Python for Accounting Applications

NTPU
國立臺北大學
National Taipei University

Control Logic and Loops

1131PAA04
ACC2, NTPU (U2004) (Fall 2024)
Wed 6, 7, 8, (14:10-17:00) (9:10-12:00) (B3F10)

aws educate | Cloud Ambassador
2020 Cohort
aws academy Accredited Educator
aws certified Cloud Practitioner
aws certified Solutions Architect Associate

Min-Yuh Day, Ph.D,
Professor

Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday

2024-10-09

Syllabus

Week	Date	Subject/Topics
1	2024/09/11	Introduction to Python for Accounting Applications
2	2024/09/18	Python Programming and Data Science
3	2024/09/25	Foundations of Python Programming
4	2024/10/02	Data Structures
5	2024/10/09	Control Logic and Loops
6	2024/10/16	Functions and Modules; Files and Exception Handling
7	2024/10/23	Data Analytics and Visualization with Python
8	2024/10/30	Midterm Project Report

Syllabus

Week	Date	Subject/Topics
9	2024/11/06	Self-Learning
10	2024/11/13	Obtaining Data From the Web with Python
11	2024/11/20	Statistical Analysis with Python
12	2024/11/27	Machine Learning with Python
13	2024/12/04	Text Analytics with Generative AI and Python
14	2024/12/11	Applications of Accounting Data Analytics with Python
15	2024/12/18	Applications of ESG Data Analytics with Python
16	2024/12/25	Final Project Report

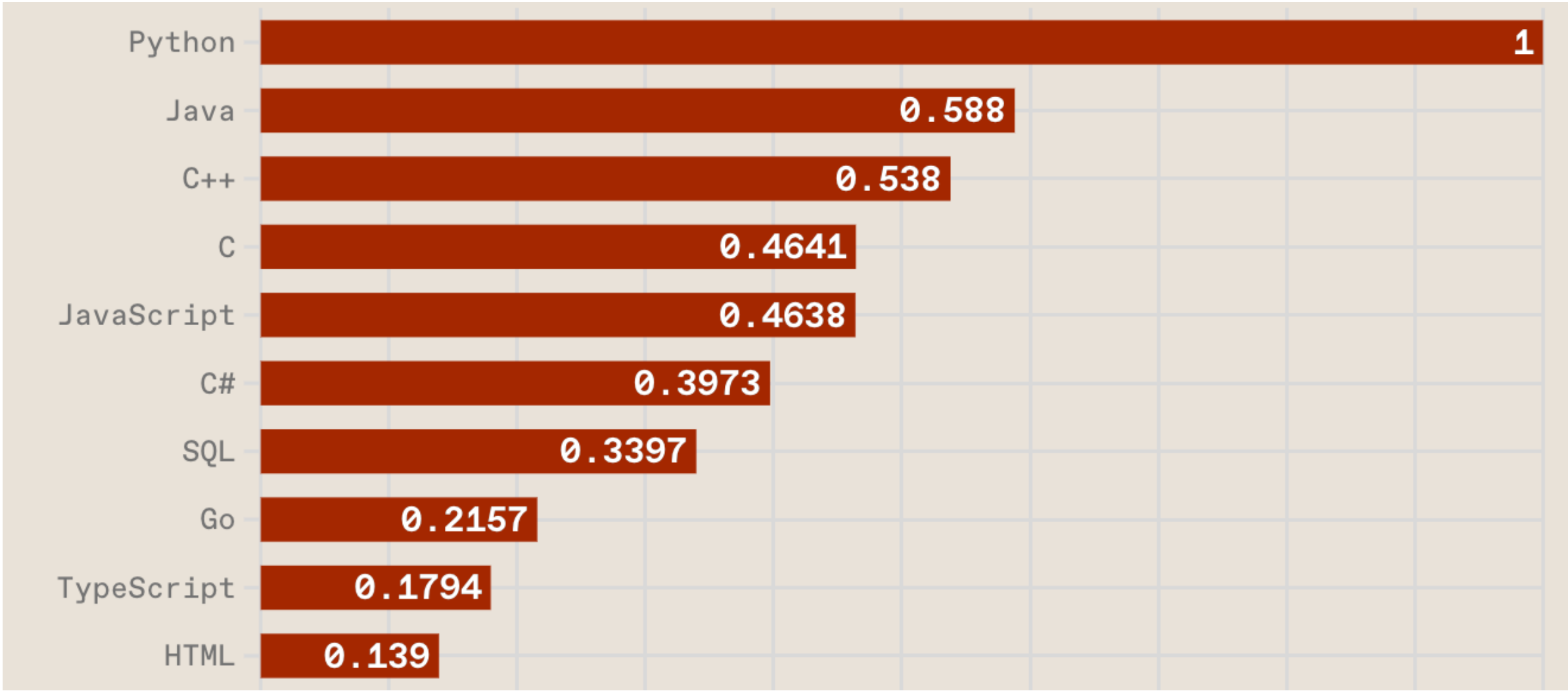
Outline

- Python if else
  - if elif else
  - Booleans: True, False
  - Operators: ==, !=, >, <, >=, <=, and, or, not
- Python for Loops
  - for
- Python while Loops
  - While
    - break
    - continue

Python

Programming

Top Programming Languages



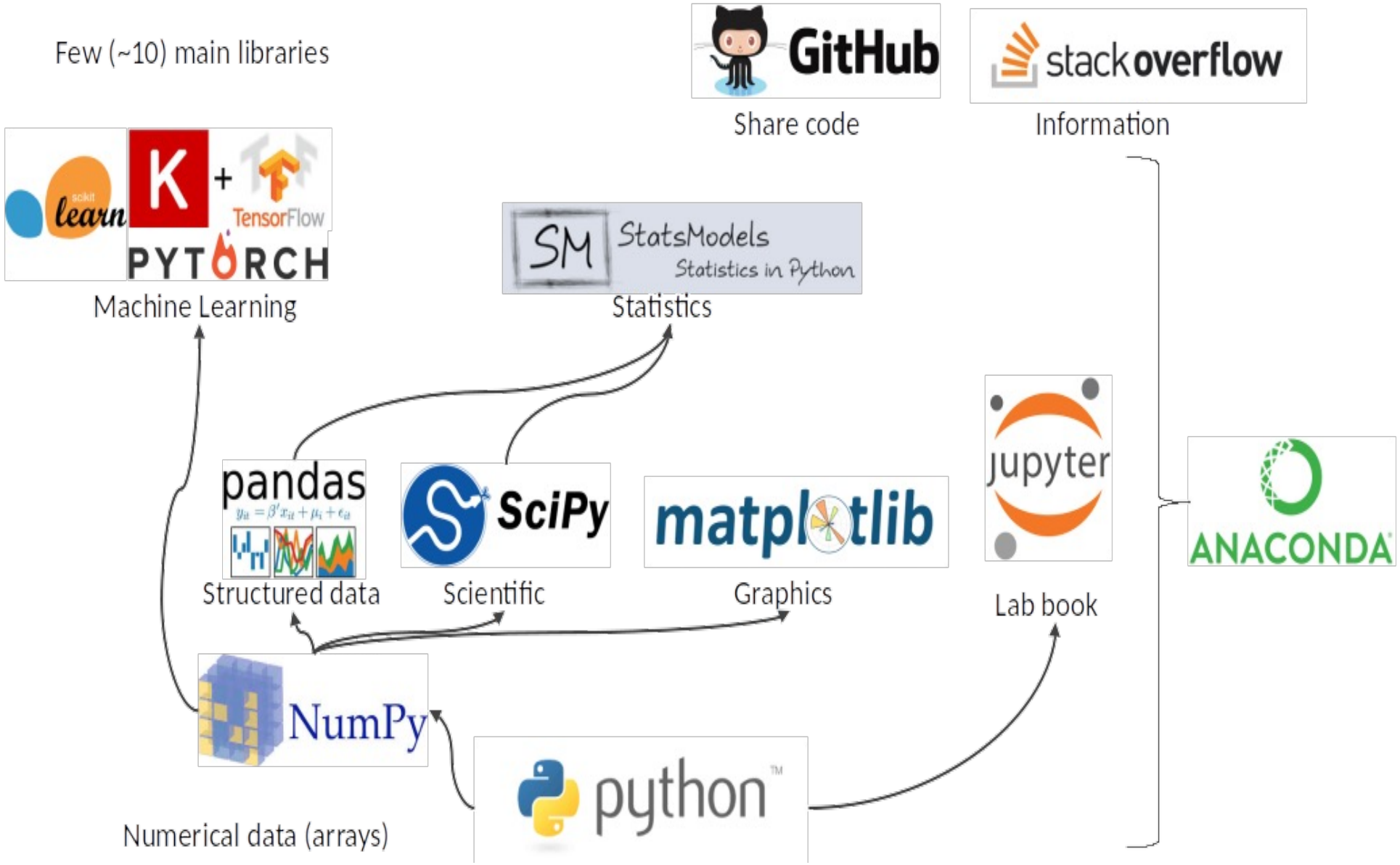
https://spectrum.ieee.org/the-top-programming-languages-2023

python™

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

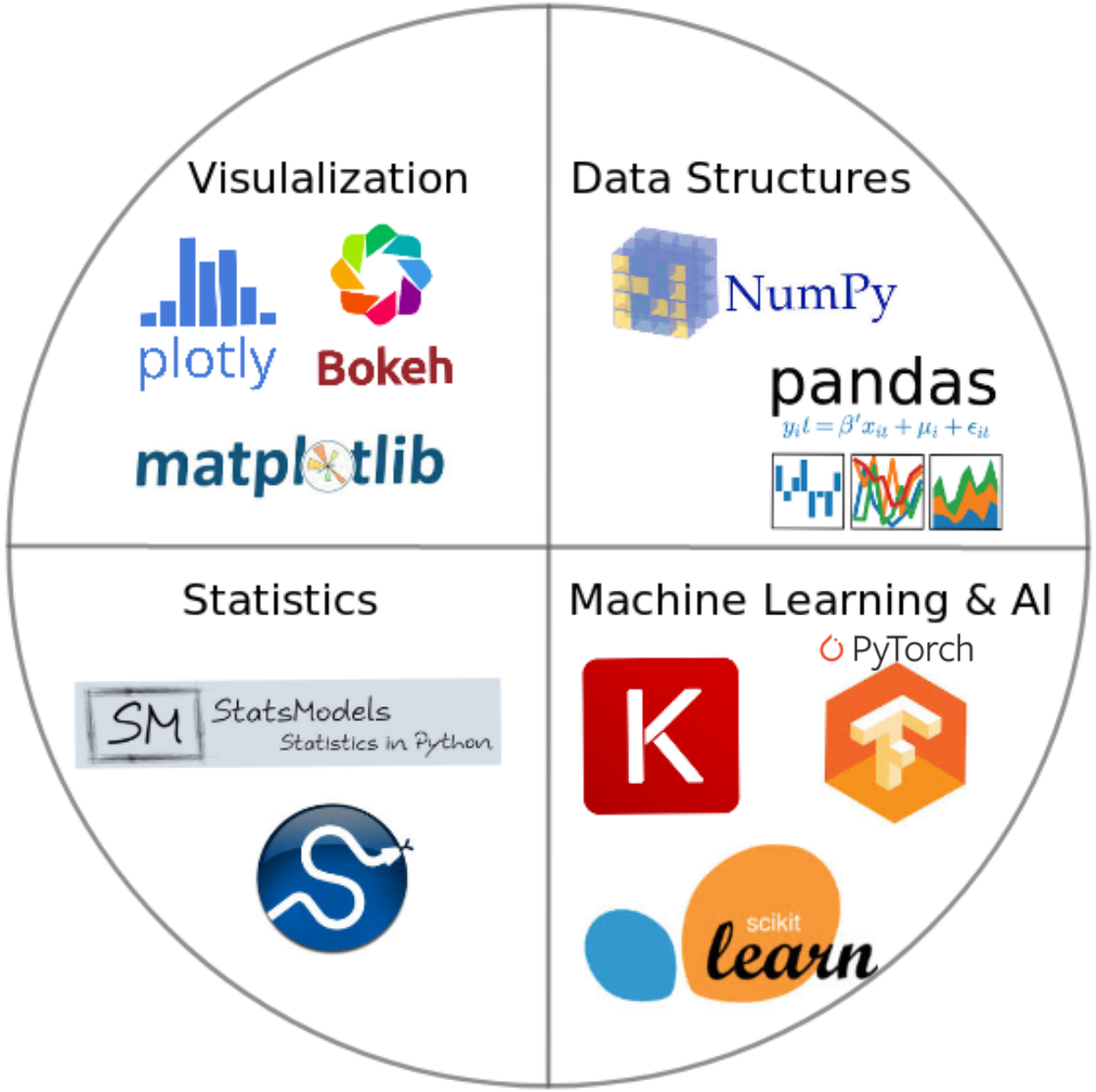
Source: https://www.python.org/doc/essays/blurb/

Python Ecosystem for Data Science



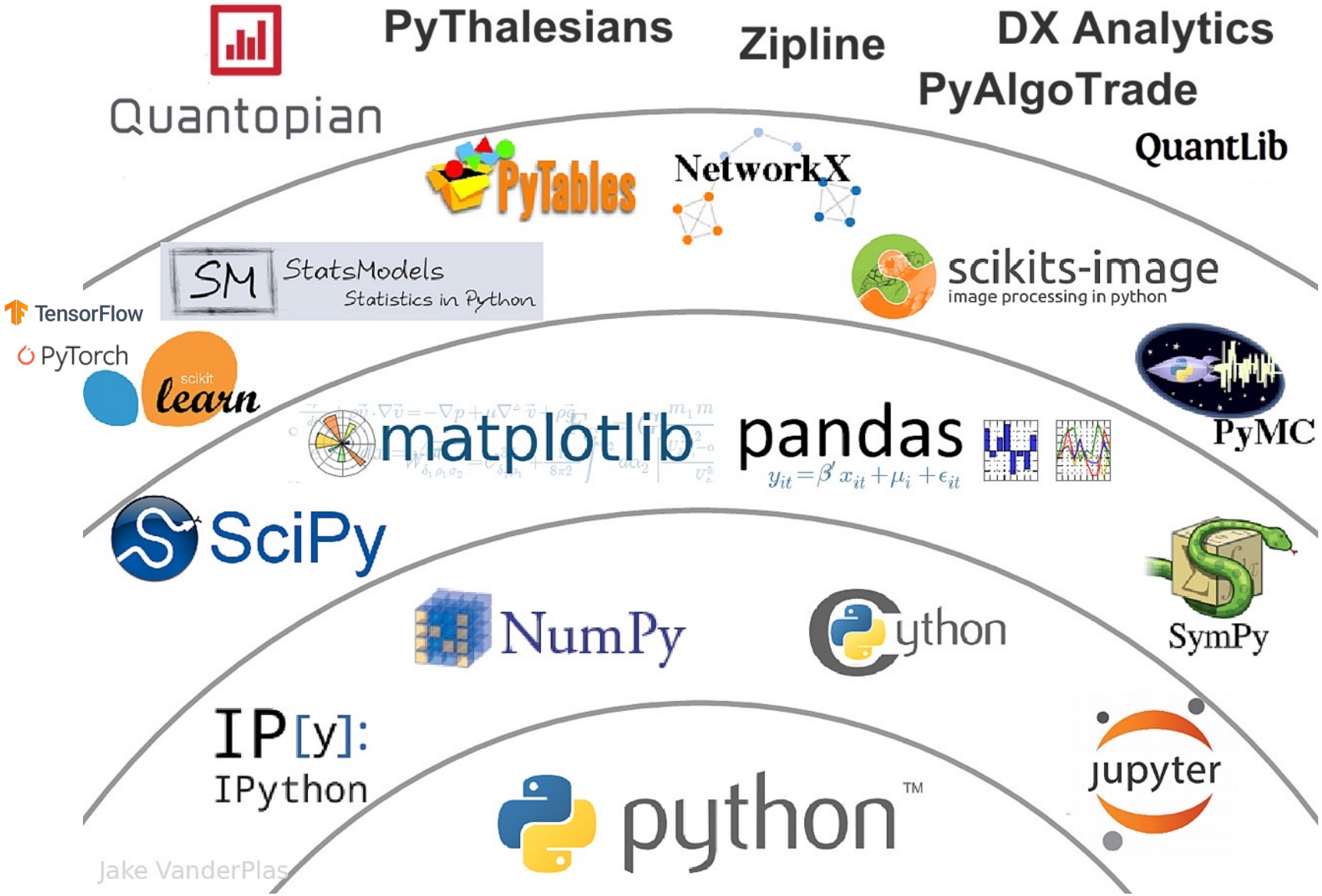
Source: https://medium.com/pyfinance/why-python-is-best-choice-for-financial-data-modeling-in-2019-c0d0d1858c45

Python Ecosystem for Data Science



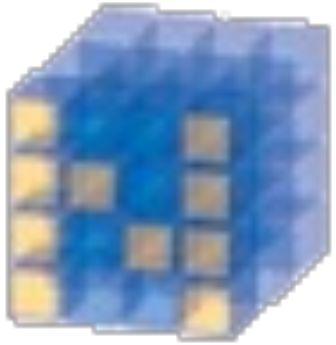
Source:https://duchesnay.github.io/pystatsml/introduction/python_ecosystem.html

The Quant Finance PyData Stack



Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#/5

Numpy



NumPy

Base

N-dimensional array

package

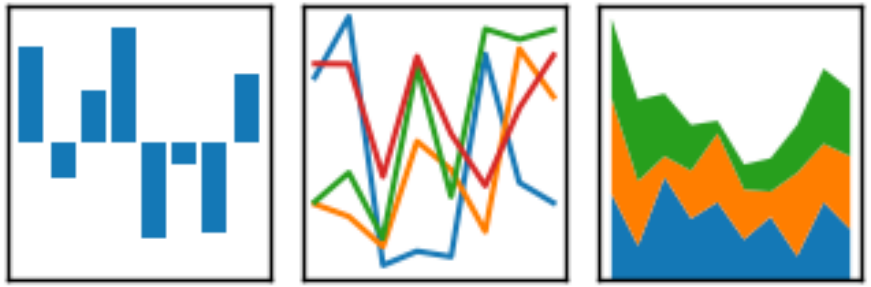
Python matplotlib

Source: https://matplotlib.org/

Python Pandas

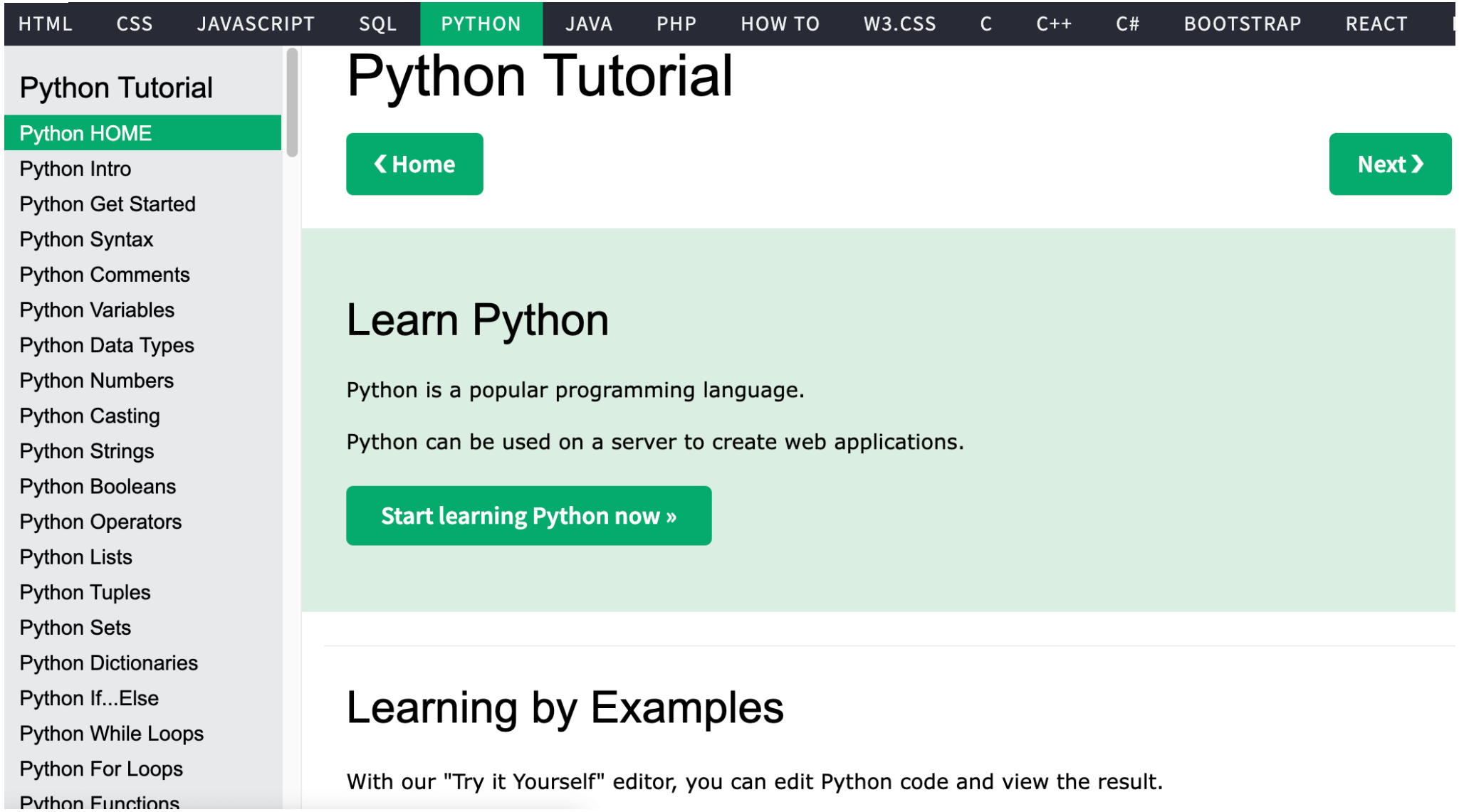
pandas

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



http://pandas.pydata.org/

W3Schools Python

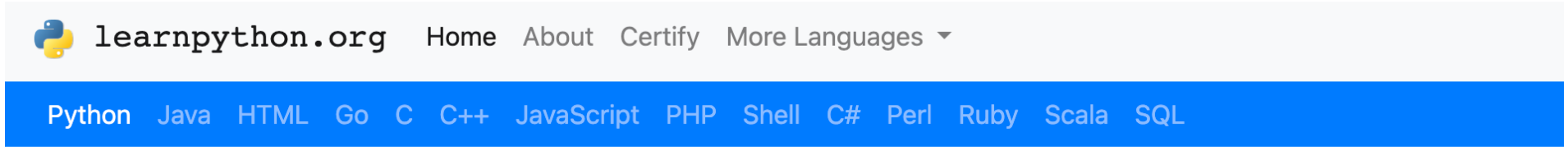


https://www.w3schools.com/python/

W3Schools Python: Try Python

https://www.w3schools.com/python/trypython.asp?filename=demo_default

LearnPython.org



Get started learning Python with DataCamp's free Intro to Python tutorial. Learn Data Science by completing interactive coding challenges and watching videos by expert instructors. Start Now!

Ready to take the test? Head onto LearnX and get your Python Certification!

This site is generously supported by DataCamp. DataCamp offers online interactive Python Tutorials for Data Science. Join **11 millions** other learners and get started learning Python for data science today!

Good news! You can save 25% off your Datacamp annual subscription with the code LEARNPYTHON23ALE25 - Click here to redeem your discount!

Welcome

Welcome to the LearnPython.org interactive Python tutorial.

Whether you are an experienced programmer or not, this website is intended for everyone who wishes to learn the Python programming language.

You are welcome to join our group on Facebook for questions, discussions and updates.

After you complete the tutorials, you can get certified at LearnX and add your certification to your LinkedIn profile.

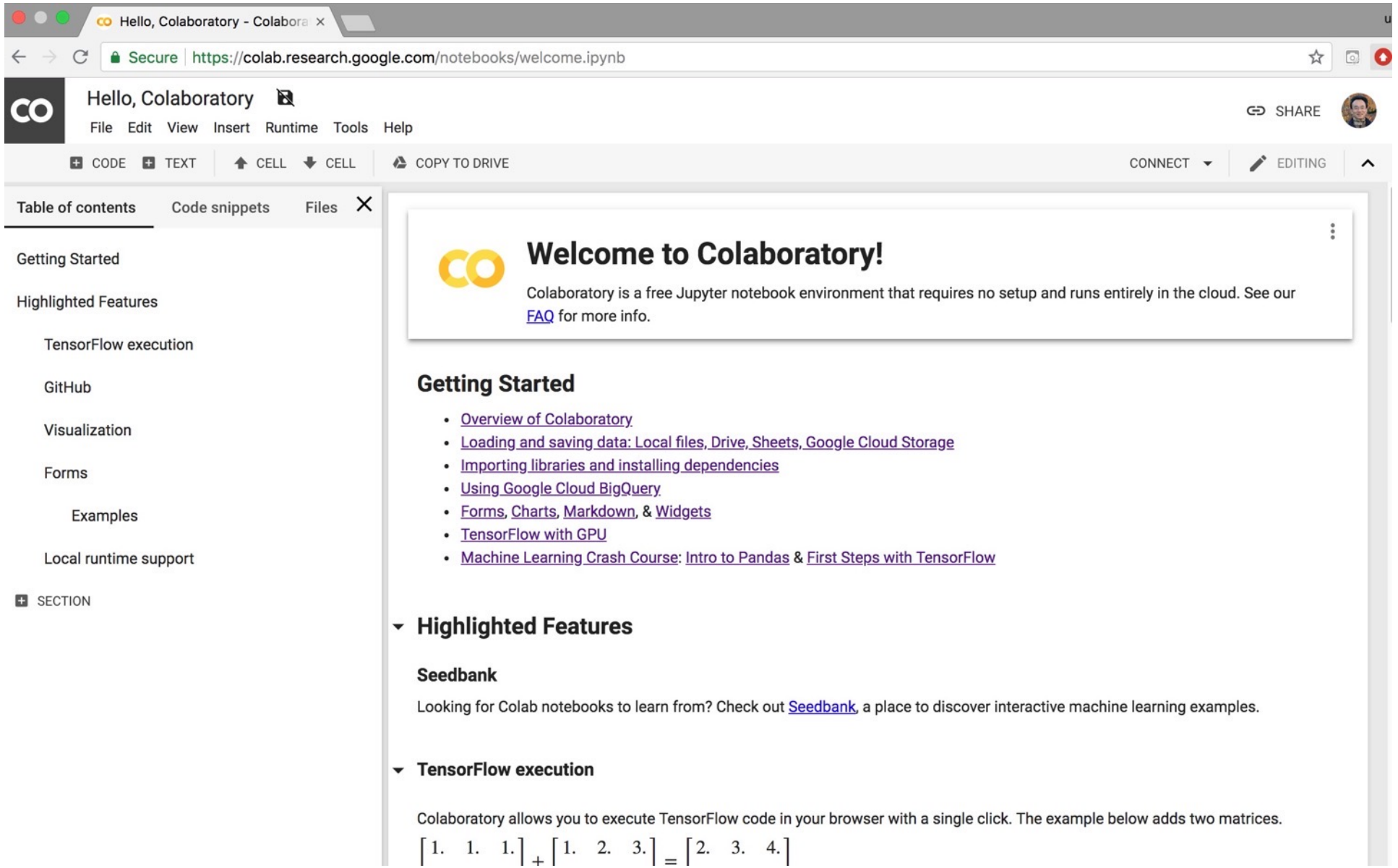
Just click on the chapter you wish to begin from, and follow the instructions. Good luck!

https://www.learnpython.org/

Google's Python Class

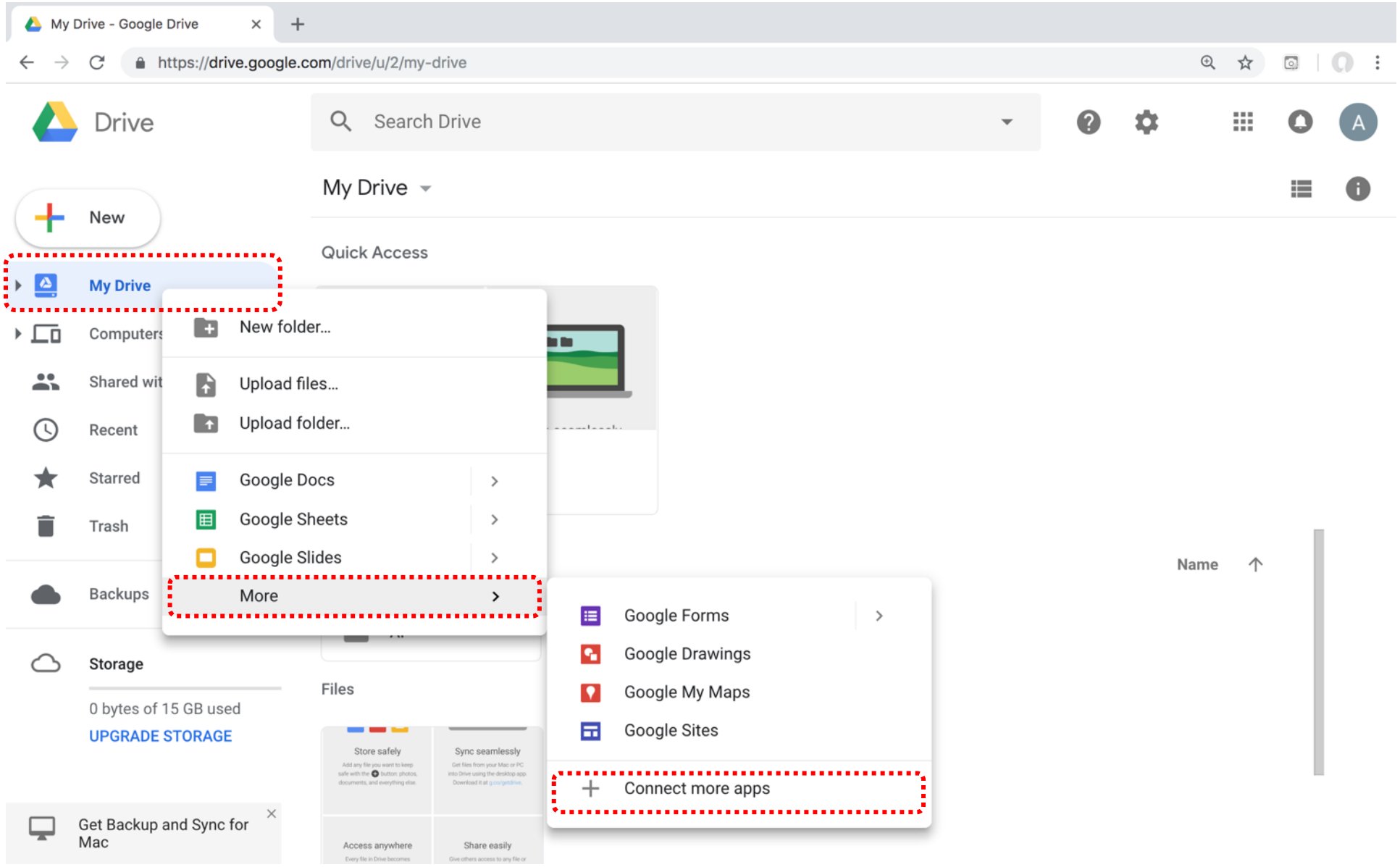
https://developers.google.com/edu/python

Google Colab

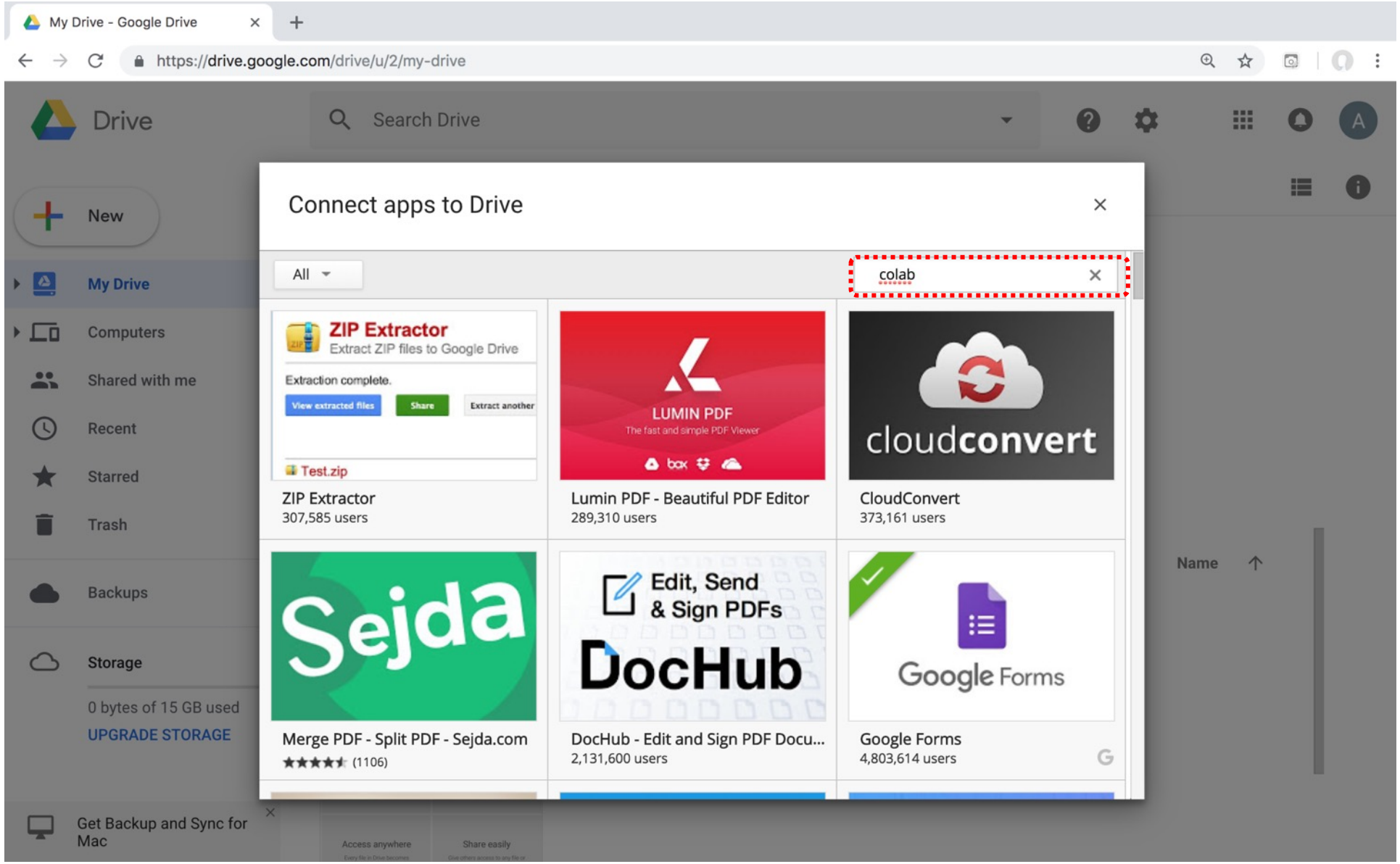


https://colab.research.google.com/notebooks/welcome.ipynb

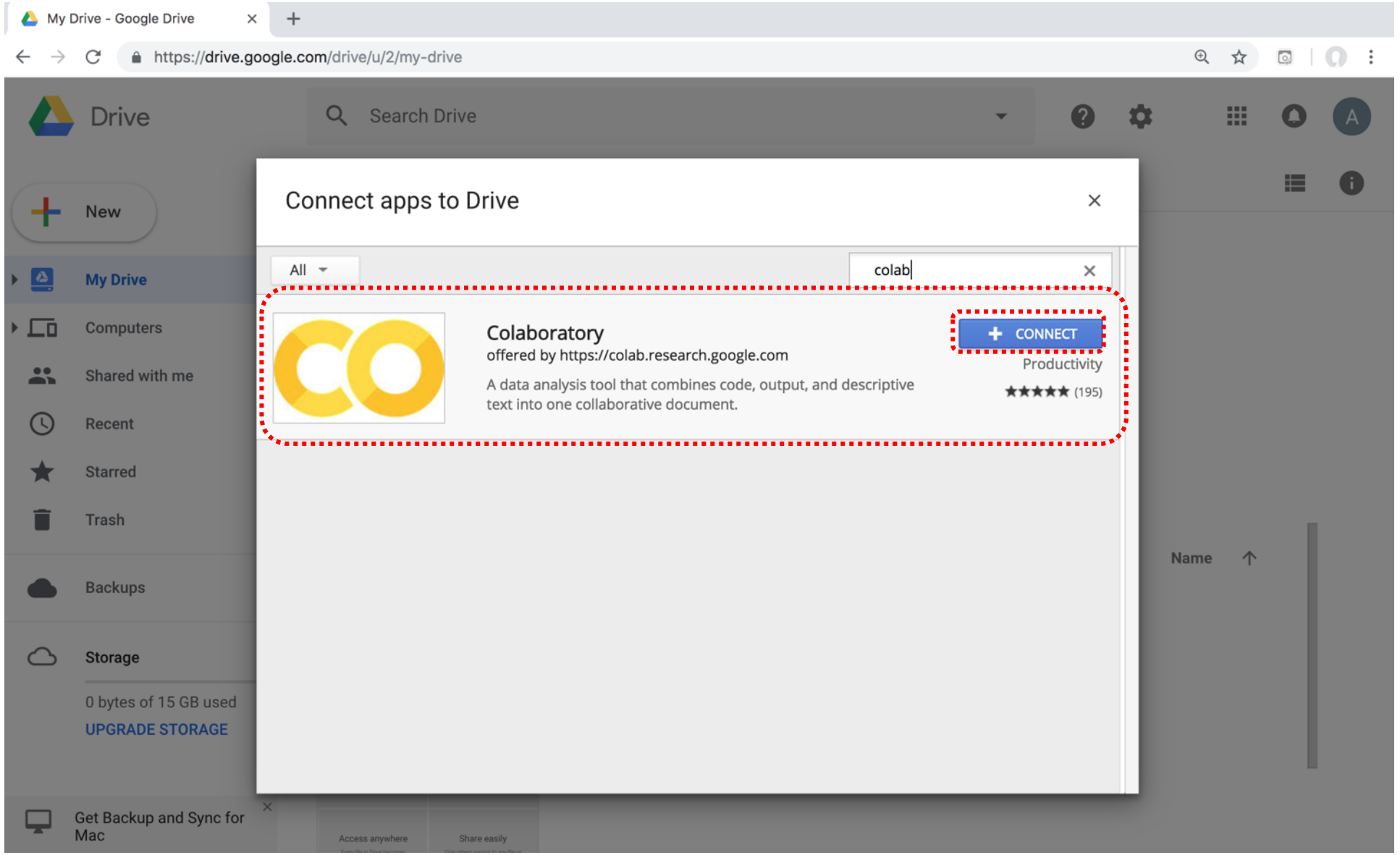
Connect Google Colab in Google Drive



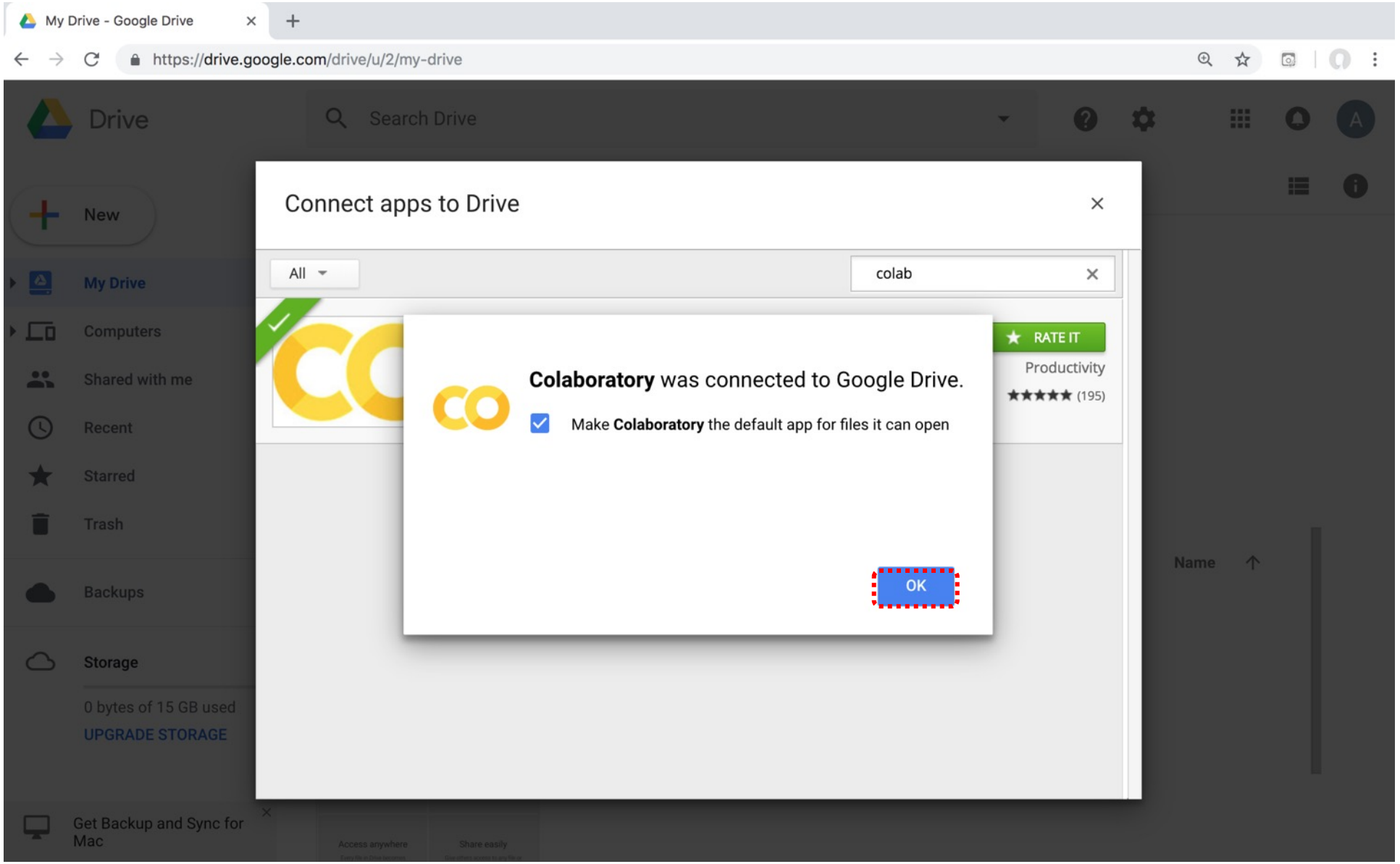
Google Colab



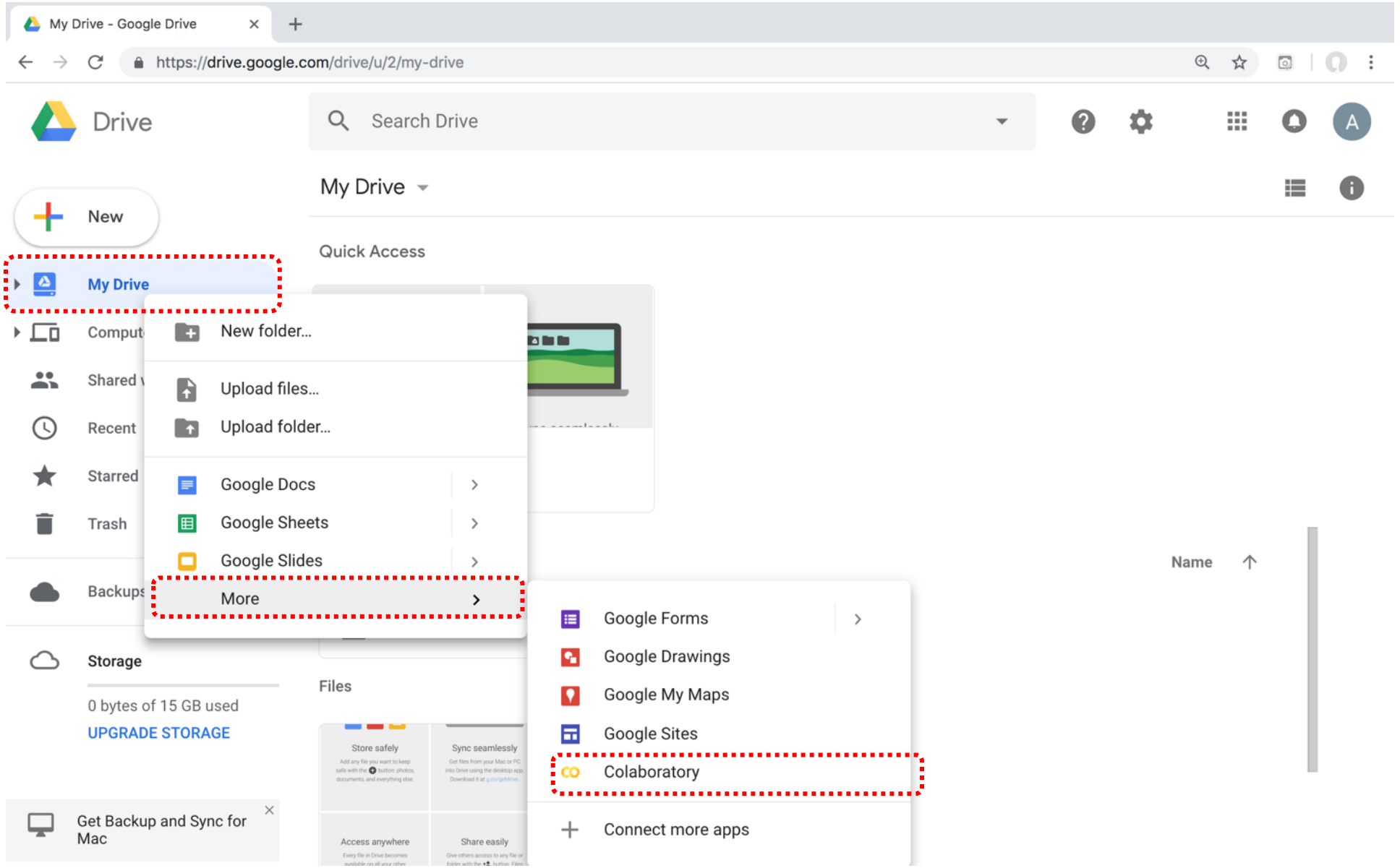
Google Colab



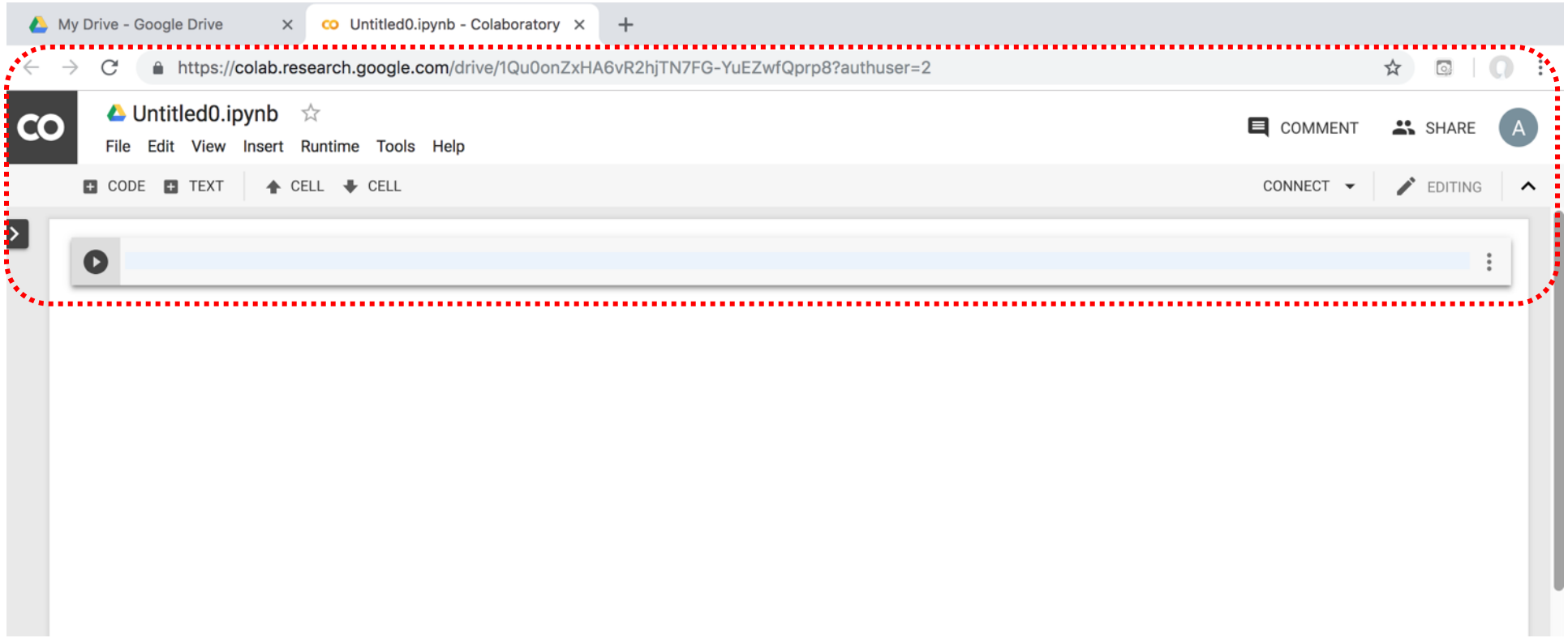
Connect Colaboratory to Google Drive



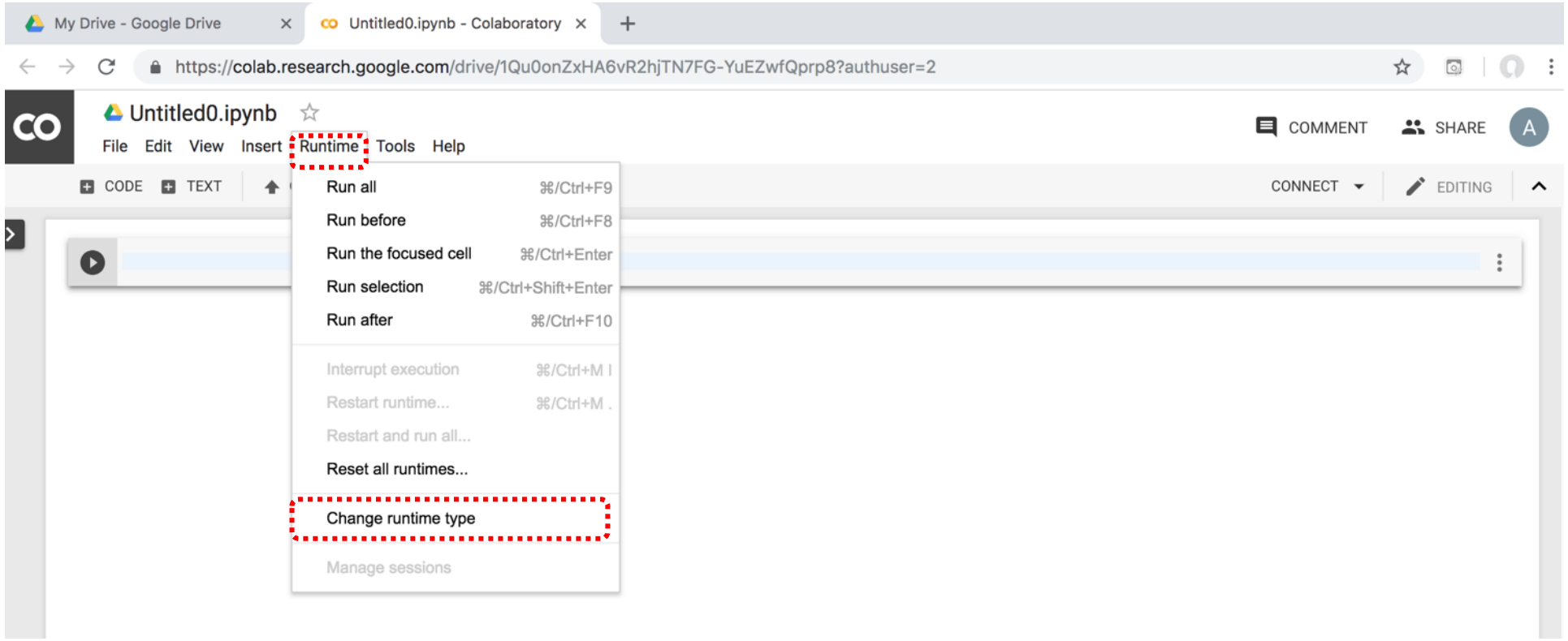
Google Colab



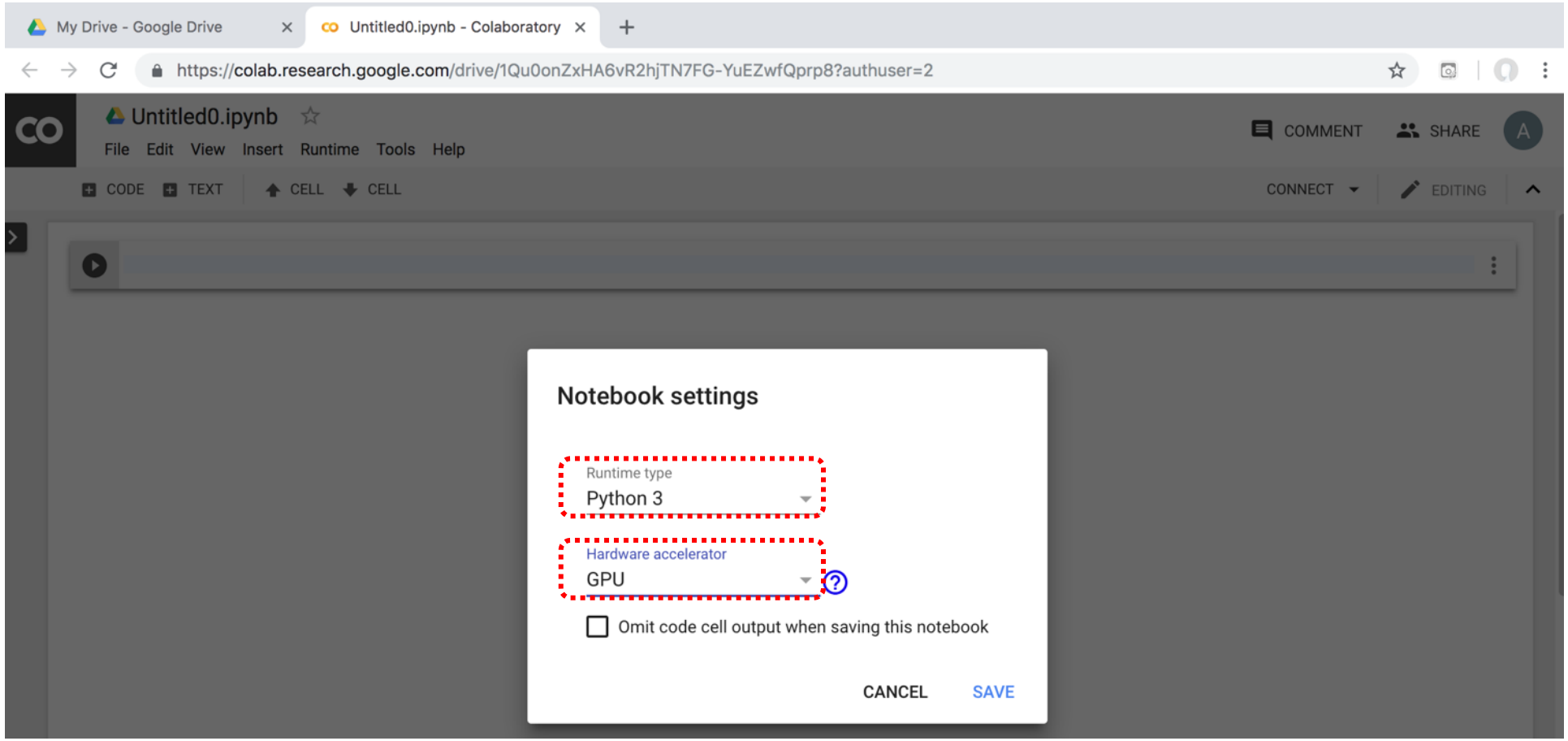
Google Colab



Google Colab



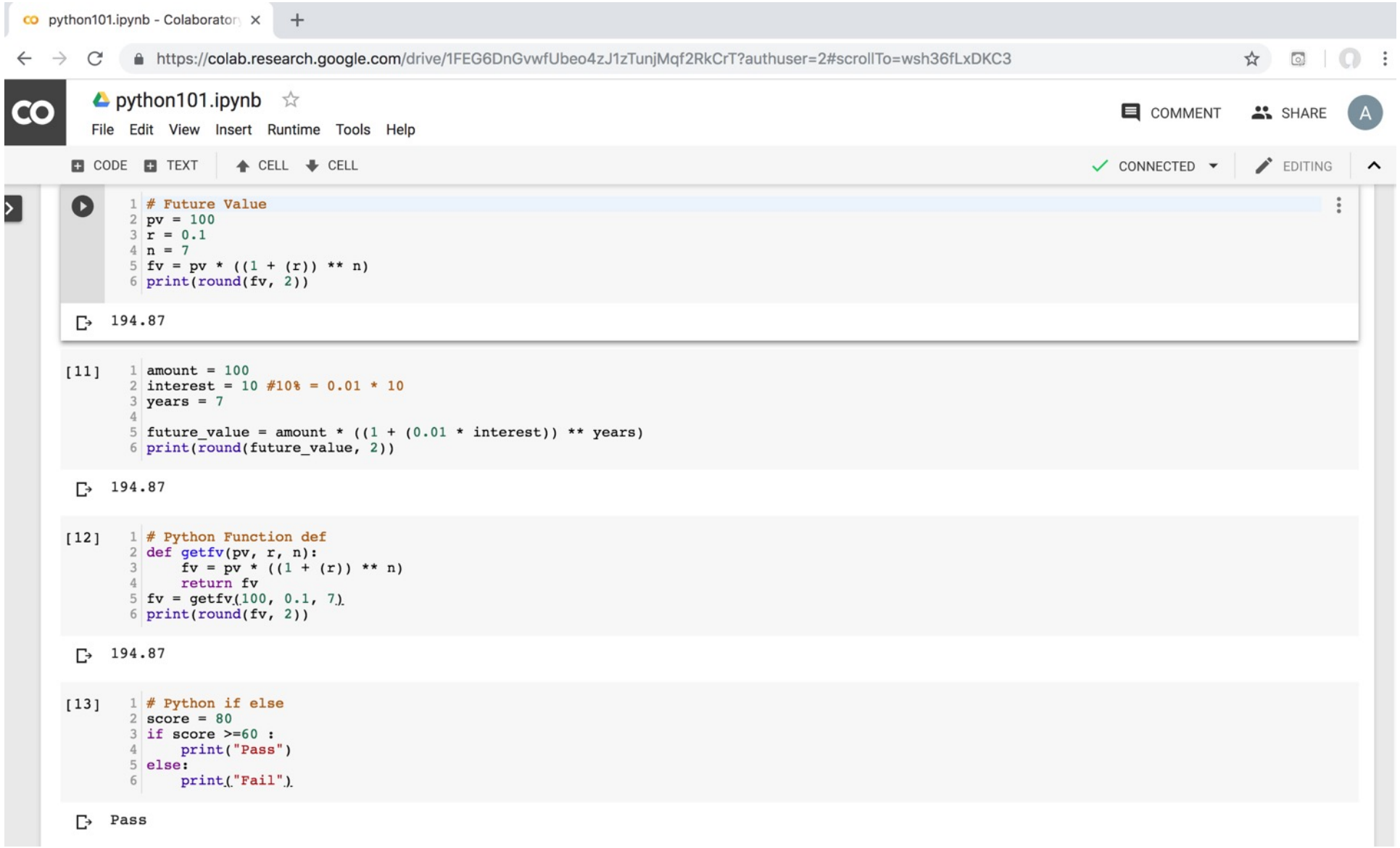
Run Jupyter Notebook
Python3 GPU
Google Colab



Google Colab Python Hello World
print('Hello World')

Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

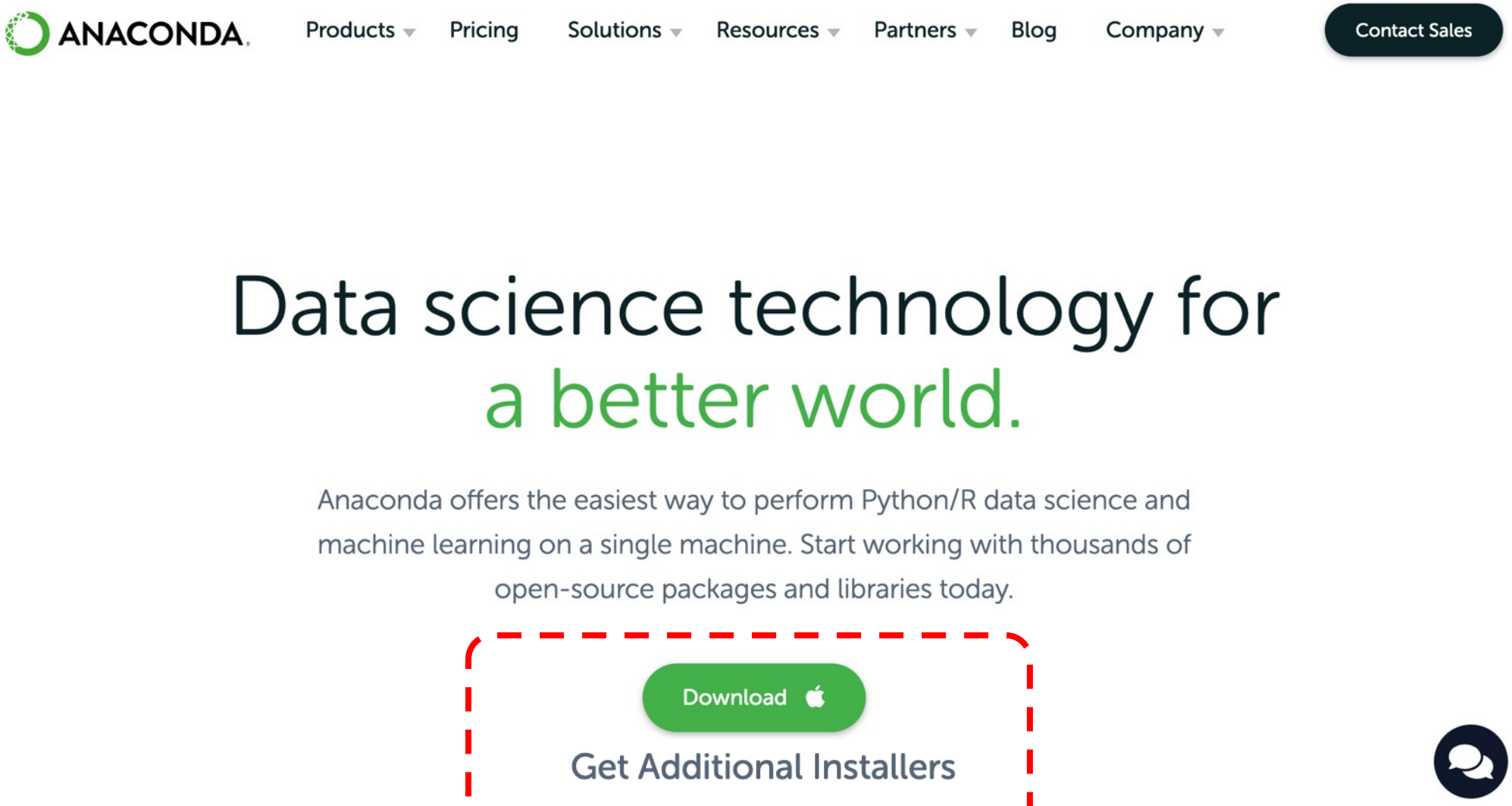


https://tinyurl.com/aintpupython101

Anaconda
The Most Popular
Python
Data Science Platform

Source: https://www.anaconda.com/

Download Anaconda

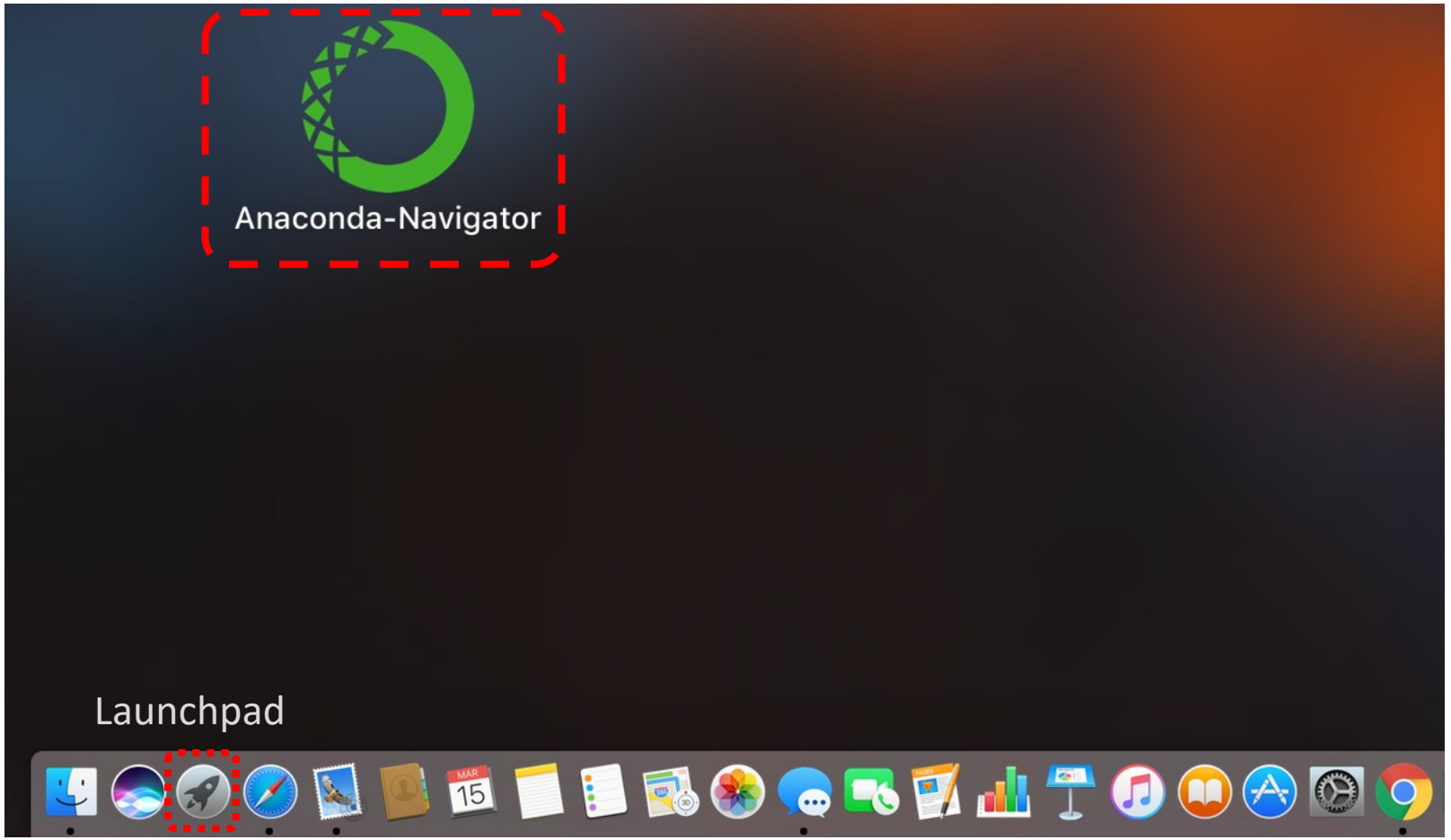


https://www.anaconda.com/download

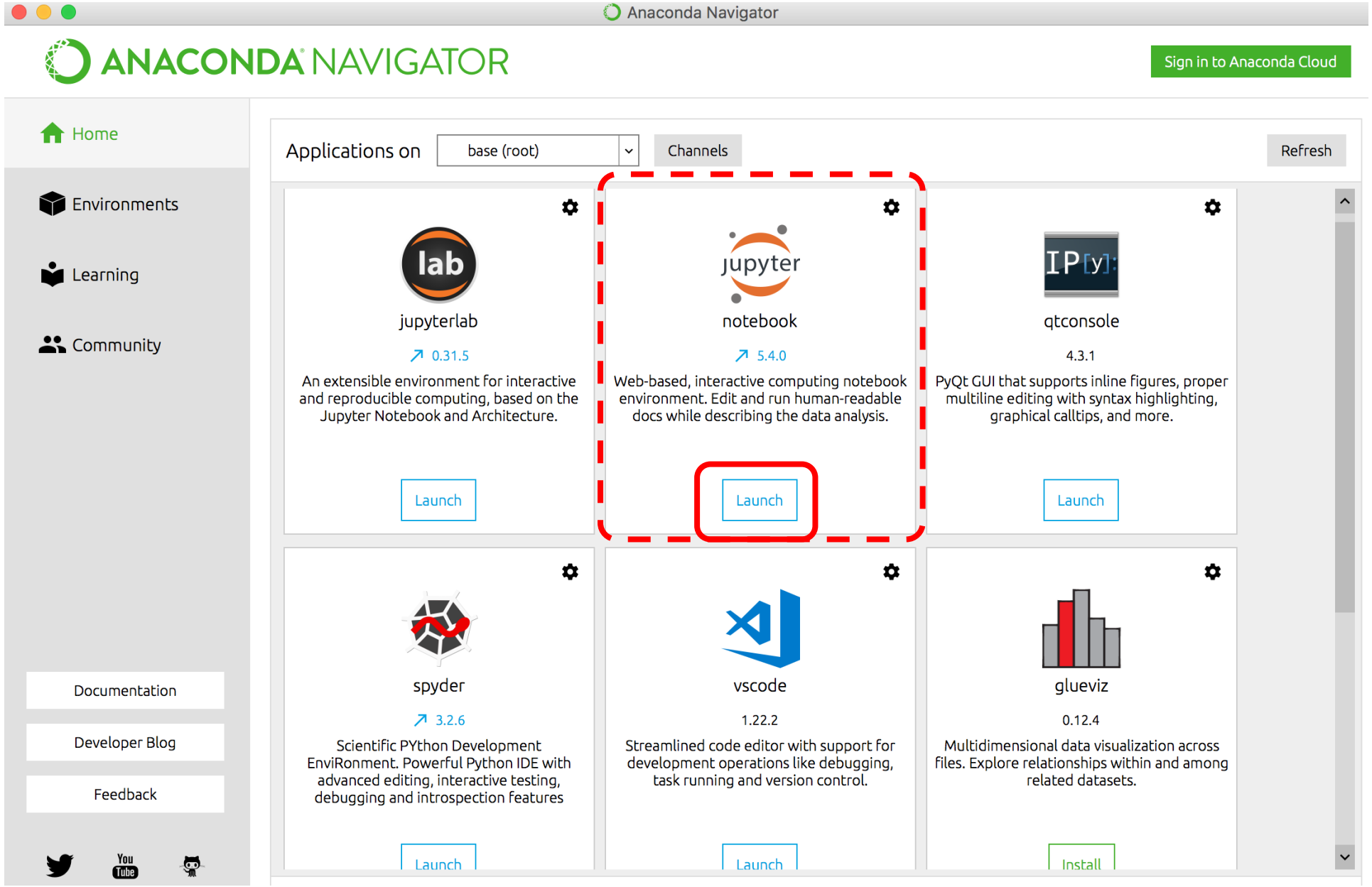
Python

HelloWorld

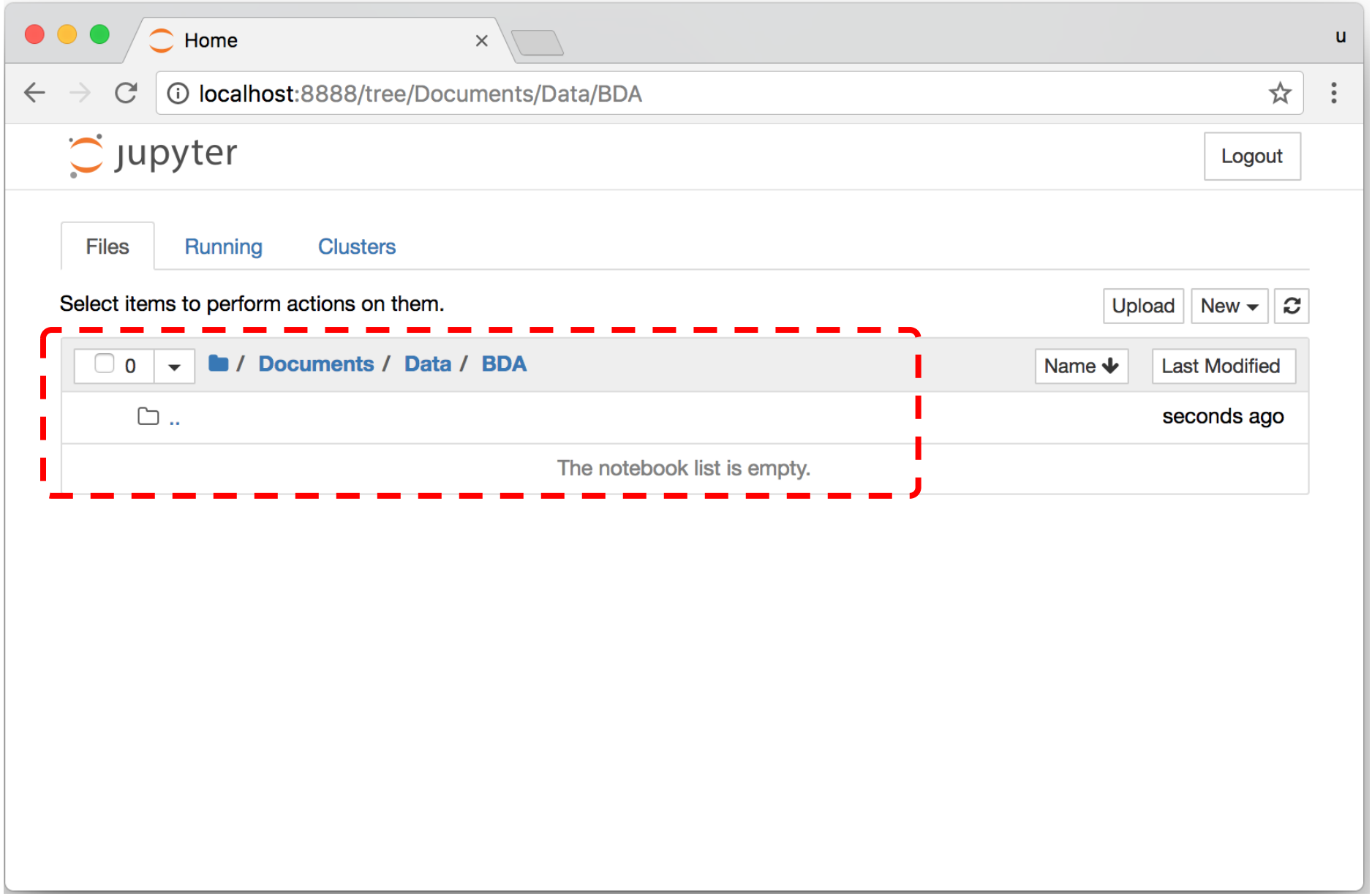
Anaconda-Navigator



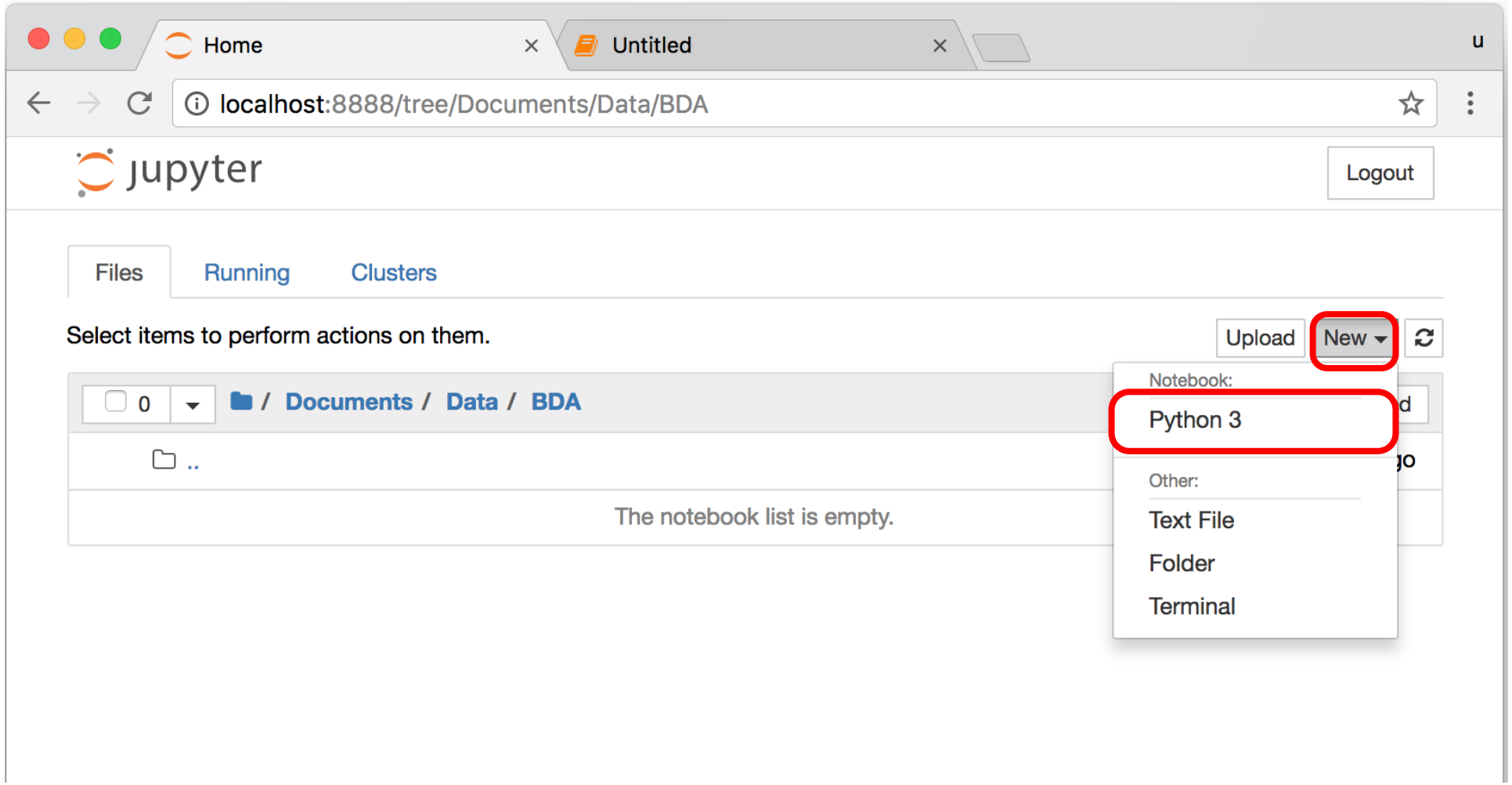
Anaconda Navigator



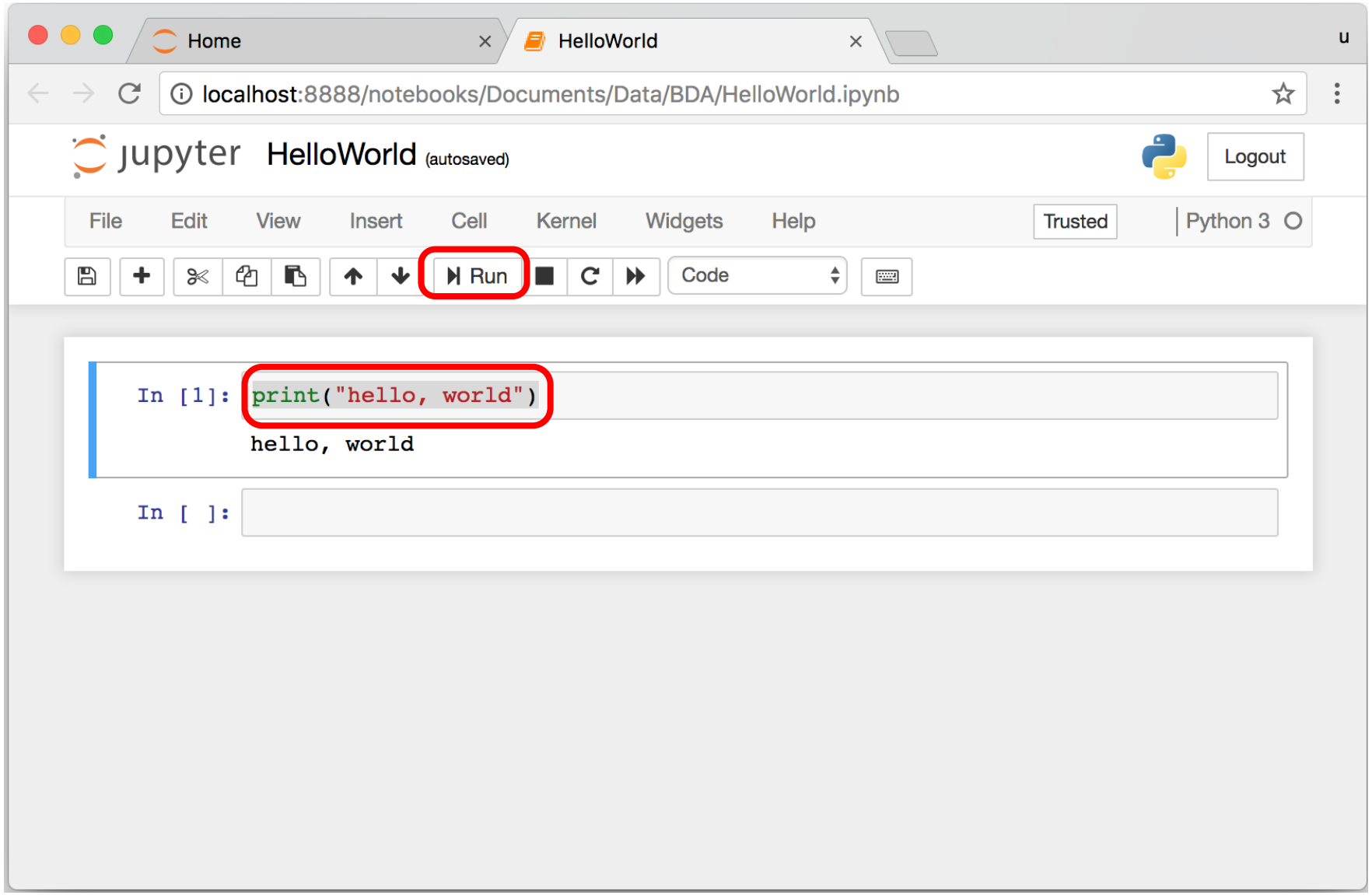
Jupyter Notebook



Jupyter Notebook
New Python 3



print("hello, world")



Source: https://www.python.org/community/logos/

Python

Programming

Foundations of Python Programming

- Python Syntax
  - Python Comments
- Python Variables
- Python Data Types
  - Python Numbers
  - Python Casting
  - Python Strings
- Python Operators
- Python Booleans

Python Hello World
print("Hello World")

```
print("Hello World")
```

Python Syntax

`# comment`

```
# comment
```

Python Syntax

Indentation

the spaces at the beginning of a code line

4 spaces

```
score = 80
if score >=60 :
    print("Pass")
```

Python Variables

```
# Python Variables
x = 2
price = 2.5
word = 'Hello'


word = 'Hello'
word = "Hello"
word = '''Hello'''
```

Python Variables

```
x = 2
y = x + 1
```

python_version()

```
# comment
from platform import python_version
print("Python Version:", python_version())
```

```
Python Version: 3.10.12
```

Python Data Types

```
x = "Hello World"   #str
x = 2               #int
x = 2.5             #float
x = 7j              #complex
```

Python Data Types

```
x = ["apple", "banana", "cherry"] #list
x = ("apple", "banana", "cherry") #tuple
x = range(6) #range
x = {"name" : "Tom", "age" : 20} #dict
x = {"apple", "banana", "cherry"} #set
x = frozenset({"apple", "banana", "cherry"})
#frozenset
```

Python Data Types

```
x = True #bool
x = b"Hello" #bytes
x = bytearray(5) #bytearray
x = memoryview(bytes(5)) #memoryview
x = None #NoneType
```

Python Casting

```
x = str(3) # x will be '3'
y = int(3) # y will be 3
z = float(3) # z will be 3.0
print(x, type(x))
print(y, type(y))
print(z, type(z))
```

```
3 <class 'str'>
3 <class 'int'>
3.0 <class 'float'>
```

Python Numbers

```
x = 2 # int
y = 3.4 # float
z = 7j #complex
print(x, type(x))
print(y, type(y))
print(z, type(z))
```

```
2 <class 'int’>
3.4 <class 'float’>
7j <class 'complex'>
```

Python Arithmetic Operators

```
Operator Name Example
+ Addition 7 + 2 = 9
- Subtraction 7 - 2 = 5
* Multiplication 7 * 2 = 14
/ Division 7 / 2 = 3.5
// Floor division 7 // 2 = 3   (Quotient)
% Modulus 7 % 2 = 1                (Remainder)
** Exponentiation 7 ** 2 = 49
```

Python Basic Operators

```
print('7 + 2 =', 7 + 2)
print('7 - 2 =', 7 - 2)
print('7 * 2 =', 7 * 2)
print('7 / 2 =', 7 / 2)
print('7 // 2 =', 7 // 2)
print('7 % 2 =', 7 % 2)
print('7 ** 2 =', 7 ** 2)
```

```
7 + 2 = 9
7 - 2 = 5
7 * 2 = 14
7 / 2 = 3.5
7 // 2 = 3
7 % 2 = 1
7 ** 2 = 49
```

Python Booleans: True or False

```
# Python Booleans: True or False
print(3 > 2)
print(3 == 2)
print(3 < 2)
```

Python BMI Calculator

```
# BMI Calculator in Python
height_cm = 170
weight_kg = 60
height_m = height_cm/100
BMI = (weight_kg/(height_m**2))

print("Your BMI is: " + str(round(BMI,1)))
```

```
Your BMI is: 20.8
```

Future value of a specified principal amount, rate of interest, and a number of years

Source: https://www.w3resource.com/python-exercises/python-basic-exercise-39.php

How much is your $100 worth after 7 years?

```
# How much is your $100 worth after 7 years?
fv = 100 * 1.1 ** 7
print('fv = ', round(fv, 2))
# output = 194.87
```

```
fv = 194.87
```

Future Value

```
# Future Value
pv = 100
r = 0.1
n = 7
fv = pv * ((1 + (r)) ** n)
print(round(fv, 2))
```

```
194.87
```

Future Value

```
# Future Value
amount = 100
interest = 10 #10% = 0.01 * 10
years = 7

future_value = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))
```

```
194.87
```

Python
Data Structures

Python Data Structures

- Python Lists []
- Python Tuples ()
- Python Sets {}
- Python Dictionaries {k:v}

Python Data Structures

```
fruits = ["apple", "banana", "cherry"] #lists []
colors = ("red", "green", "blue") #tuples ()
animals = {'cat', 'dog'} #sets {}
person = {"name" : "Tom", "age" : 20} #dictionaries {}
```

Python Data Types

```
x = ["apple", "banana", "cherry"] #list
x = ("apple", "banana", "cherry") #tuple
x = {"name" : "Tom", "age" : 20} #dict
x = {"apple", "banana", "cherry"} #set
```

Python Collections

- There are four collection data types in the Python programming language
- List []
  - a collection which is ordered and changeable. Allows duplicate members.
- Tuple ()
  - a collection which is ordered and unchangeable. Allows duplicate members.
- Set {}
  - a collection which is unordered, unchangeable, and unindexed. No duplicate members.
- Dictionary {k:v}
  - a collection which is ordered and changeable. No duplicate members.

Python Dictionaries {k:v}

- **As of Python version 3.7, dictionaries are ordered.**
- **In Python 3.6 and earlier, dictionaries are unordered.**

Lists []

```
x = [60, 70, 80, 90]
print(len(x))
print(x[0])
print(x[1])
print(x[-1])
```

```
4
60
70
90
```

Lists []

- len(): how many items
- type(): data type
- list() constructor: creating a new list

Python List Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

https://www.w3schools.com/python/python_lists_methods.asp

Tuples ()

A tuple in Python is a collection that cannot be modified.
A tuple is defined using parenthesis.

```
x = (10, 20, 30, 40, 50)
print(x[0])
print(x[1])
print(x[2])
print(x[-1])
```

```
10
20
30
50
```

Source: http://pythonprogramminglanguage.com/tuples/

Sets {}

```
animals = {'cat', 'dog'}
print('cat' in animals)      True
print('fish' in animals)     False
animals.add('fish')
print('fish' in animals)     True
print(len(animals))          3
animals.add('cat')
print(len(animals))          3
animals.remove('cat')
print(len(animals))          2
```

Source: http://cs231n.github.io/python-numpy-tutorial/

Dictionary {key : value}

Python Dictionary

Key → Value

'EN' → 'English'

'FR' → 'French'

```
k = { 'EN':'English', 'FR':'French' }
print(k['EN'])
```

```
English
```

Source: http://pythonprogramminglanguage.com/dictionary/

Python Data Structures

```
fruits = ["apple", "banana", "cherry"] #lists []
colors = ("red", "green", "blue") #tuples ()
animals = {'cat', 'dog'} #sets {}
person = {"name" : "Tom", "age" : 20} #dictionaries {}
```

https://tinyurl.com/aintpupython101

Python Control Logic and Loops

Python Control Logic and Loops

- Python if else
  - if elif else
  - Booleans: True, False
  - Operators: ==, !=, >, <, >=, <=, and, or, not
- Python for Loops
  - for
- Python while Loops
  - While
    - break
    - continue

Python if...else

- Python if...else
  - if elif else
  - Booleans: True, False
  - Operators: ==, !=, >, <, >=, <=, and, or, not

Python Conditions and If statements

- Python supports the usual logical conditions from mathematics:
  - Equals: a == b
  - Not Equals: a != b
  - Less than: a < b
  - Less than or equal to: a <= b
  - Greater than: a > b
  - Greater than or equal to: a >= b

Python Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Source: https://www.w3schools.com/python/python_operators.asp

Python Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Source: https://www.w3schools.com/python/python_operators.asp

Python if

```
# Python if
score = 80
if score >=60 :
   print("Pass")
```

Python if else

```
# Python if else
score = 80
if score >=60 :
    print("Pass")
else:
    print("Fail")
```

Python if elif else

```
score = 95
if score >= 90 :
    print("A")
elif score >=60 :
    print("Pass")
else:
    print("Fail")
```

Python if elif else

```
# Python if elif else
score = 90
grade = ""
if score >=90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "E"
print(grade)
```

Python for Loops

```
for i in range(1,6):
    print(i)
```

```
1
2
3
4
5
```

Python for loops

```
# for loops
for i in range(1,10):
    for j in range(1,10):
    print(i, ' * ' , j , ' = ', i*j)
```

https://tinyurl.com/aintpupython101

Python while Loops

- while
  - break
  - continue

Python while loops

```
# while loops
age = 10
while age < 20:
    print(age)
    age = age + 1
```

Summary

- Python if else
  - if elif else
  - Booleans: True, False
  - Operators: ==, !=, >, <, >=, <=, and, or, not
- Python for Loops
  - for
- Python while Loops
  - while
    - break
    - continue

References

- Wes McKinney (2022), "Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter", 3rd Edition, O'Reilly Media.
- Aurélien Géron (2023), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 3rd Edition, O'Reilly Media.
- Steven D'Ascoli (2022), Artificial Intelligence and Deep Learning with Python: Every Line of Code Explained For Readers New to AI and New to Python, Independently published.
- Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson.
- Denis Rothman (2024), Transformers for Natural Language Processing and Computer Vision - Third Edition: Explore Generative AI and Large Language Models with Hugging Face, ChatGPT, GPT-4V, and DALL-E 3, 3rd ed. Edition, Packt Publishing
- Ben Auffarth (2023), Generative AI with LangChain: Build large language model (LLM) apps with Python, ChatGPT and other LLMs, Packt Publishing.
- Varun Grover, Roger HL Chiang, Ting-Peng Liang, and Dongsong Zhang (2018), "Creating Strategic Business Value from Big Data Analytics: A Research Framework", Journal of Management Information Systems, 35, no. 2, pp. 388-423.
- Junliang Wang, Chuqiao Xu, Jie Zhang, and Ray Zhong (2022). "Big data analytics for intelligent manufacturing systems: A review." Journal of Manufacturing Systems 62 (2022): 738-752.
- Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
- Python Programming, https://pythonprogramming.net/
- Python, https://www.python.org/
- Python Programming Language, http://pythonprogramminglanguage.com/
- Numpy, http://www.numpy.org/
- Pandas, http://pandas.pydata.org/
- Skikit-learn, http://scikit-learn.org/
- W3Schools Python, https://www.w3schools.com/python/
- Learn Python, https://www.learnpython.org/
- Google's Python Class, https://developers.google.com/edu/python
- Harvard University (2024), CS50x 2024 - Lecture 6 - Python, http://www.youtube.com/watch?v=EHi0RDZ31VA
- Min-Yuh Day (2024), Python 101, https://tinyurl.com/aintpupython101