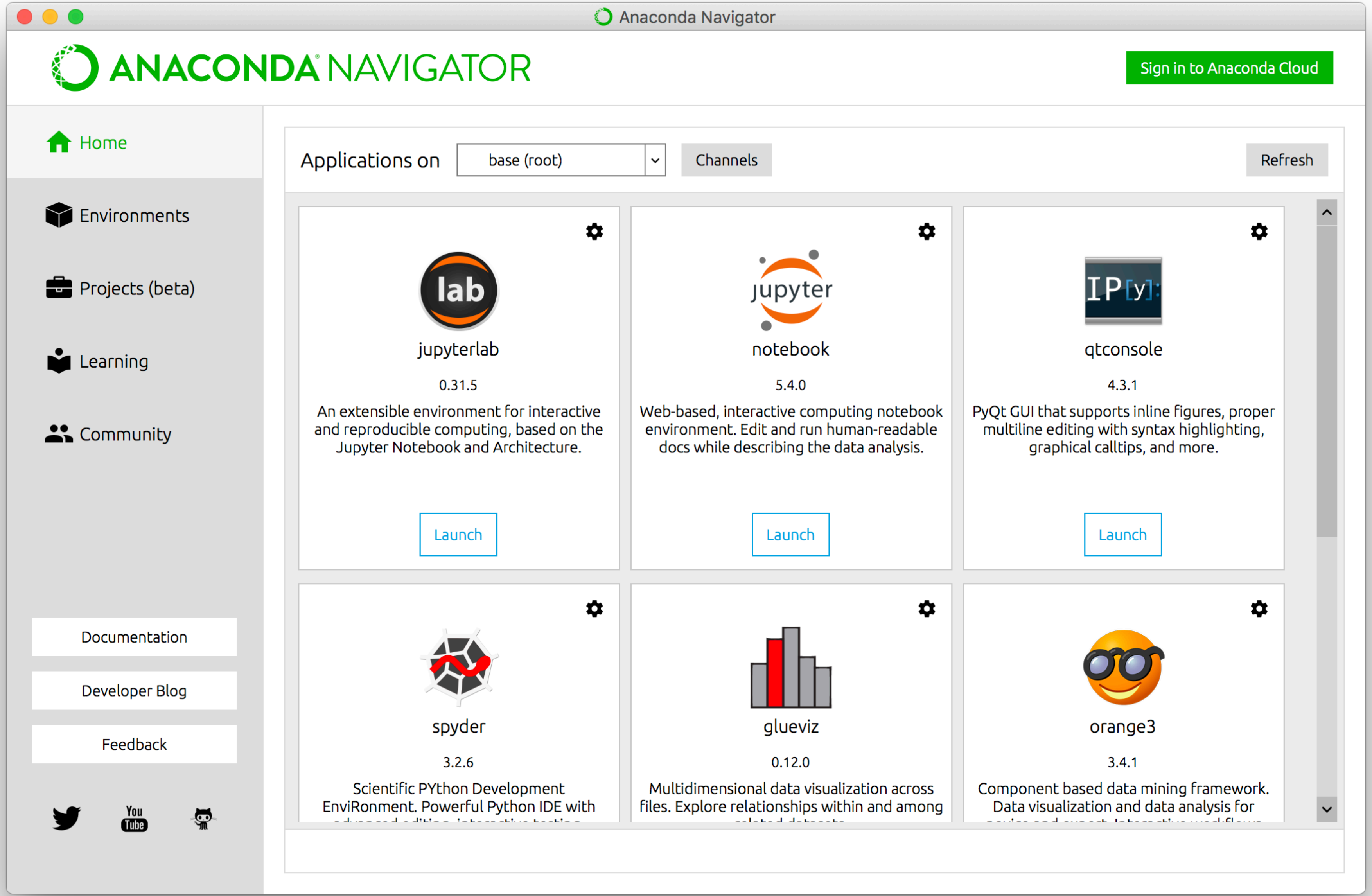
PYTHON
THE FASTEST
GROWING
OPEN DATA
SCIENCE
PLATFORM

ANACONDA®

Modern open source analytics platform
powered by Python

DOWNLOAD FOR FREE

http://www.continuum.io



An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

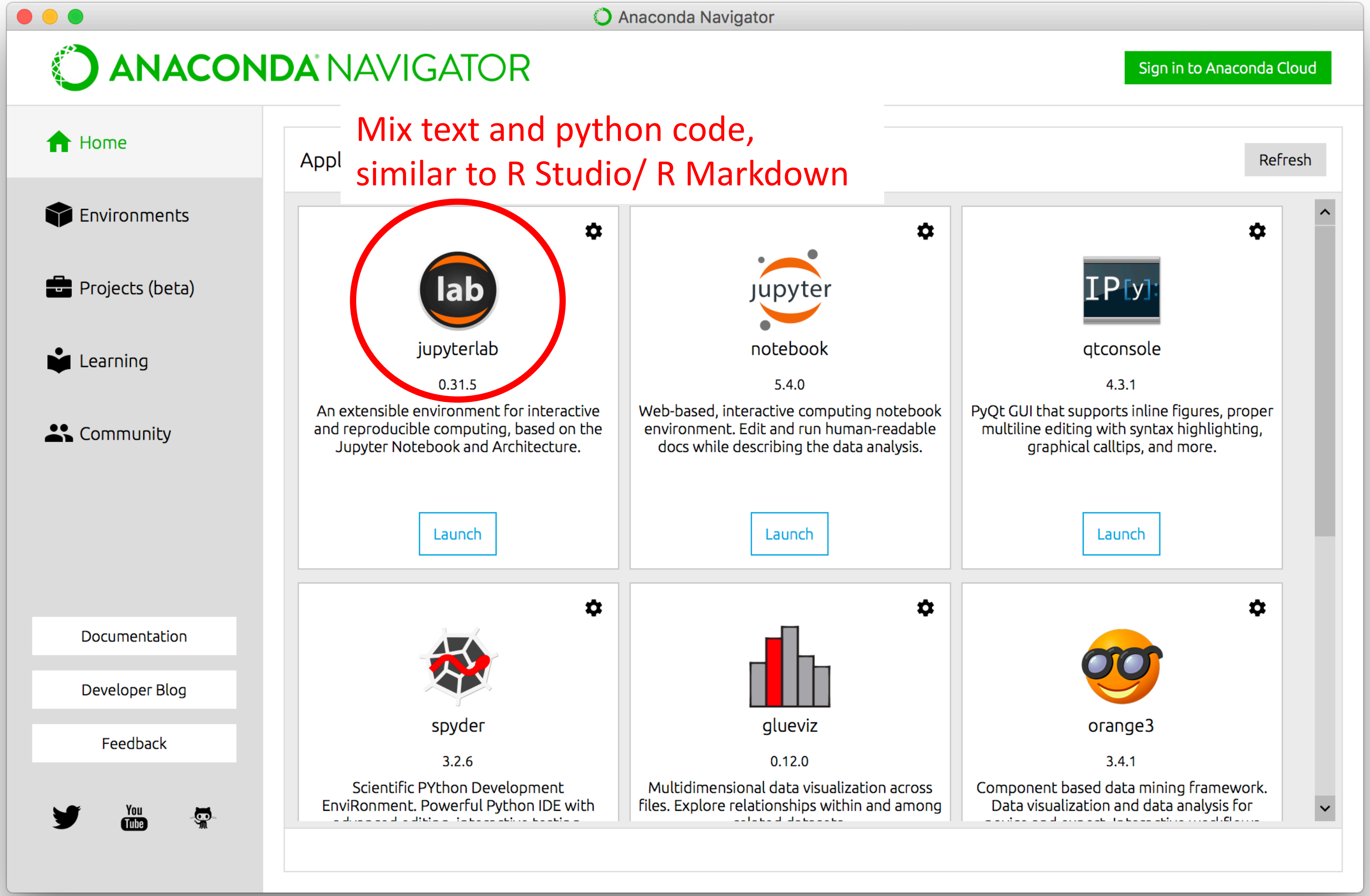
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Scientific PYthon Development EnviRonment. Powerful Python IDE with

Multidimensional data visualization across files. Explore relationships within and among

Component based data mining framework. Data visualization and data analysis for



Mix text and python code, similar to R Studio/ R Markdown

jupyterlab

0.31.5

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

notebook

5.4.0

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

qtconsole

4.3.1

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

spyder

3.2.6

Scientific PYthon Development EnviRonment. Powerful Python IDE with

glueviz

0.12.0

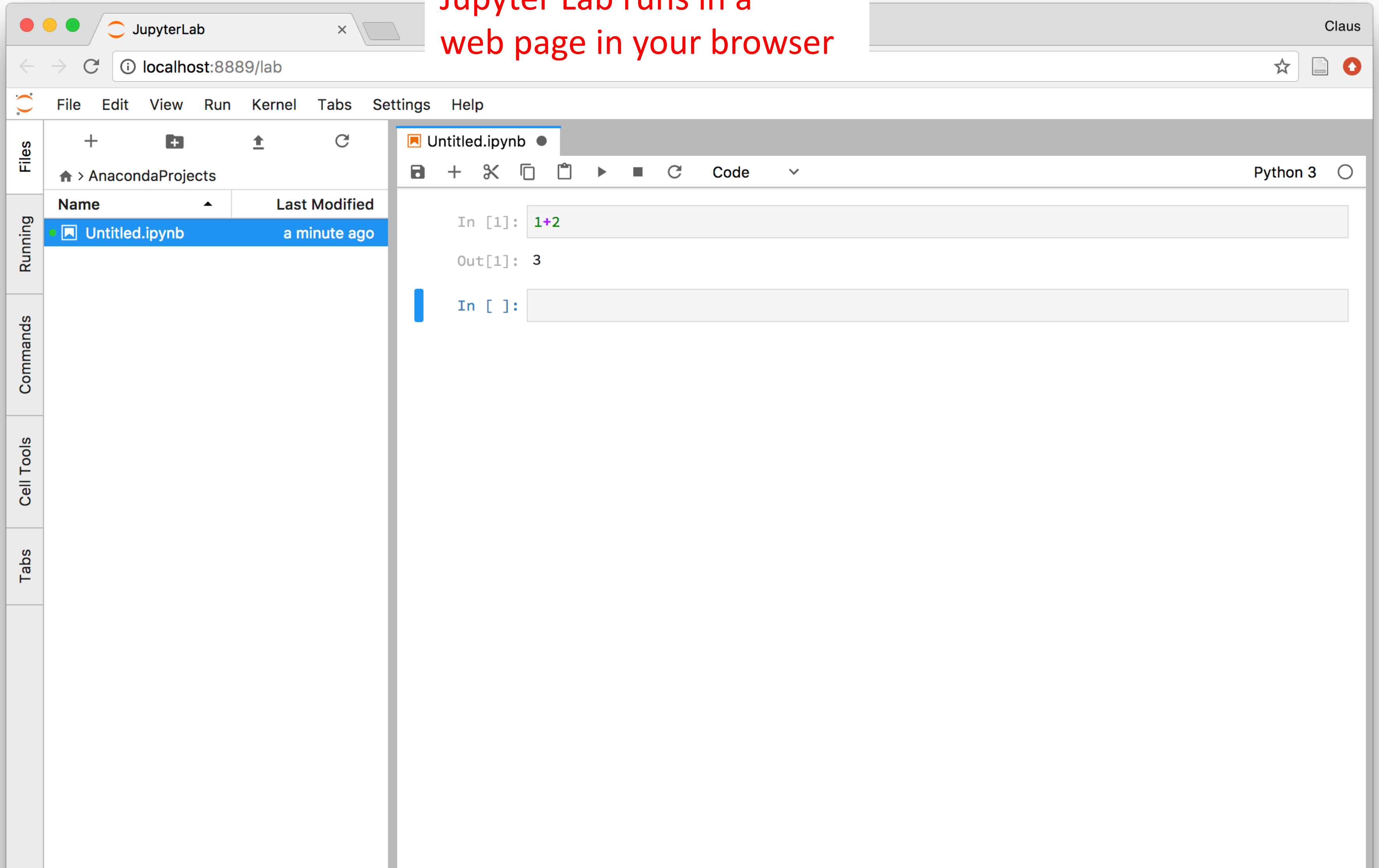
Multidimensional data visualization across files. Explore relationships within and among

orange3

3.4.1

Component based data mining framework. Data visualization and data analysis for

# Jupyter Lab



```
In [1]: 1+2

Out[1]: 3

In [ ]:
```

Jupyter Lab runs in a web page in your browser

# Counting like a computer scientist

```
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, ...
```

# Indexing in Python

| | | | | | |
|---|---|---|---|---|---|
| P | y | t | h | o | n |
| 0 | 1 | 2 | 3 | 4 | 5 |

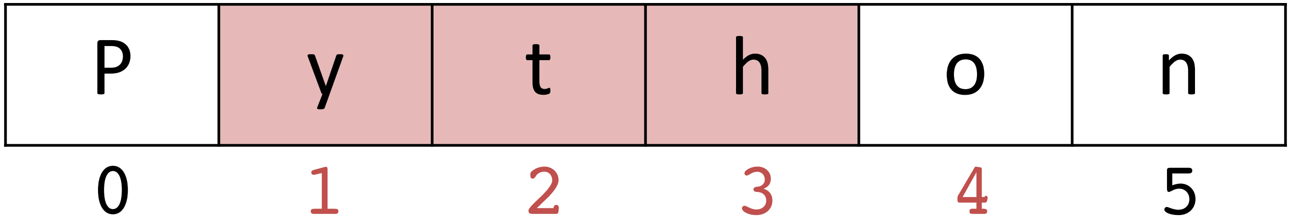
# Indexing in Python

| | | | | | |
|---|---|---|---|---|---|
| P | y | t | h | o | n |
| 0 | 1 | 2 | 3 | 4 | 5 |

```
In [1]: x="Python"

In [2]: x[0]
Out[2]: 'P'
```

# Indexing in Python



| P | y | t | h | o | n |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

```
In [1]: x="Python"

In [2]: x[1:4]
Out[2]: 'yth'
```

← We index from the first element to one past the last element

# Indexing in Python

| P | y | t | h | o | n |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

```
In [1]: x="Python"

In [2]: x[3:]
Out[2]: 'hon'
```

← Missing number means “to the end”

# We can also index in reverse

| | | | | | |
|---|---|---|---|---|---|
| P | y | t | h | o | n |
| −6 | −5 | −4 | −3 | −2 | −1 |

# We can also index in reverse

| | | | | | |
|---|---|---|---|---|---|
| P | y | t | h | o | n |
| -6 | -5 | -4 | -3 | -2 | -1 |

```
In [1]: x="Python"

In [2]: x[-6]
Out[2]: 'P'
```

# We can also index in reverse

| | | | | | |
|---|---|---|---|---|---|
| P | y | t | h | o | n |
| −6 | −5 | −4 | −3 | −2 | −1 |

```
In [1]: x="Python"

In [2]: x[-5:-2]
Out[2]: 'yth'
```

Again, we index one past the last element

# We can also index in reverse

| P | y | t | h | o | n |
|---|---|---|---|---|---|
| -6 | -5 | -4 | -3 | -2 | -1 |

```
In [1]: x="Python"

In [2]: x[-3:]
Out[2]: 'hon'
```

← This captures the last 3 characters