

Python for Natural Language Processing

1121AITA03

MBA, IM, NTPU (M5265) (Fall 2023)

Tue 2, 3, 4 (9:10-12:00) (B3F17)



<https://meet.google.com/miy-fbif-max>



Min-Yuh Day, Ph.D,
Associate Professor

Institute of Information Management, National Taipei University

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



Syllabus

Week Date Subject/Topics

- 1 2023/09/13 Introduction to Artificial Intelligence for Text Analytics
- 2 2023/09/20 Foundations of Text Analytics:
Natural Language Processing (NLP)
- 3 2023/09/27 Python for Natural Language Processing
- 4 2023/10/04 Natural Language Processing with Transformers
- 5 2023/10/11 Case Study on Artificial Intelligence for Text Analytics I
- 6 2023/10/18 Text Classification and Sentiment Analysis

Syllabus

Week Date Subject/Topics

7 2023/10/25 Multilingual Named Entity Recognition (NER)

8 2023/11/01 Midterm Project Report

9 2023/11/08 Text Similarity and Clustering

10 2023/11/15 Text Summarization and Topic Models

11 2023/11/22 Text Generation with Large Language Models (LLMs)

12 2023/11/29 Case Study on Artificial Intelligence for Text Analytics II

Syllabus

Week Date Subject/Topics

13 2023/12/06 Question Answering and Dialogue Systems

14 2023/12/13 Deep Learning, Generative AI, Transfer Learning,
Zero-Shot, and Few-Shot Learning for Text Analytics

15 2023/12/20 Final Project Report I

16 2023/12/27 Final Project Report II

Python for Natural Language Processing

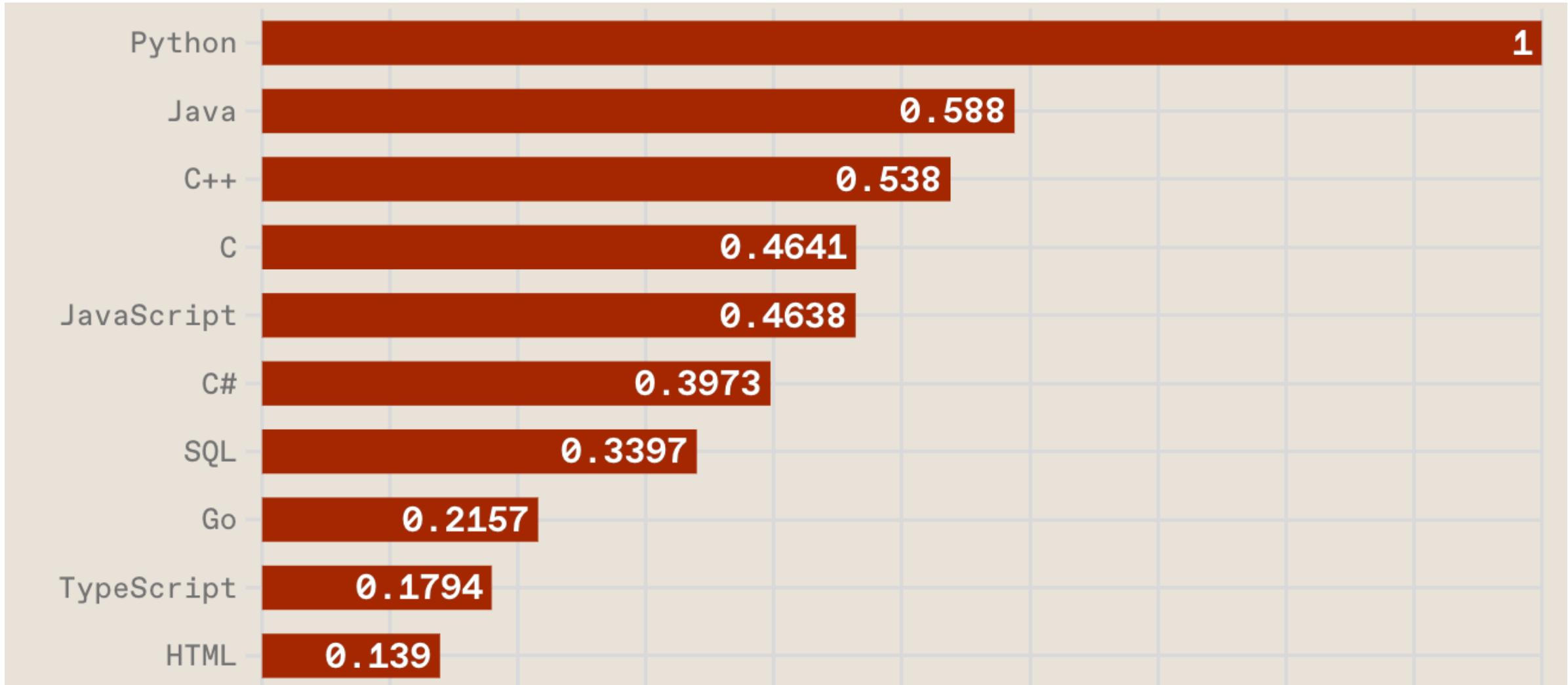
Outline

- Python for Natural Language Processing
 - Python Ecosystem for Data Science
 - Python
 - Programming language
 - Numpy
 - Scientific computing
 - SpaCy
 - Natural Language Processing



Python Programming

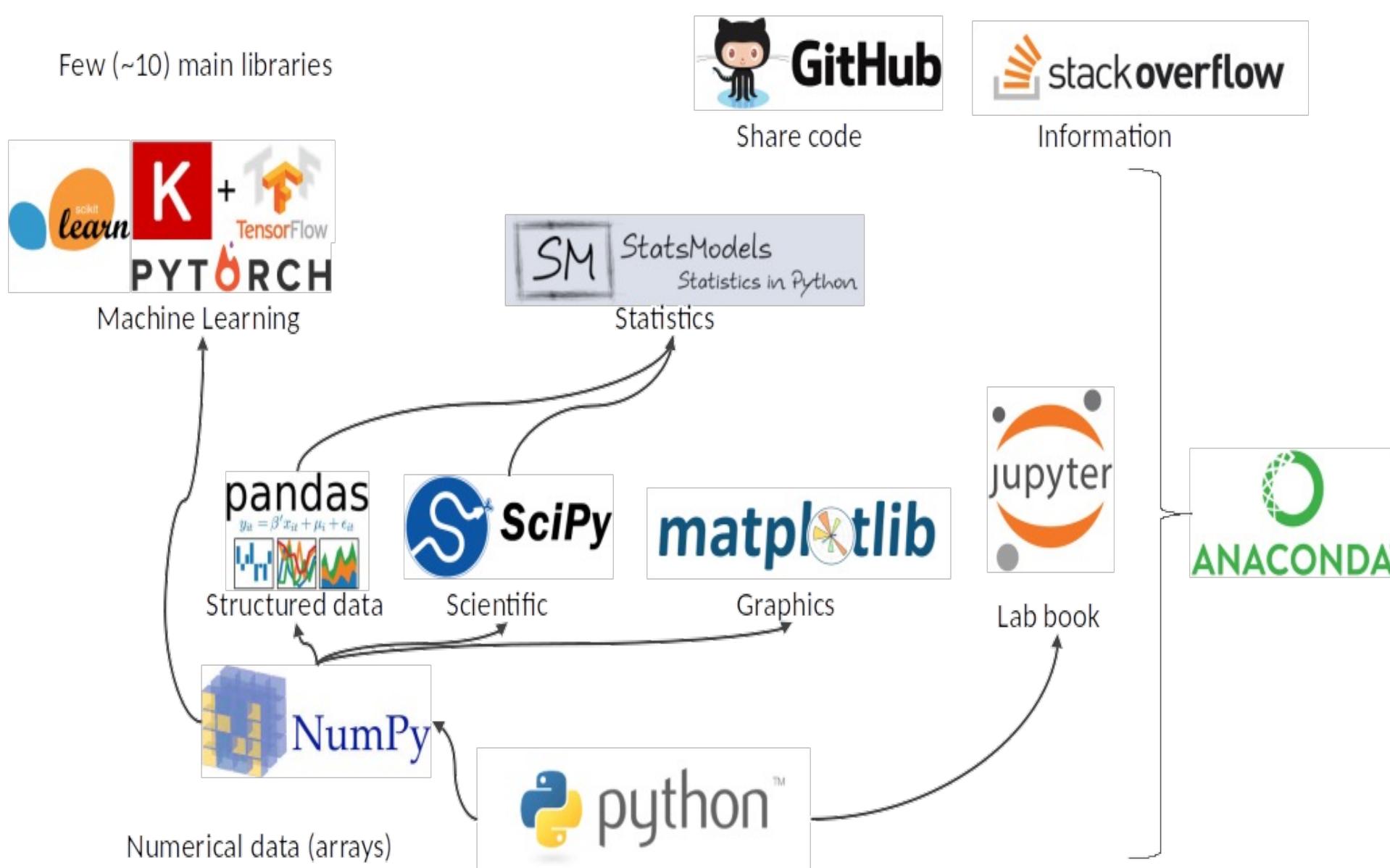
Top Programming Languages



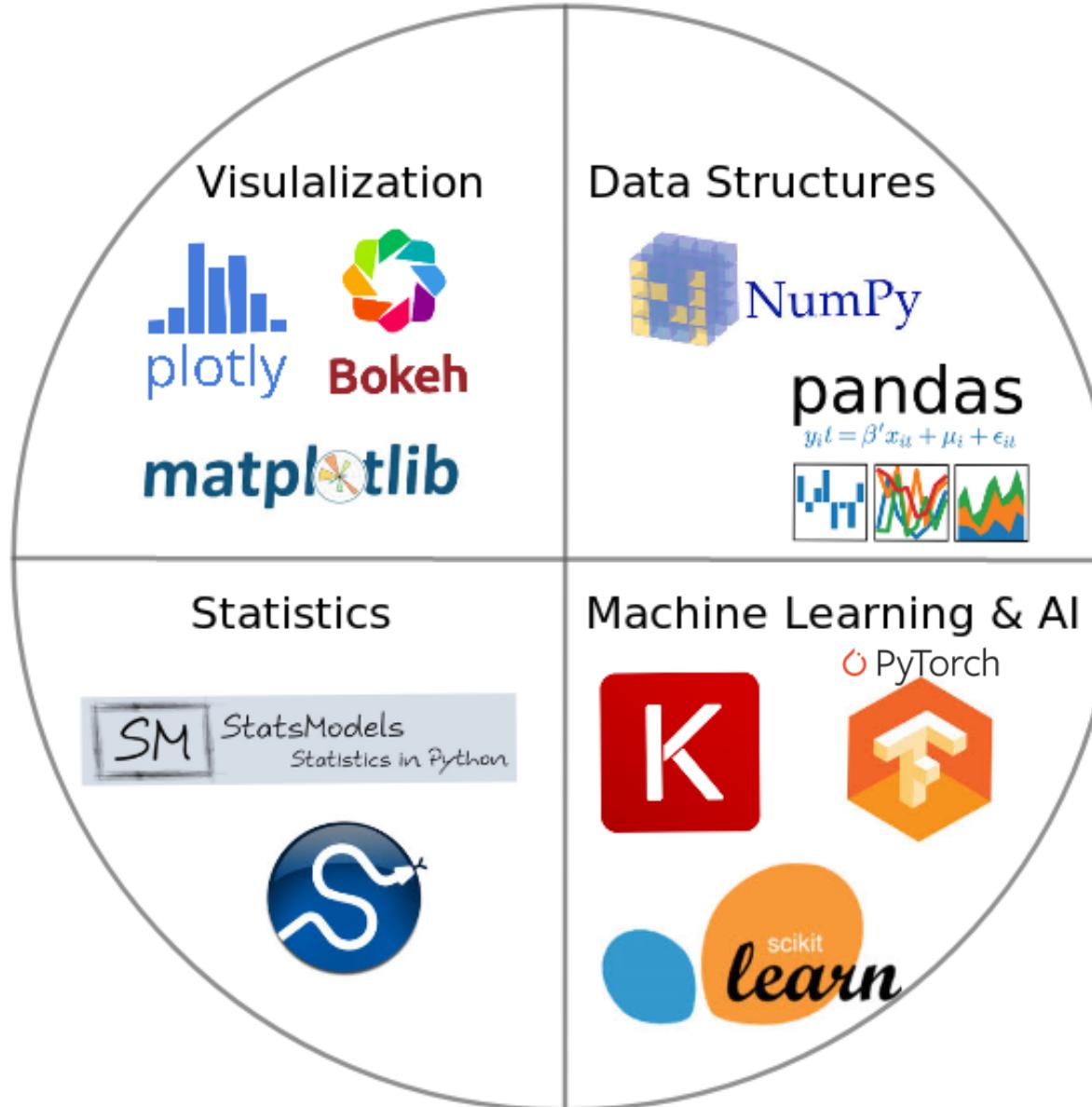


**Python is an
interpreted,
object-oriented,
high-level
programming language
with
dynamic semantics.**

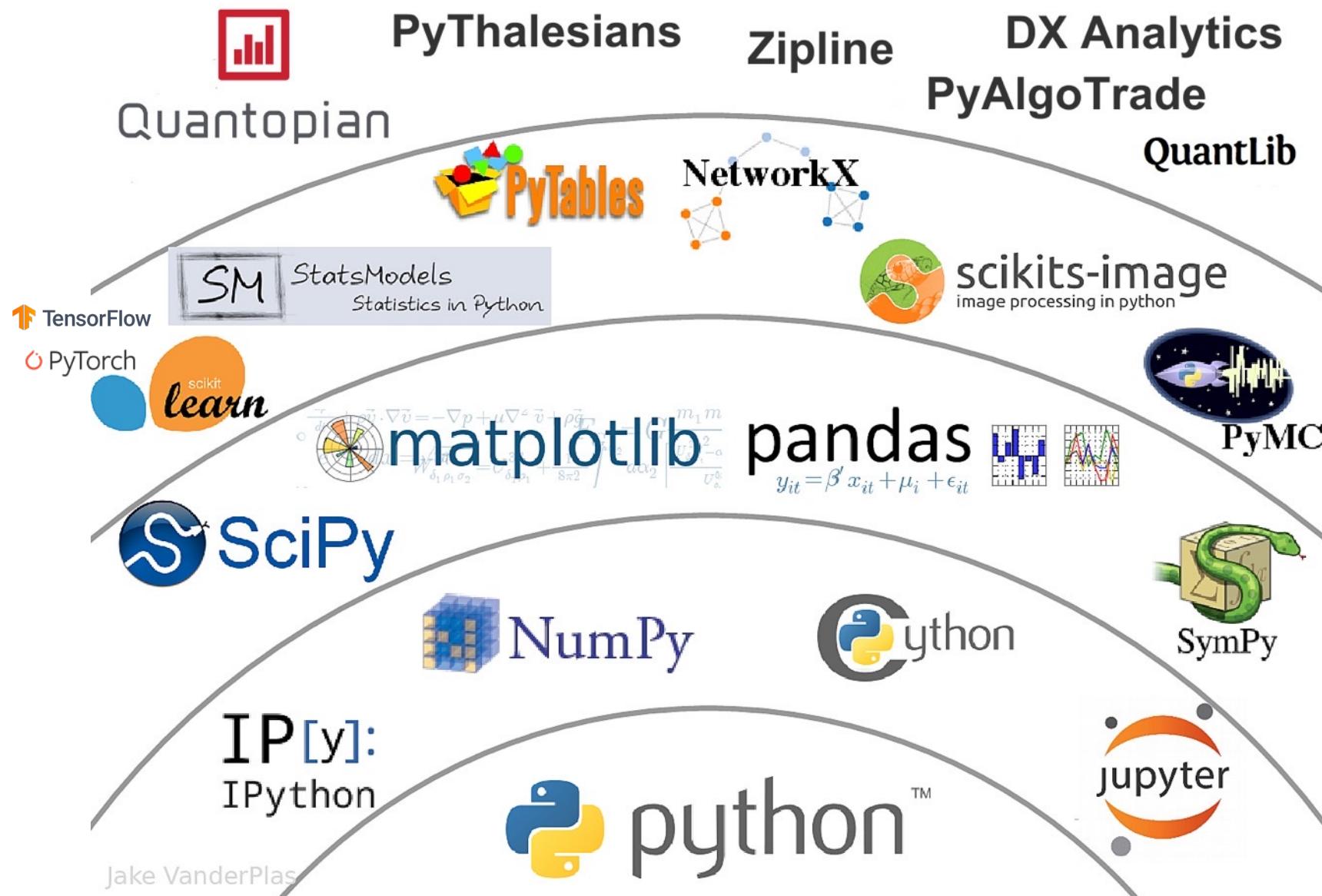
Python Ecosystem for Data Science



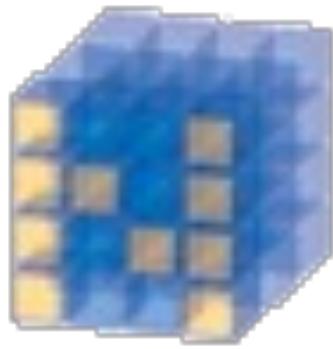
Python Ecosystem for Data Science



The Quant Finance PyData Stack



Numpy



NumPy
Base
N-dimensional array
package

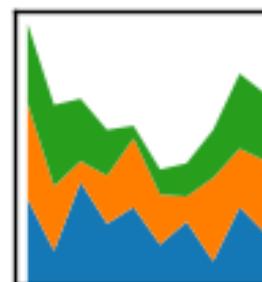
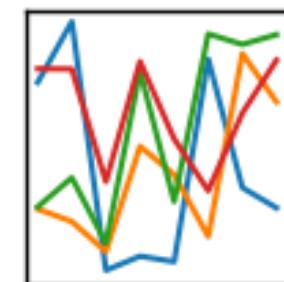
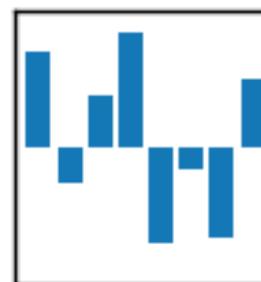
Python matplotlib

matplotlib

Python Pandas

pandas

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



W3Schools Python

HTML CSS JAVASCRIPT SQL PYTHON JAVA PHP HOW TO W3.CSS C C++ C# BOOTSTRAP REACT

Python Tutorial

Python HOME

- Python Intro
- Python Get Started
- Python Syntax
- Python Comments
- Python Variables
- Python Data Types
- Python Numbers
- Python Casting
- Python Strings
- Python Booleans
- Python Operators
- Python Lists
- Python Tuples
- Python Sets
- Python Dictionaries
- Python If...Else
- Python While Loops
- Python For Loops
- Python Functions

Python Tutorial

« Home

Next »

Learn Python

Python is a popular programming language.

Python can be used on a server to create web applications.

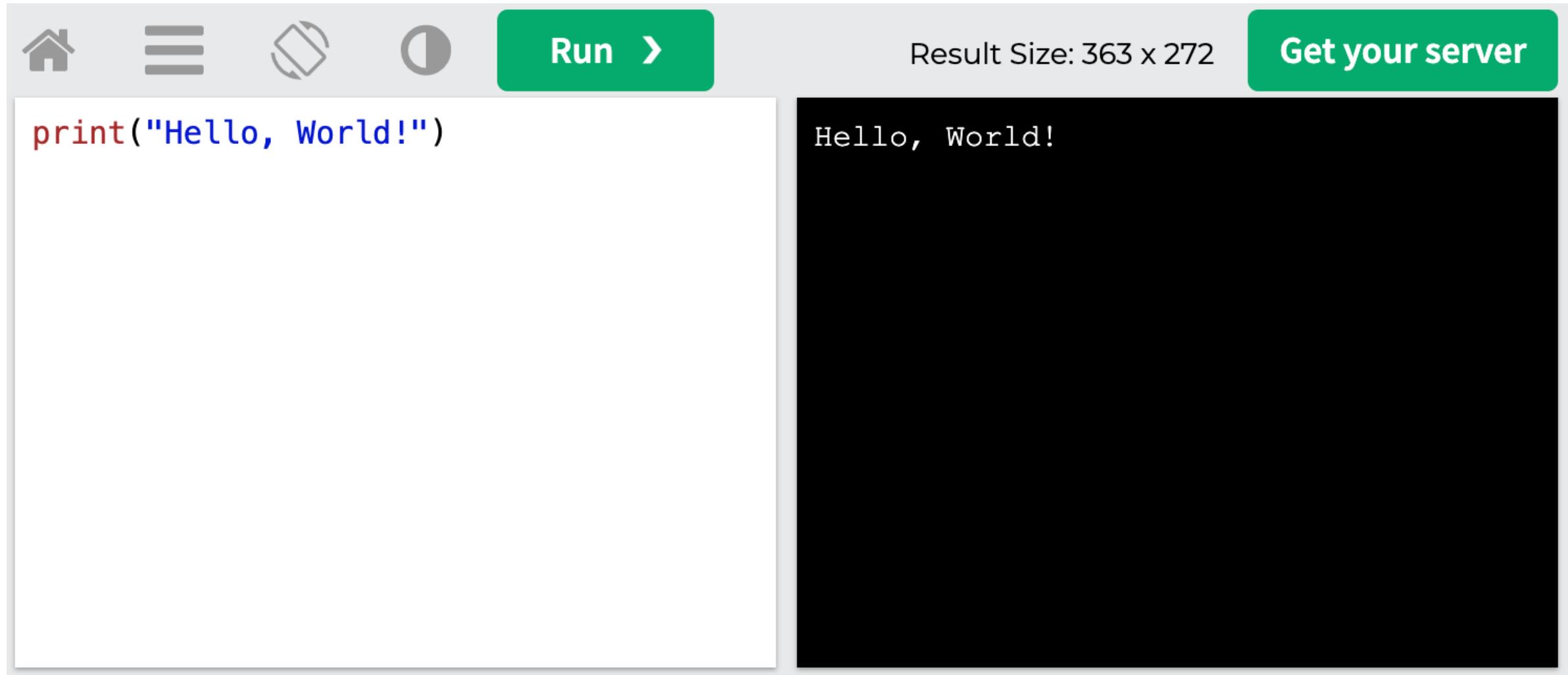
[Start learning Python now »](#)

Learning by Examples

With our "Try it Yourself" editor, you can edit Python code and view the result.

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

W3Schools Python: Try Python



The screenshot shows the W3Schools Python Try Python interface. At the top, there is a navigation bar with icons for Home, Menu, Device orientation, and Dark mode, followed by a green 'Run' button with a right-pointing arrow. To the right of the Run button, it says 'Result Size: 363 x 272'. Next to that is a green button labeled 'Get your server'. Below the navigation bar, on the left, is a code editor window containing the Python code: `print("Hello, World!")`. On the right is a results window with a black background displaying the output: `Hello, World!`.

LearnPython.org



learnpython.org

Home About Certify More Languages ▾

Python Java HTML Go C C++ JavaScript PHP Shell C# Perl Ruby Scala SQL

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

Google for Education > Python

Search

English



Filter

Overview

Python Set Up

Python Intro

Strings

Lists

Sorting

Dicts and Files

Regular Expressions

Utilities

Lecture Videos

1.1 Introduction, strings

1.2 Lists and sorting

1.3 Dicts and files

2.1 Regular expr

2.2 Utilities

2.3 Utilities urllib

2.4 Conclusions

Python Exercises

Home > Products > Google for Education > Python

Was this helpful?

Google's Python Class



Welcome to Google's Python Class -- this is a free class for people with a little bit of programming experience who want to learn Python. The class includes written materials, lecture videos, and lots of code exercises to practice Python coding. These materials are used within Google to introduce Python to people who have just a little programming experience. The first exercises work on basic Python concepts like strings and lists, building up to the later exercises which are full programs dealing with text files, processes, and http connections. The class is geared for people who have a little bit of programming experience in some language, enough to know what a "variable" or "if statement" is. Beyond that, you do not need to be an expert programmer to use this material.

To get started, the Python sections are linked at the left -- [Python Set Up](#) to get Python installed on your machine, [Python Introduction](#) for an introduction to the language, and then [Python Strings](#) starts the coding material, leading to the first exercise. The end of each written section includes a link to the code exercise for that section's material. The lecture videos parallel the written materials, introducing Python, then strings, then first exercises, and so on. At Google, all this material makes up an intensive 2-day class, so the videos are organized as the day-1 and day-2 sections.

This material was created by [Nick Parlante](#) working in the engEDU group at Google. Special thanks for the help from my Google colleagues John Cox, Steve Glassman, Piotr Kaminski, and Antoine Picard. And finally thanks to Google and my director Maggie Johnson for the enlightened generosity to put these materials out on the internet for free under the [Creative Commons Attribution 2.5](#) license -- share and enjoy!

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

Google Colab

The screenshot shows the Google Colaboratory interface. At the top, there's a navigation bar with tabs for 'FILE', 'EDIT', 'VIEW', 'INSERT', 'RUNTIME', 'TOOLS', and 'HELP'. Below the navigation bar, there are buttons for '+ CODE', '+ TEXT', 'CELL UP', 'CELL DOWN', 'COPY TO DRIVE', 'CONNECT', and 'EDITING'. On the left side, there's a sidebar with a 'Table of contents' section containing links to 'Getting Started', 'Highlighted Features', 'TensorFlow execution', 'GitHub', 'Visualization', 'Forms', 'Examples', and 'Local runtime support'. There's also a '+ SECTION' button. The main content area features a 'Welcome to Colaboratory!' section with a 'CO' logo, followed by a 'Getting Started' section with a list of links, a 'Highlighted Features' section with a 'Seedbank' subsection, and a 'TensorFlow execution' section with a code snippet.

Hello, Colaboratory

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

File Edit View Insert Runtime Tools Help

+ CODE + TEXT CELL UP CELL DOWN COPY TO DRIVE CONNECT EDITING

Table of contents Code snippets Files

Getting Started

Highlighted Features

TensorFlow execution

GitHub

Visualization

Forms

Examples

Local runtime support

+ SECTION

Welcome to Colaboratory!

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. See our [FAQ](#) for more info.

Getting Started

- [Overview of Colaboratory](#)
- [Loading and saving data: Local files, Drive, Sheets, Google Cloud Storage](#)
- [Importing libraries and installing dependencies](#)
- [Using Google Cloud BigQuery](#)
- [Forms, Charts, Markdown, & Widgets](#)
- [TensorFlow with GPU](#)
- [Machine Learning Crash Course: Intro to Pandas & First Steps with TensorFlow](#)

Highlighted Features

Seedbank

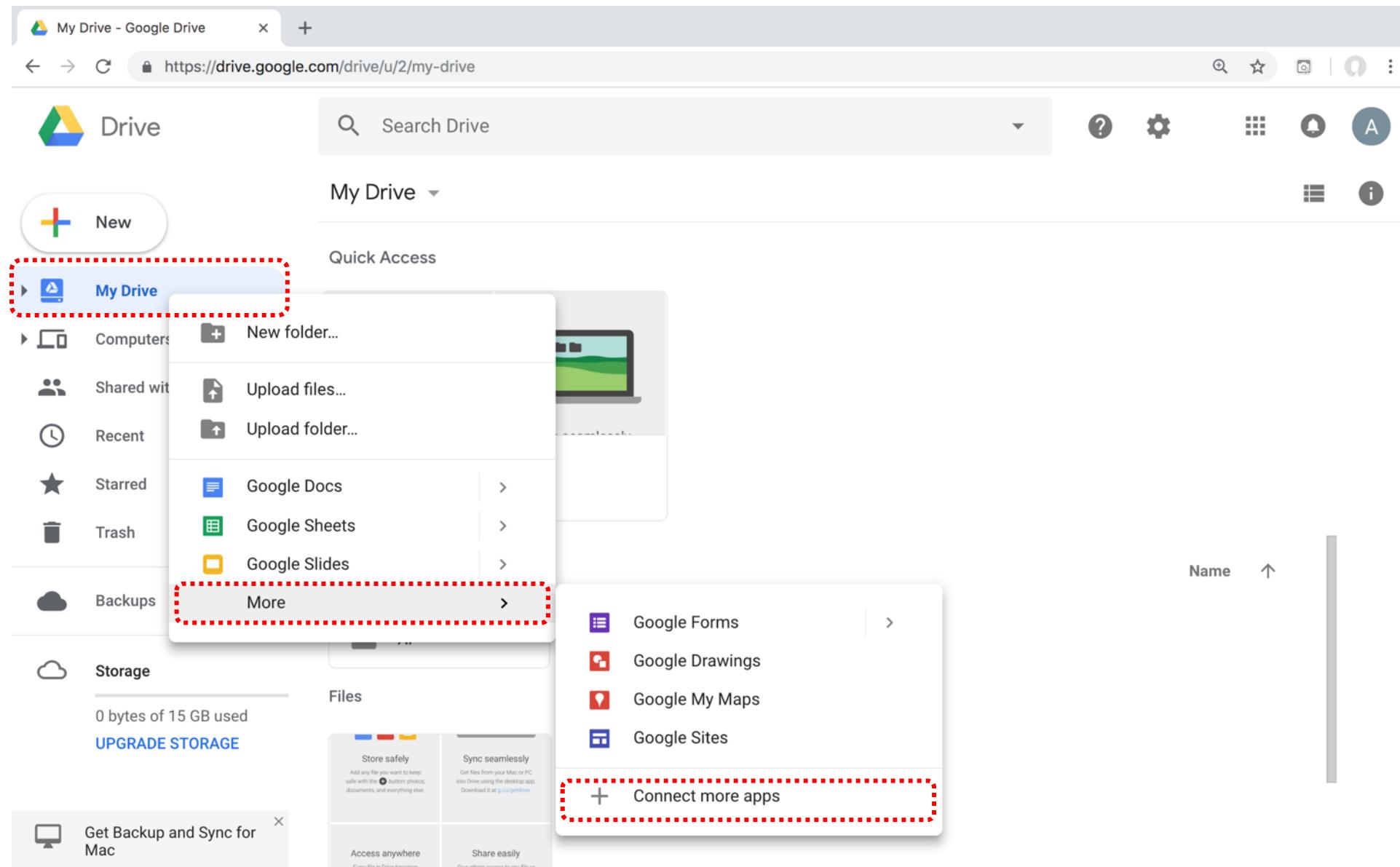
Looking for Colab notebooks to learn from? Check out [Seedbank](#), a place to discover interactive machine learning examples.

TensorFlow execution

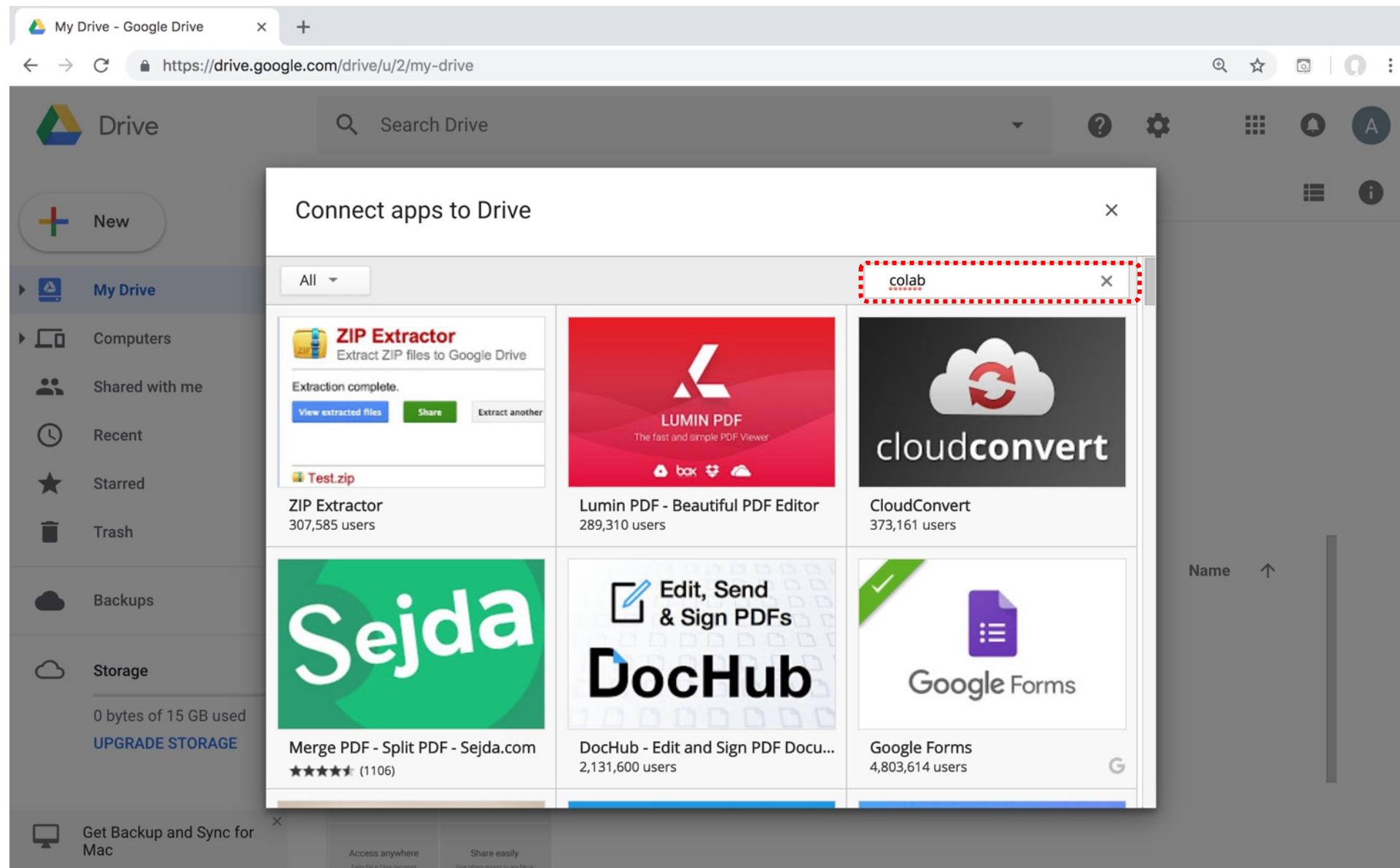
Colaboratory allows you to execute TensorFlow code in your browser with a single click. The example below adds two matrices.

$$\begin{bmatrix} 1. & 1. & 1. \end{bmatrix} + \begin{bmatrix} 1. & 2. & 3. \end{bmatrix} = \begin{bmatrix} 2. & 3. & 4. \end{bmatrix}$$

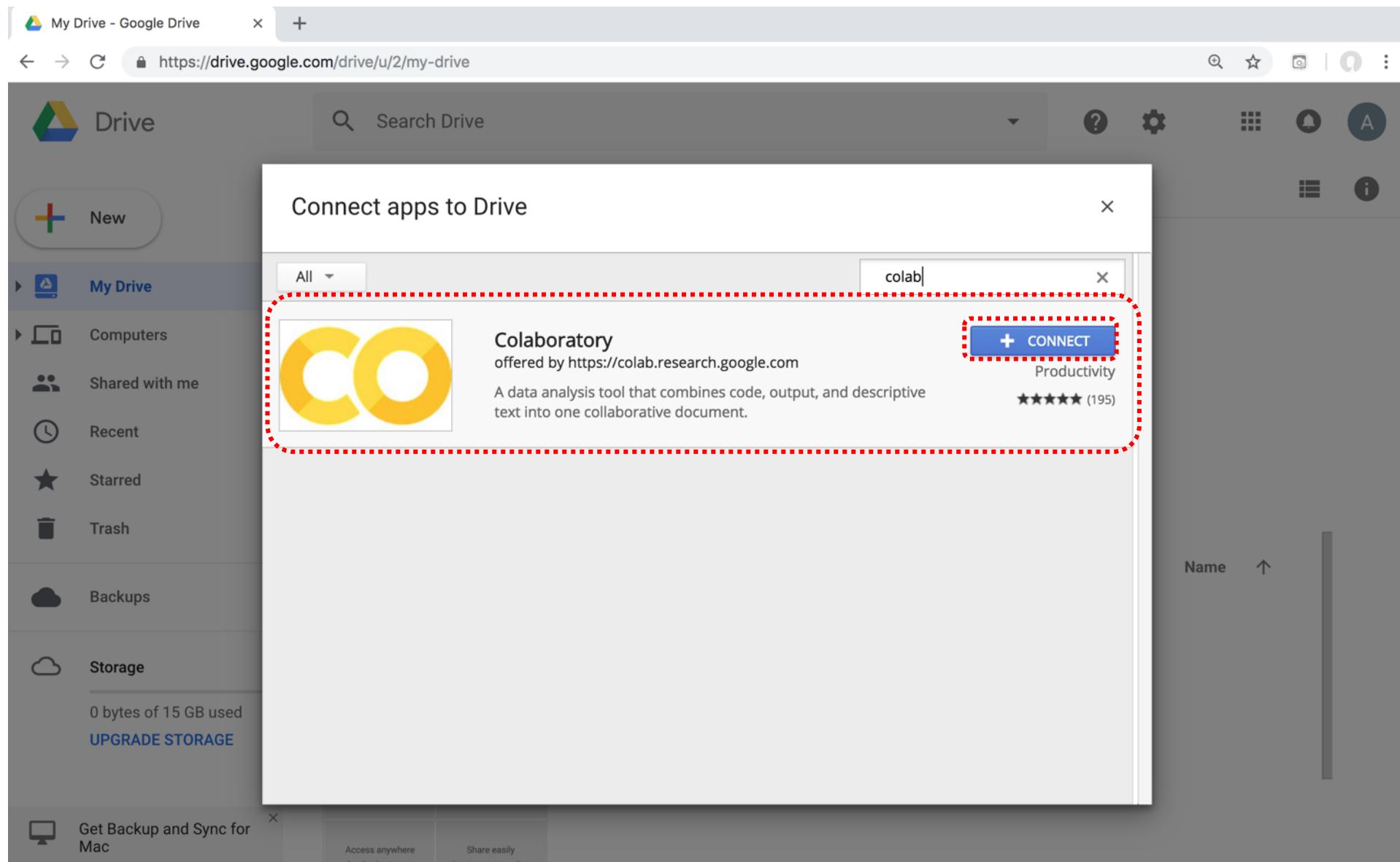
Connect Google Colab in Google Drive



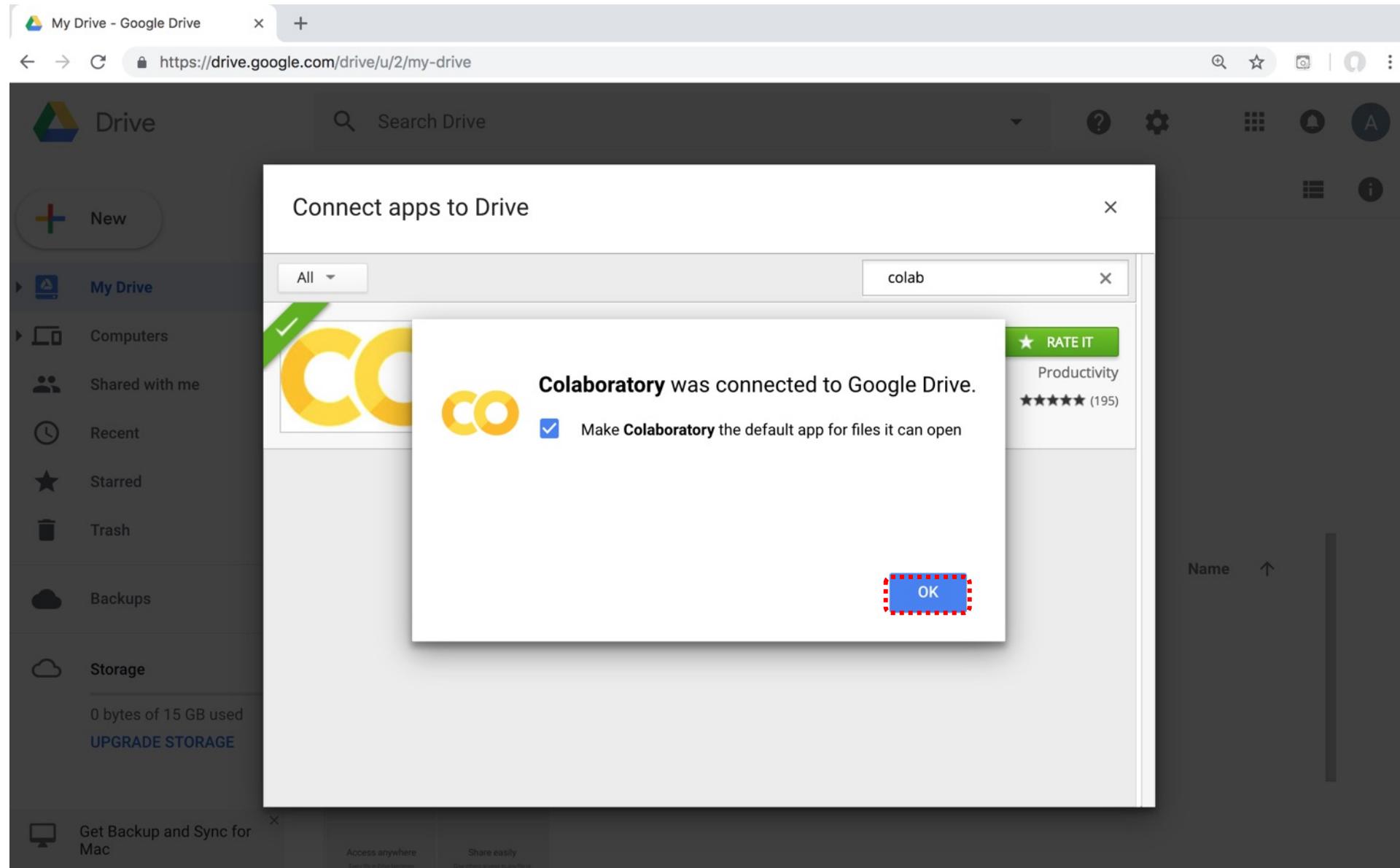
Google Colab



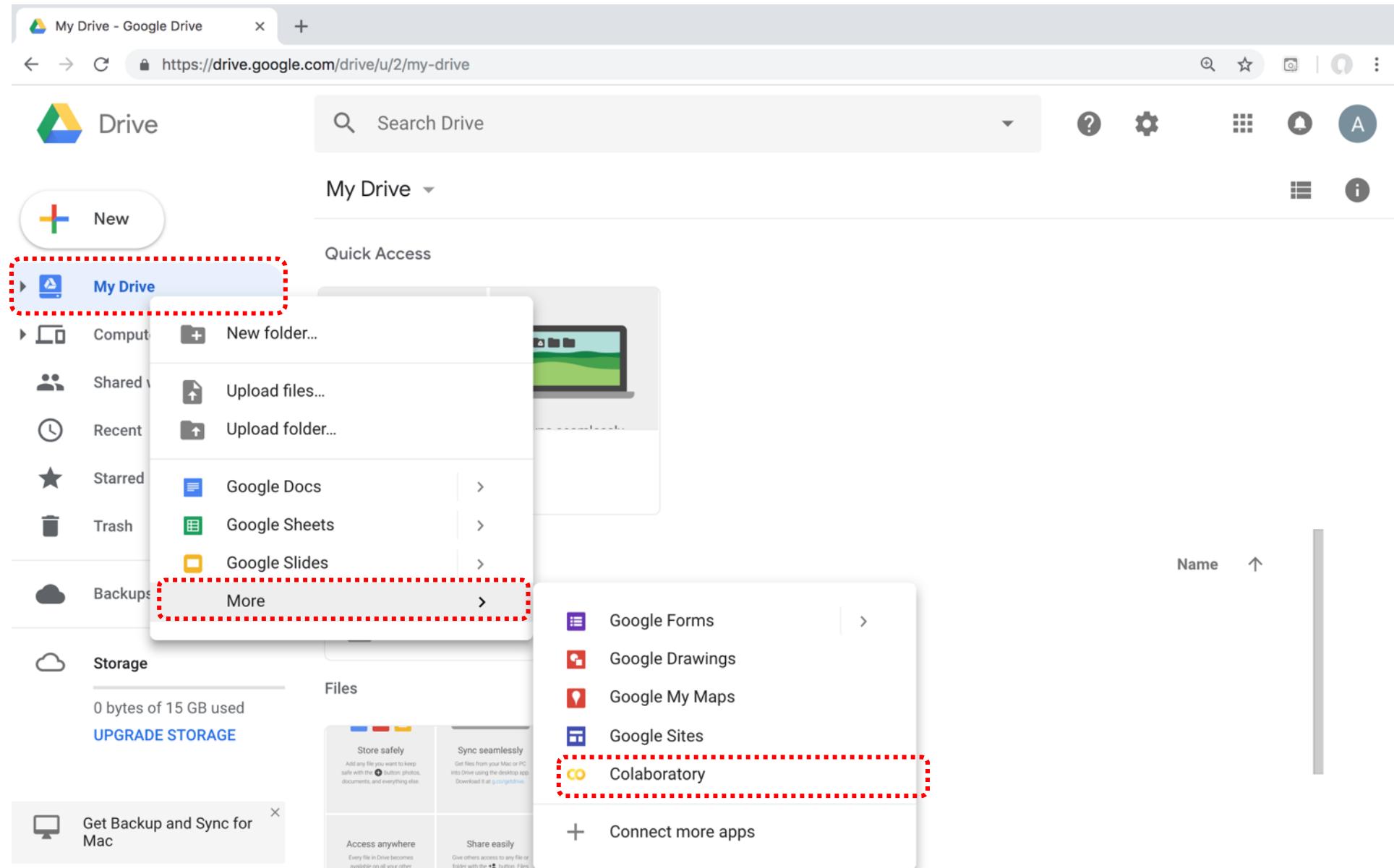
Google Colab



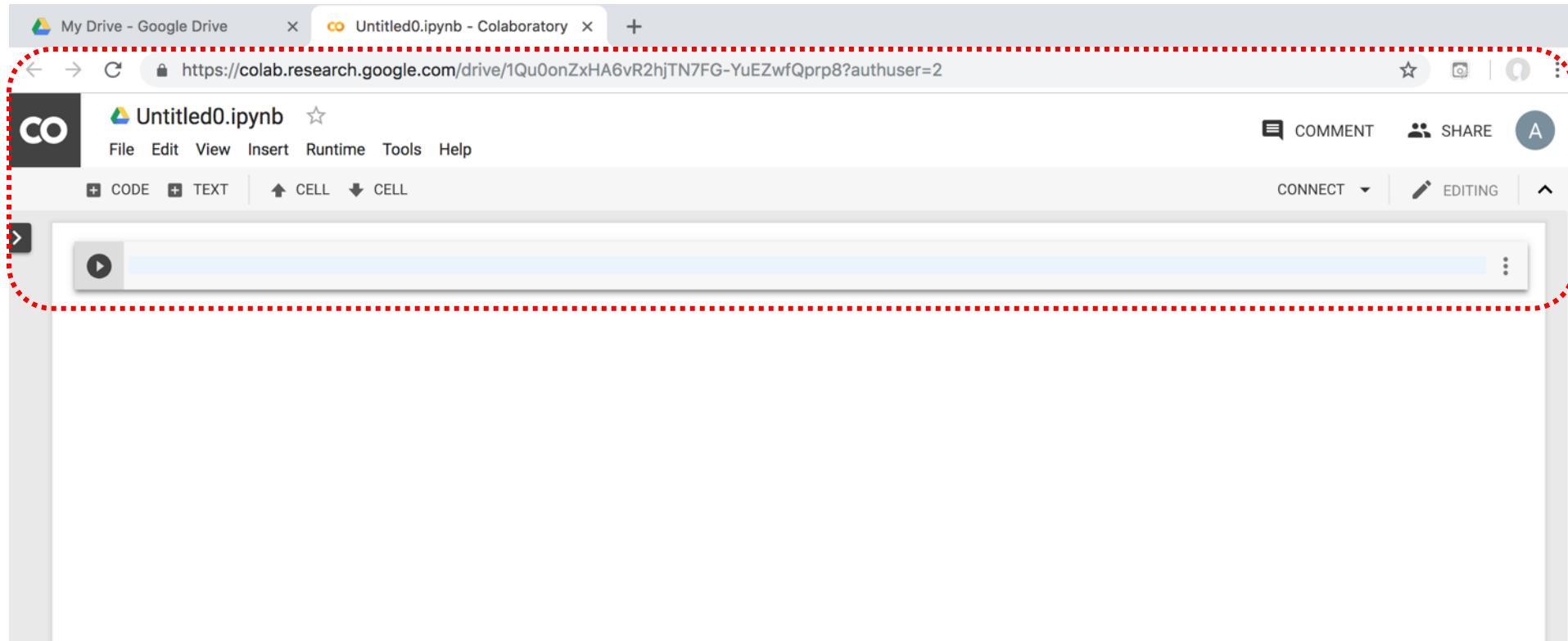
Connect Colaboratory to Google Drive



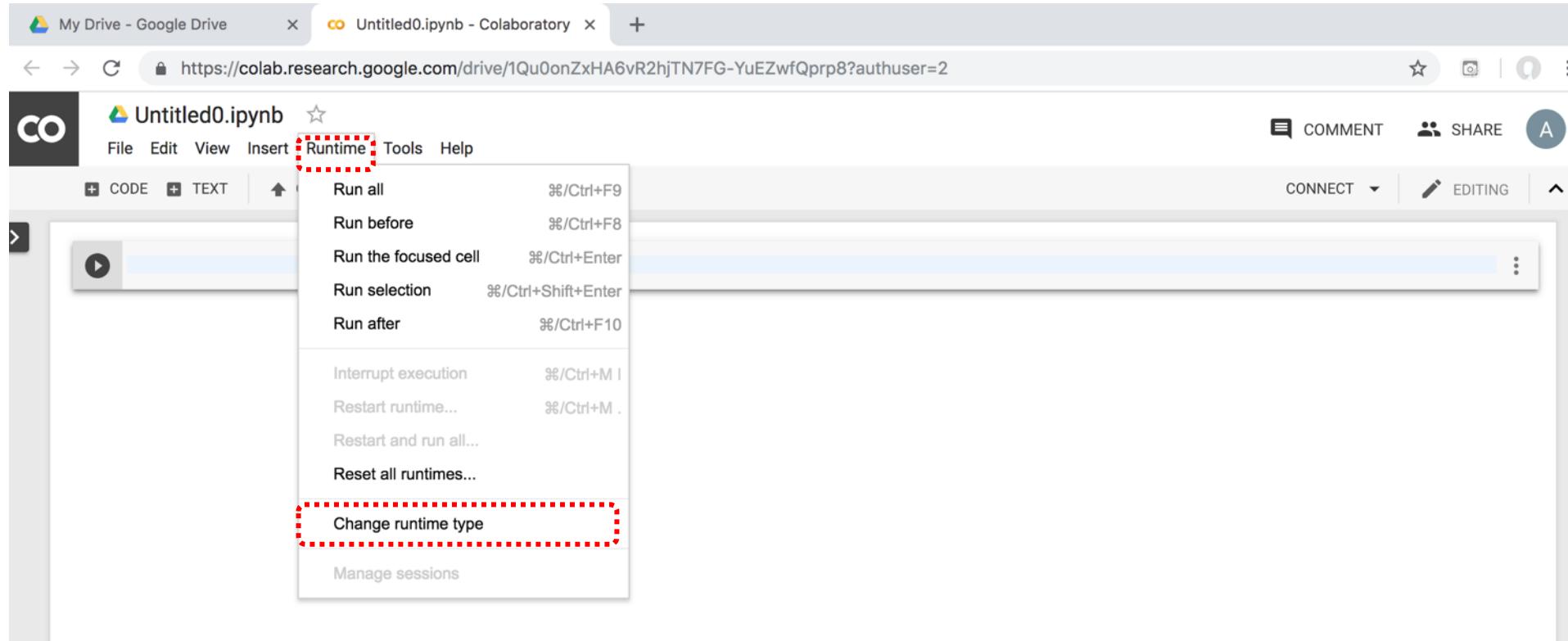
Google Colab



Google Colab



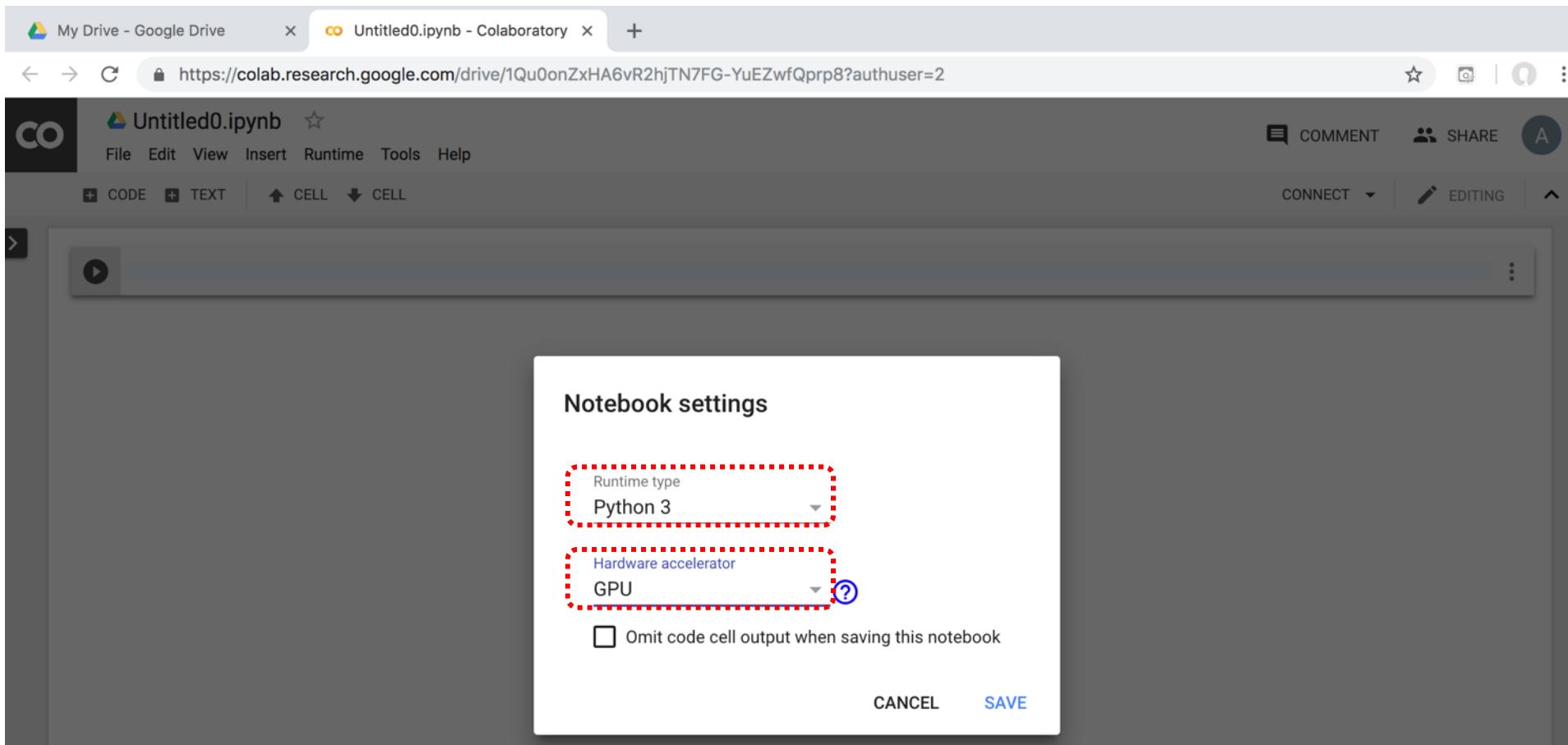
Google Colab



Run Jupyter Notebook

Python3 GPU

Google Colab



Google Colab Python Hello World

print('Hello World')



The screenshot shows a Google Colab notebook titled "Untitled0.ipynb". The code cell contains the Python command `print('Hello World')`. The output of the cell is "Hello World", displayed in a light blue box. The Colab interface includes a toolbar with "CODE", "TEXT", "CELL", and "COMMENT" buttons, and a status bar indicating "CONNECTED".

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a toolbar with file operations, a sidebar with a "CODE" tab selected, and a main workspace with four code cells. The first cell contains a script to calculate future value using simple interest. The second cell uses a different approach with variables. The third cell defines a function named "getfv". The fourth cell demonstrates an if-else statement. All cells produce the same output: 194.87.

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

amount = 100
interest = 10 #10% = 0.01 * 10
years = 7
future_value = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))

# Python Function def
def getfv(pv, r, n):
    fv = pv * ((1 + (r)) ** n)
    return fv
fv = getfv(100, 0.1, 7.)
print(round(fv, 2))

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

<https://tinyurl.com/aintpuppython101>



Anaconda

The Most Popular

Python

Data Science Platform

Download Anaconda



Products ▾ Pricing Solutions ▾ Resources ▾ Partners ▾ Blog Company ▾

Contact Sales

Data science technology for
a better world.

Anaconda offers the easiest way to perform Python/R data science and machine learning on a single machine. Start working with thousands of open-source packages and libraries today.

Download

Get Additional Installers



<https://www.anaconda.com/download>

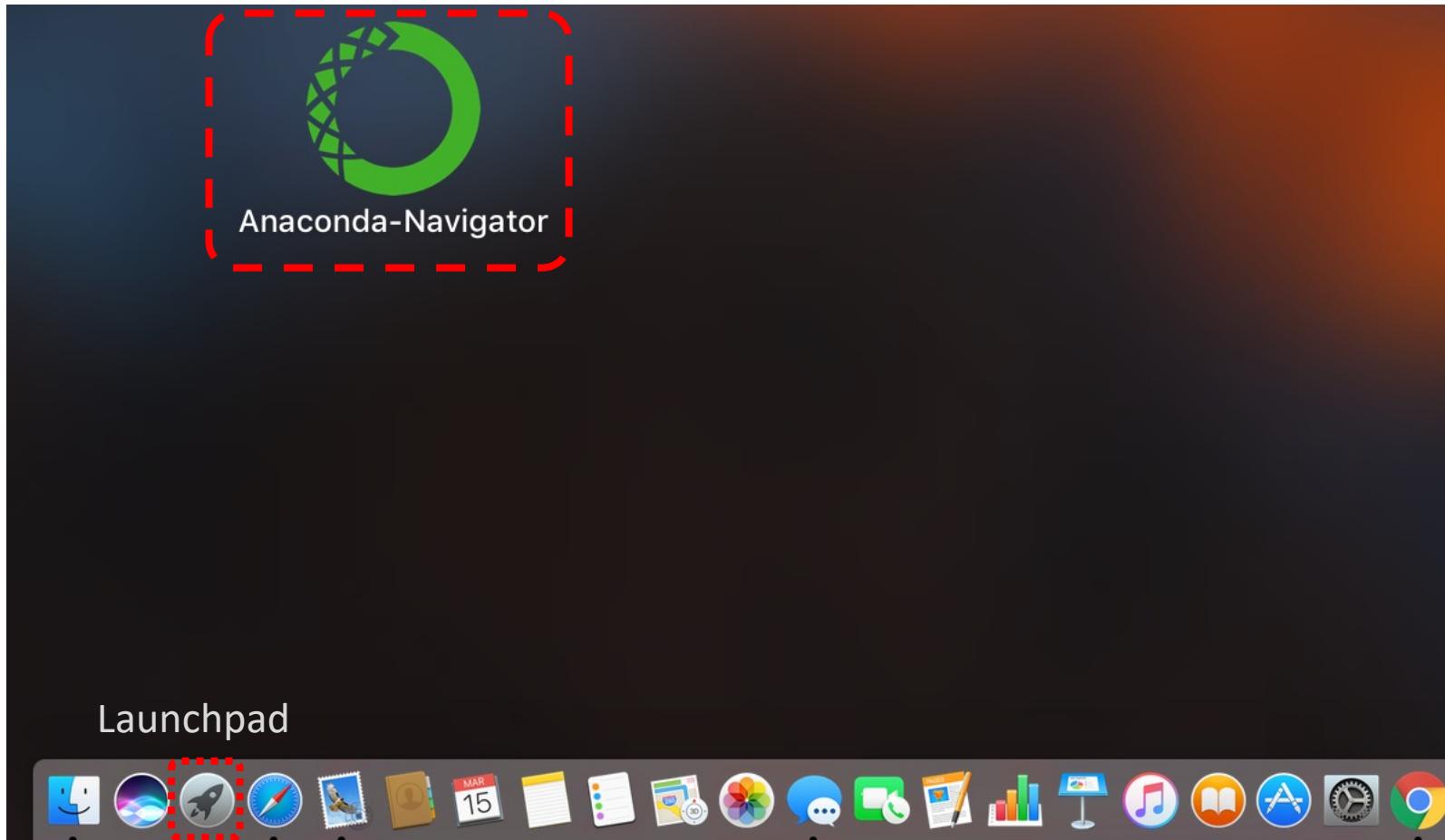




Python

Helloworld

Anaconda-Navigator



Anaconda Navigator

The screenshot shows the Anaconda Navigator interface. On the left is a sidebar with icons for Home, Environments, Learning, and Community, along with links for Documentation, Developer Blog, and Feedback. At the bottom are social media links for Twitter, YouTube, and GitHub.

The main area displays a grid of applications:

- jupyterlab** (version 0.31.5): An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. Includes a "Launch" button.
- jupyter notebook** (version 5.4.0): Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. Includes a "Launch" button.
- qtconsole** (version 4.3.1): PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. Includes a "Launch" button.
- spyder** (version 3.2.6): Scientific PYthon Development EnvIRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features. Includes a "Launch" button.
- vscode** (version 1.22.2): Streamlined code editor with support for development operations like debugging, task running and version control. Includes a "Launch" button.
- glueviz** (version 0.12.4): Multidimensional data visualization across files. Explore relationships within and among related datasets. Includes an "Install" button.

At the top right, there is a "Sign in to Anaconda Cloud" button and a "Refresh" button. The title bar says "Anaconda Navigator".

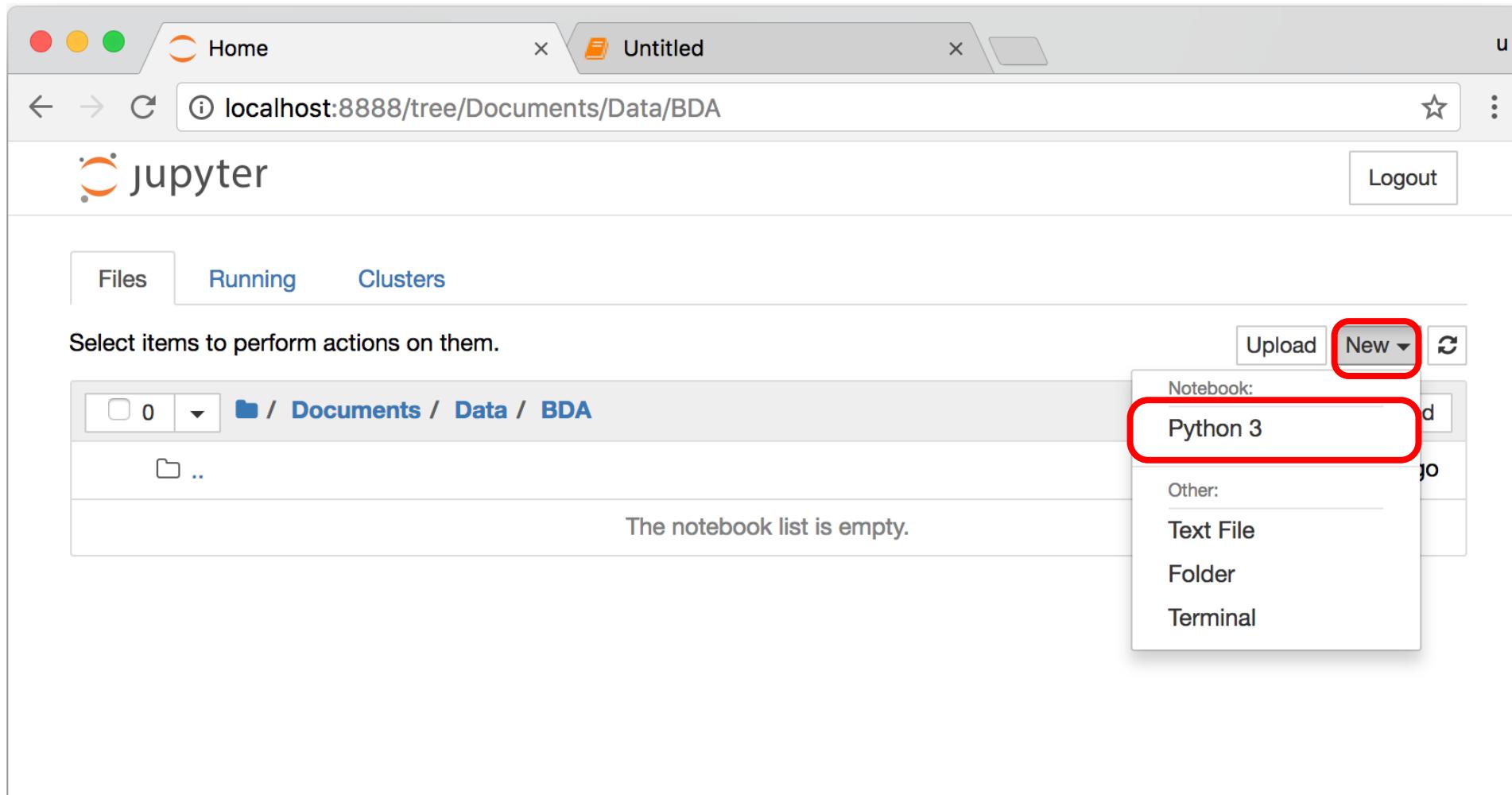
Jupyter Notebook

A screenshot of the Jupyter Notebook web interface. The title bar shows the URL `localhost:8888/tree/Documents/Data/BDA`. The main area displays a file tree under the path `Documents / Data / BDA`, which is currently empty. A red dashed box highlights the file tree area. The interface includes tabs for `Files`, `Running`, and `Clusters`, and buttons for `Upload`, `New`, and refresh. The status bar at the bottom indicates the list is empty.

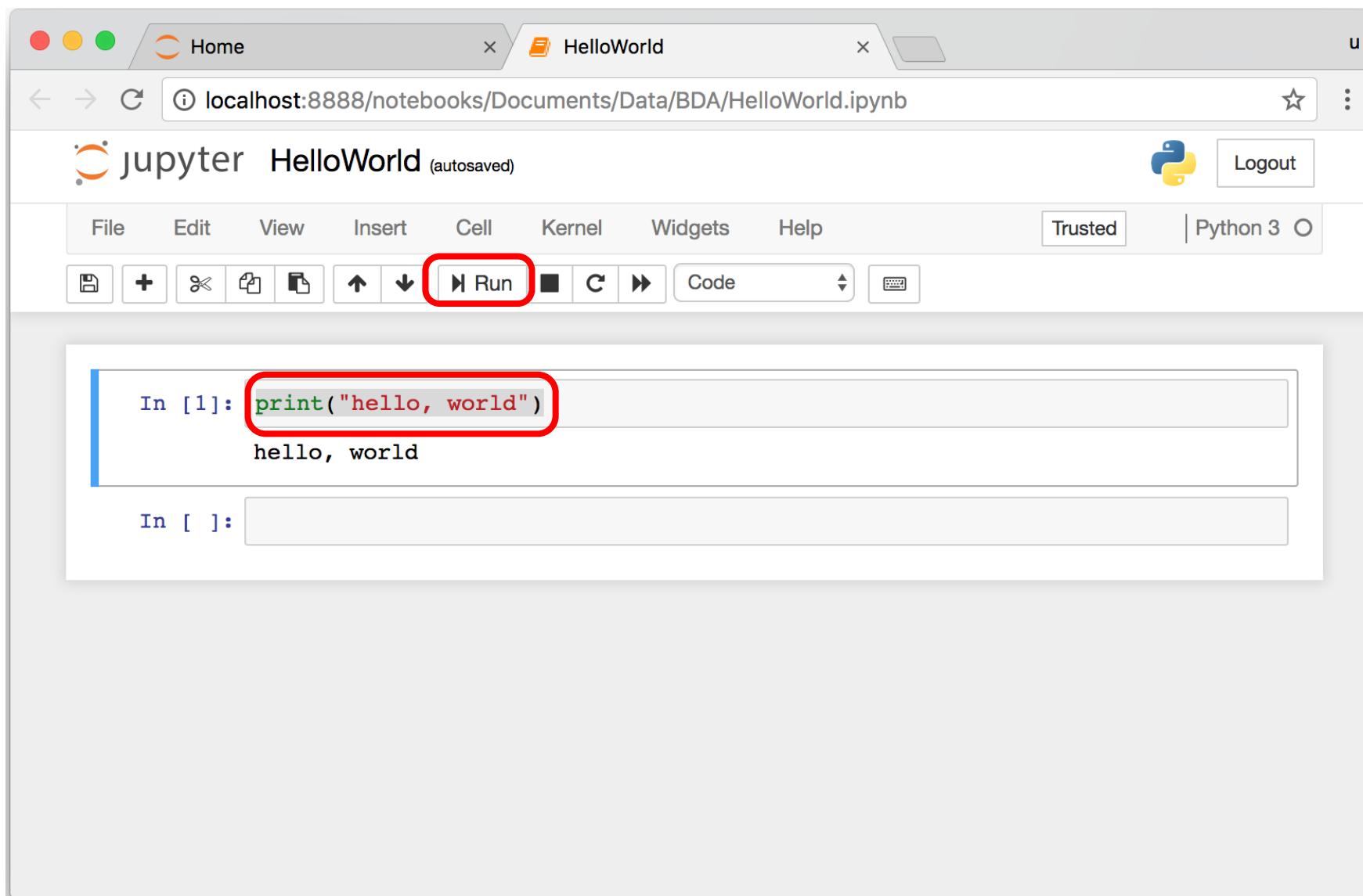
The notebook list is empty.

Jupyter Notebook

New Python 3



print("hello, world")





Python Fiddle

The screenshot shows the Python Fiddle web application interface. At the top, there's a header bar with a back/forward button, a search bar containing "pythonfiddle.com", and a menu icon. Below the header are navigation buttons for "Run", "Reset", "Share", "Import", "Login", and "Language". To the right of these is a "G+1" button and a "2.6k" badge.

The main area features a logo for "Python Fiddle" with a cloud icon containing a Python logo, and the text "Python Cloud IDE". On the left, a sidebar titled "Examples" lists various Python concepts like "Chaining comparison operators", "Decorators", and "Function argument unpacking". Below this are sections for "Packages" and "Hotkeys".

The central code editor contains the following Python code:

```
1 print("Hello Python Fiddle")  
2
```

At the bottom of the code editor is a "Save" button. To the right of the code editor are fields for "Title", "Description", and "Tags", with a note explaining that tags should be a comma-separated list.

<http://pythonfiddle.com/>



Text input and output

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

```
print("Hello World\nThis is a message")
```

```
x = 3  
print(x)
```

```
x = 2  
y = 3  
print(x, ' ', y)
```

```
name = input("Enter a name: ")
```

```
x = int(input("What is x? "))
```

```
x = float(input("Write a number "))
```

Python in Google Colab

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

The screenshot shows a Google Colab notebook titled "python101.ipynb". The notebook contains four code cells:

- Cell 1:**

```
1 print("hello, world")
```

 Output: `hello, world`
- Cell 2:**

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

 Output: `Python Version: 3.6.6`
- Cell 3:**

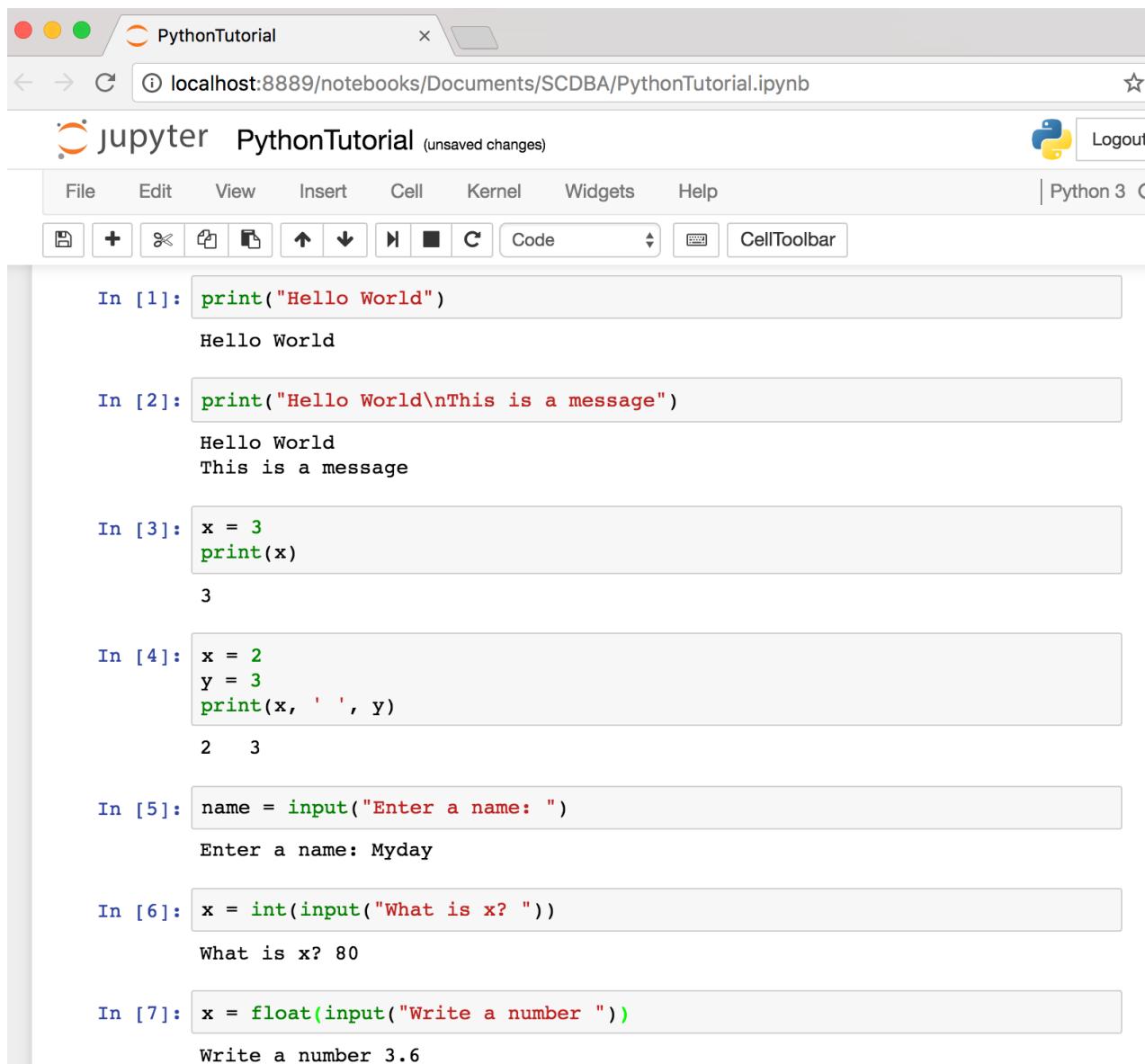
```
1 # https://www.learnpython.org/en/
2 # LearnPython.org interactive Python tutorial
3 print("Hello World")
4 print("Hello World\nThis is a message")
5 x = 3
6 print(x)
7 x = 2
8 y = 3
9 print(x, ' ', y).
```

 Output:
`Hello World
Hello World
This is a message
3
2 3`
- Cell 4:**

```
1 # Python Variables
2 x = 2
3 price = 2.5
4 word = 'Hello'
5
6 word = 'Hello'
7 word = "Hello"
8 word = '''Hello'''
```

<https://tinyurl.com/aintpuppython101>

Text input and output



```
In [1]: print("Hello World")
Hello World

In [2]: print("Hello World\nThis is a message")
Hello World
This is a message

In [3]: x = 3
print(x)
3

In [4]: x = 2
y = 3
print(x, ' ', y)
2 3

In [5]: name = input("Enter a name: ")
Enter a name: Myday

In [6]: x = int(input("What is x? "))
What is x? 80

In [7]: x = float(input("Write a number "))
Write a number 3.6
```

Variables

```
x = 2  
price = 2.5  
word = 'Hello'
```

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

```
x = 2  
x = x + 1  
x = 5
```

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)
```

```
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
```



BMI Calculator in Python

```
height_cm = float(input("Enter your height in cm: "))
weight_kg = float(input("Enter your weight in kg: "))

height_m = height_cm/100
BMI = (weight_kg/(height_m**2))

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

BMI Calculator in Python

jupyter PythonTutorial Last Checkpoint: a minute ago (unsaved changes)  Logout | Python 3

In [1]:

```
height_cm = float(input("Enter your height in cm: "))
weight_kg = float(input("Enter your weight in kg: "))

height_m = height_cm/100
BMI = (weight_kg/(height_m**2))

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

Enter your height in cm: 170
Enter your weight in kg: 60
Your BMI is: 20.8

In []:

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

Future Value (FV)

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

```
print(100 * 1.1 ** 7)
```

```
194.871000000012
```

Future Value (FV)

```
pv = 100  
r = 0.1  
n = 7
```

```
fv = pv * ((1 + (r)) ** n)  
print(round(fv, 2))
```

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

194.87

Future Value (FV)

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

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

```
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

if statements

> greater than
< smaller than
== equals
!= is not

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

Pass

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

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)
# grade = "A"
```

A

<http://pythontutor.com/visualize.html>
<https://goo.gl/E6w5ph>

for loops

```
for i in range(1,11):  
    print(i)
```

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

for loops

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

```
9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

while loops

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

```
10  
11  
12  
13  
14  
15  
16  
17  
18  
19
```

def Functions

```
def convertCMTOM(xcm):  
    m = xcm/100  
    return m  
  
cm = 180  
m = convertCMTOM(cm)  
print(str(m))
```

1.8

Lists []

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

```
4  
60  
70  
90
```

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

Dictionary {key : value}

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

Dictionary

'EN' → 'English'

'FR' → 'French'

English

Sets {}

```
animals = {'cat', 'dog'}
```

```
animals = {'cat', 'dog'}
print('cat' in animals) # Check if an element is in a set; prints "True"
print('fish' in animals) # prints "False"
animals.add('fish') # Add an element to a set
print('fish' in animals) # Prints "True"
print(len(animals)) # Number of elements in a set; prints "3"
animals.add('cat') # Adding an element that is already in the set does nothing
print(len(animals)) # Prints "3"
animals.remove('cat') # Remove an element from a set
print(len(animals)) # Prints "2"
```

```
True
False
True
3
3
2
```

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

File Input / Output

```
with open('myfile.txt', 'w') as file:  
    file.write('Hello World\nThis is Python File Input Output')  
  
with open('myfile.txt', 'r') as file:  
    text = file.read()  
print(text)
```

```
with open('myfile.txt', 'w') as file:  
    file.write('Hello World\nThis is Python File Input Output')
```

```
with open('myfile.txt', 'r') as file:  
    text = file.read()  
print(text)
```

```
Hello World  
This is Python File Input Output
```

```
text
```

```
'Hello World\nThis is Python File Input Output'
```

File Input / Output

```
with open('myfile.txt', 'a+') as file:  
    file.write('\n' + 'New line')  
  
with open('myfile.txt', 'r') as file:  
    text = file.read()  
print(text)
```

```
with open('myfile.txt', 'a+') as file:  
    file.write('\n' + 'New line')
```

```
with open('myfile.txt', 'r') as file:  
    text = file.read()  
print(text)
```

Hello World
This is Python File Input Output
New line

try except finally

```
try:  
    file = open("myfile.txt")  
    #file = open("myfile.txt", 'w')  
    file.write("Python write file")  
    print("file saved")  
  
except:  
    print("Exception file Error")  
  
finally:  
    file.close()  
    print("finally process")
```

Exception file Error
finally process

class

```
class Person:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
    def myfunc(self):  
        print("Hello my name is " + self.name)  
  
p1 = Person("Alan", 20)  
p1.myfunc()  
print(p1.name)  
print(p1.age)
```

Hello my name is Alan
Alan
20

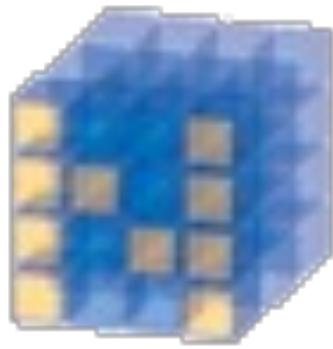
Big Data Analytics

with

Numpy

in Python

Numpy



NumPy
Base
N-dimensional array
package

NumPy
is the
fundamental package
for
scientific computing
with Python.



NumPy

- **NumPy provides a multidimensional array object to store homogenous or heterogeneous data; it also provides optimized functions/methods to operate on this array object.**

NumPy



NumPy

Scipy.org

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the [BSD license](#), enabling reuse with few restrictions.

Getting Started

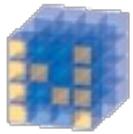
-
- [Getting NumPy](#)
 - [Installing the SciPy Stack](#)
 - [NumPy and SciPy documentation page](#)
 - [NumPy Tutorial](#)
 - [NumPy for MATLAB® Users](#)
 - [NumPy functions by category](#)
 - [NumPy Mailing List](#)

For more information on the SciPy Stack (for which NumPy provides the fundamental array data structure), see [scipy.org](#).

About NumPy

License

Old array packages



NumPy

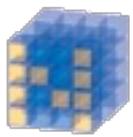
NumPy ndarray

One-dimensional Array (1-D Array)

0	1	n-1
1	2	3

Two-dimensional Array (2-D Array)

0	1	n-1			
0	1	2	3	4	5
1	6	7	8	9	10
m-1	11	12	13	14	15
16	17	18	19	20	



NumPy

NumPy

```
v = list(range(1, 6))
```

```
v
```

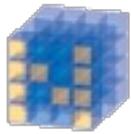
```
2 * v
```

```
import numpy as np
```

```
v = np.arange(1, 6)
```

```
v
```

```
2 * v
```



NumPy

Base

N-dimensional
array package

```
1 v = list(range(1, 6))  
2 v
```

```
[1, 2, 3, 4, 5]
```

```
1 2 * v
```

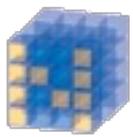
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

```
1 import numpy as np  
2 v = np.arange(1, 6)  
3 v
```

```
array([1, 2, 3, 4, 5])
```

```
1 2 * v
```

```
array([ 2,  4,  6,  8, 10])
```



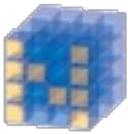
NumPy

NumPy Create Array

```
import numpy as np  
a = np.array([1, 2, 3])  
b = np.array([4, 5, 6])  
c = a * b  
c
```

```
import numpy as np  
a = np.array([1, 2, 3])  
b = np.array([4, 5, 6])  
c = a * b  
c
```

```
array([ 4, 10, 18])
```



NumPy

NumPy

```
1 import numpy as np
2
3 a = np.zeros((2,2)) # Create an array of all zeros
4 print(a)           # Prints "[[ 0.  0.]
5                      #          [ 0.  0.]]"
6
7 b = np.ones((1,2)) # Create an array of all ones
8 print(b)           # Prints "[[ 1.  1.]]"
9
10 c = np.full((2,2), 7) # Create a constant array
11 print(c)            # Prints "[[ 7.  7.]
12                      #          [ 7.  7.]]"
13
14 d = np.eye(2)       # Create a 2x2 identity matrix
15 print(d)            # Prints "[[ 1.  0.]
16                      #          [ 0.  1.]]"
17
18 e = np.random.random((2,2)) # Create an array filled with random values
19 print(e)             # Might print "[[ 0.91940167  0.08143941]
20                      #          [ 0.68744134  0.87236687]]"
```

```
[[0.  0.]
 [0.  0.]]
[[1.  1.]]
[[7  7]
 [7  7]]
[[1.  0.]
 [0.  1.]]
[[0.66258211  0.65552598]
 [0.00429934  0.21695824]]
```

Numpy Quickstart Tutorial



SciPy.org



[Scipy.org](#)

[Docs](#)

[NumPy v1.13.dev0 Manual](#)

[NumPy User Guide](#)

[index](#)

[next](#)

[previous](#)

Quickstart tutorial

Prerequisites

Before reading this tutorial you should know a bit of Python. If you would like to refresh your memory, take a look at the [Python tutorial](#).

If you wish to work the examples in this tutorial, you must also have some software installed on your computer. Please see <http://scipy.org/install.html> for instructions.

The Basics

NumPy's main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[ 1., 0., 0.],  
 [ 0., 1., 2.]]
```

NumPy's array class is called `ndarray`. It is also known by the alias `array`. Note that `numpy.array` is not the same as the Standard Python Library class `array.array`, which only handles one-dimensional arrays and offers less functionality. The more important attributes of an `ndarray` object are:

`ndarray.ndim`

the number of axes (dimensions) of the array. In the Python world, the number of dimensions is referred to as *rank*.

`ndarray.shape`

Table Of Contents

- [Quickstart tutorial](#)
 - [Prerequisites](#)
 - [The Basics](#)
 - [An example](#)
 - [Array Creation](#)
 - [Printing Arrays](#)
 - [Basic Operations](#)
 - [Universal Functions](#)
 - [Indexing, Slicing and Iterating](#)
 - [Shape Manipulation](#)
 - [Changing the shape of an array](#)
 - [Stacking together different arrays](#)
 - [Splitting one array into several smaller ones](#)
 - [Copies and Views](#)
 - [No Copy at All](#)
 - [View or Shallow Copy](#)
 - [Deep Copy](#)
 - [Functions and Methods Overview](#)
 - [Less Basic](#)
 - [Broadcasting rules](#)
 - [Fancy indexing and index tricks](#)
 - [Indexing with Arrays of](#)

```
import numpy as np  
a = np.arange(15).reshape(3, 5)
```

a.shape
a.ndim
a.dtype.name

```
import numpy as np  
a = np.arange(15).reshape(3, 5)  
a  
  
array([[ 0,  1,  2,  3,  4],  
       [ 5,  6,  7,  8,  9],  
       [10, 11, 12, 13, 14]])
```

```
print(a.shape)
```

```
(3, 5)
```

```
a.ndim
```

```
2
```

```
a.dtype.name
```

```
'int64'
```

Matrix

m-by-*n* matrix

$a_{i,j}$ *n* columns *j* changes

m
rows

i
changes

$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	\dots
$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	\dots
$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	\dots
\vdots	\vdots	\vdots	\ddots

NumPy ndarray: Multidimensional Array Object

NumPy ndarray

One-dimensional Array (1-D Array)

0	1		n-1
1	2	3	4

Two-dimensional Array (2-D Array)

	0	1		n-1
0	1	2	3	4
1	6	7	8	9
	11	12	13	14
m-1	16	17	18	19
				20

```
import numpy as np  
a = np.array([1,2,3,4,5])
```

One-dimensional Array (1-D Array)

0	1			n-1
1	2	3	4	5

```
a = np.array([1,2,3,4,5])  
a
```

```
array([1, 2, 3, 4, 5])
```

```
a = np.array( [ [1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15],[16,17,18,19,20] ] )
```

Two-dimensional Array (2-D Array)

	0	1		n-1
0	1	2	3	4
1	6	7	8	9
	11	12	13	14
m-1	16	17	18	19
				20

```
a = np.array([[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15],[16,17,18,19,20]])  
a  
  
array([[ 1,  2,  3,  4,  5],  
       [ 6,  7,  8,  9, 10],  
       [11, 12, 13, 14, 15],  
       [16, 17, 18, 19, 20]])
```

```
import numpy as np  
a = np.array([[0, 1, 2, 3],  
[10, 11, 12, 13],  
[20, 21, 22, 23]])  
a
```

0	1	2	3
10	11	12	13
20	21	22	23

```
a = np.array([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])
```

```
a = np.array([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])  
a
```

```
array([[ 0,  1,  2,  3],  
       [10, 11, 12, 13],  
       [20, 21, 22, 23]])
```

```
print(a.ndim)
```

```
2
```

```
print(a.shape)
```

```
(3, 4)
```

0	1	2	3
10	11	12	13
20	21	22	23

NumPy Basics: Arrays and Vectorized Computation

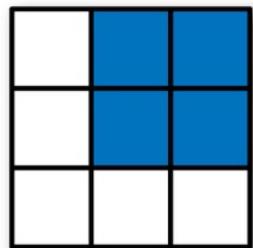
NumPy Array

		axis 1			
		0	1	2	
		0	0, 0	0, 1	0, 2
axis 0		1	1, 0	1, 1	1, 2
		2	2, 0	2, 1	2, 2

Numpy Array

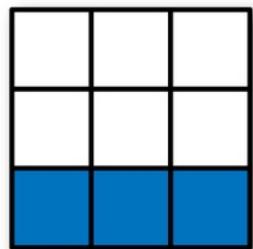
Expression

Shape



arr[:2, 1:]

(2, 2)



arr[2]

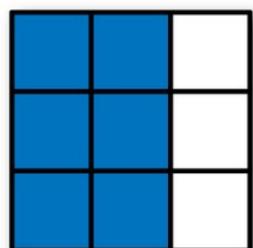
(3,)

arr[2:, :]

(3,)

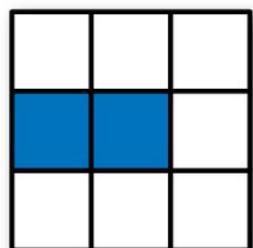
arr[2:, :]

(1, 3)



arr[:, :2]

(3, 2)



arr[1, :2]

(2,)

arr[1:2, :2]

(1, 2)

Tensor

- 3
 - a rank 0 tensor; this is a **scalar** with shape []
- [1., 2., 3.]
 - a rank 1 tensor; this is a **vector** with shape [3]
- [[1., 2., 3.], [4., 5., 6.]]
 - a rank 2 tensor; a **matrix** with shape [2, 3]
- [[[1., 2., 3.]], [[7., 8., 9.]]]
 - a rank 3 **tensor** with shape [2, 1, 3]

Scalar

80

Vector

[50 60 70]

Matrix

$$\begin{bmatrix} 50 & 60 & 70 \\ 55 & 65 & 75 \end{bmatrix}$$

Tensor

$$\begin{bmatrix} [50 & 60 & 70] & [70 & 80 & 90] \\ [55 & 65 & 75] & [75 & 85 & 95] \end{bmatrix}$$

Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media.

Materials and IPython notebooks for "Python for Data Analysis" by Wes McKinney, published by O'Reilly Media

52 commits	2 branches	0 releases	6 contributors
Branch: 2nd-edition	New pull request	Find file	Clone or download
betatim committed with wesm Add requirements (#71)			Latest commit ea47998 5 days ago
 datasets Add Kaggle titanic dataset			5 months ago
 examples Remove sex column from tips dataset			4 months ago
 .gitignore Add gitignore			2 years ago
 COPYING Use MIT license for code examples			a month ago
 README.md Add launch in Azure Notebooks button (#70)			19 days ago
 appa.ipynb Make more cells markdown instead of raw			a month ago
 ch02.ipynb Make more cells markdown instead of raw			a month ago
 ch03.ipynb Make more cells markdown instead of raw			a month ago
 ch04.ipynb Convert all notebooks to v4 format			a month ago
 ch05.ipynb Make more cells markdown instead of raw			a month ago
 ch06.ipynb Make more cells markdown instead of raw			a month ago
 ch07.ipynb Convert all notebooks to v4 format			a month ago
 ch08.ipynb Make more cells markdown instead of raw			a month ago
 ch09.ipynb Make more cells markdown instead of raw			a month ago
 ch10.ipynb Make more cells markdown instead of raw			a month ago

<https://github.com/wesm/pydata-book>

Wes McKinney (2017), "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", 2nd Edition, O'Reilly Media.

wesm / pydata-book

Code Issues Pull requests Projects Insights

Branch: 2nd-edition pydata-book / ch04.ipynb

wesm Convert all notebooks to v4 format c2780a0 on Sep 27

2 contributors

1857 lines (1856 sloc) | 32.6 KB

Raw Blame History

NumPy Basics: Arrays and

```
In [ ]: import numpy as np
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
np.set_printoptions(precision=4, suppress=True)
```

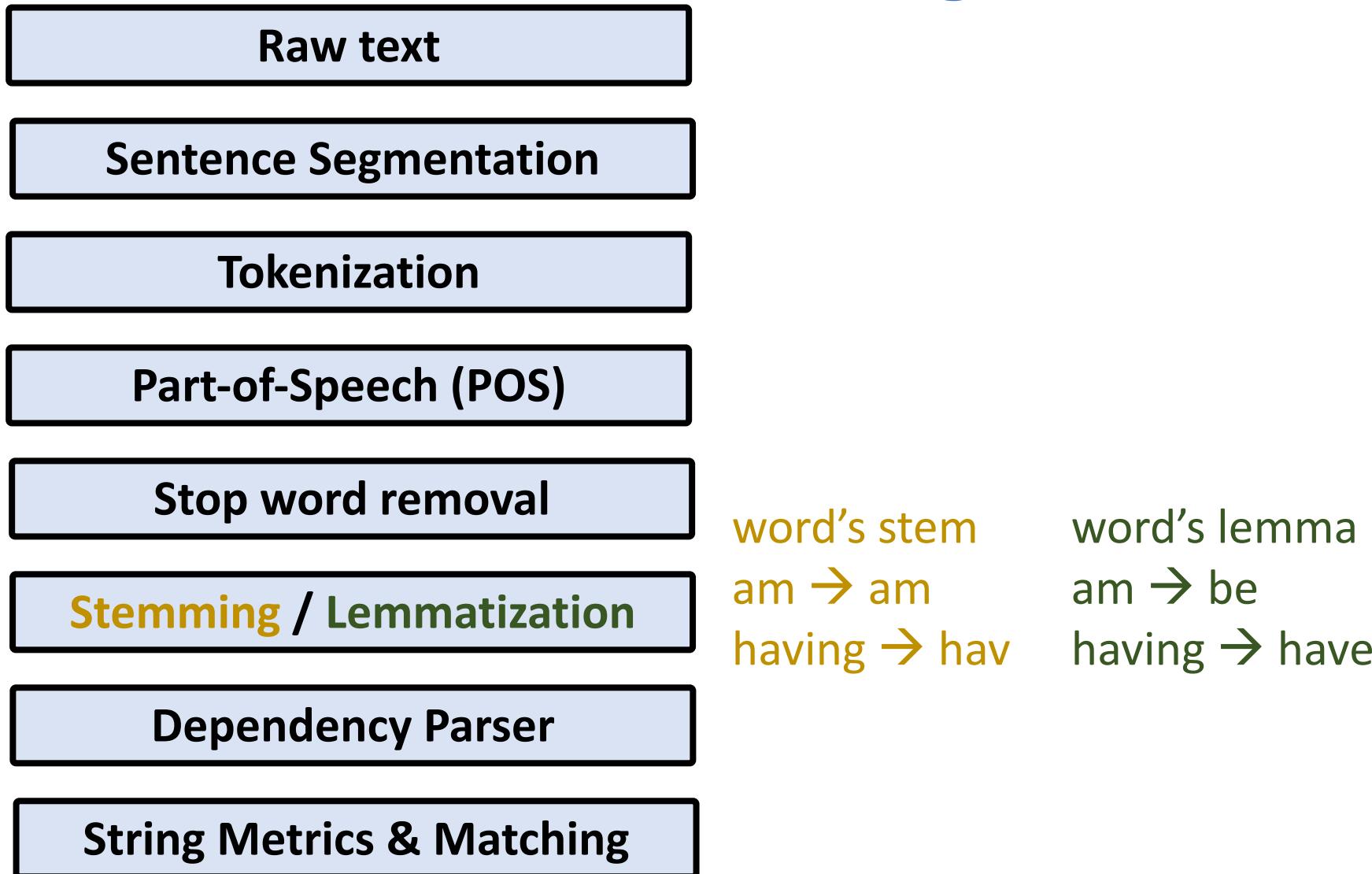
```
In [ ]: import numpy as np
my_arr = np.arange(1000000)
my_list = list(range(1000000))
```

```
In [ ]: %time for _ in range(10): my_arr2 = my_arr * 2
%time for _ in range(10): my_list2 = [x * 2 for x in my_list]
```

The NumPy ndarray: A Multidimensional Array Object

```
In [ ]: import numpy as np
# Generate some random data
data = np.random.randn(2, 3)
data
```

Natural Language Processing (NLP) and Text Mining



spaCy: Natural Language Processing

spaCy

USAGE

MODELS

API

UNIVERSE



Search docs

Industrial-Strength Natural Language Processing

IN PYTHON

Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive. We like to think of spaCy as the Ruby on Rails of Natural Language Processing.

Blazing fast

spaCy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. Independent research in 2015 found spaCy to be the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

Deep learning

spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with TensorFlow, PyTorch, scikit-learn, Gensim and the rest of Python's awesome AI ecosystem. With spaCy, you can easily construct linguistically sophisticated statistical models for a variety of NLP problems.

<https://spacy.io/>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook interface. The left sidebar contains a 'Table of contents' with various sections related to Text Analytics and Natural Language Processing (NLP). The main workspace displays Python code for entity recognition using spaCy and a resulting table of tokens and their POS tags.

Code Snippet:

```
1 text = "Steve Jobs and Steve Wozniak incorporated Apple Computer on January 3, 1977, in Cupertino, California."
2 doc = nlp(text)
3 displacy.render(doc, style="ent", jupyter=True)
```

Entity Recognition Output:

Steve Jobs PERSON and Steve Wozniak PERSON incorporated Apple Computer ORG on January 3, 1977 DATE , in Cupertino GPE , California GPE .

Code Snippet:

```
[ ] 1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Stanford University is located in California. It is a great university.")
4 import pandas as pd
5 cols = ("text", "lemma", "pos", "tag", "pos_explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, t.tag_, spacy.explain(t.pos_), t.is_stop]
9     rows.append(row)
10 df = pd.DataFrame(rows, columns=cols)
11 df
```

DataFrame Output:

	text	lemma	pos	tag	pos_explain	stopword
0	Stanford	Stanford	PROPN	NNP	proper noun	False
1	University	University	PROPN	NNP	proper noun	False
2	is	be	VERB	VBZ	verb	True
3	located	locate	VERB	VBN	verb	False
4	in	in	ADP	IN	adposition	True
5	California	California	PROPN	NNP	proper noun	False
6	.	.	PUNCT	.	punctuation	False
7	It	-PRON-	PRON	PRP	pronoun	True

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab interface with the following details:

- Title:** python101.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help, All changes saved.
- Table of Contents:** Text Analytics and Natural Language Processing (NLP), Python for Natural Language Processing, spaCy.
- Code Snippets:**
 - Text Analytics and Natural Language Processing (NLP):** spaCy: Industrial-Strength Natural Language Processing in Python, Source: <https://spacy.io/usage/spacy-101>
 - Python for Natural Language Processing:** !python -m spacy download en_core_web_sm
 - spaCy:** import spacy, nlp = spacy.load("en_core_web_sm"), doc = nlp("Apple is looking at buying U.K. startup for \$1 billion"), for token in doc:, print(token.text, token.pos_, token.dep_)
 - Output:** Apple PROPN nsubj, is AUX aux, looking VERB ROOT, at ADP prep, buying VERB pcomp, U.K. PROPN compound, startup NOUN dobj, for ADP prep, \$ SYM quantmod, 1 NUM compound, billion NUM pobj

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook titled "python101.ipynb". The code cell contains Python code to process a sentence using spaCy and pandas. The output cell displays a DataFrame with parts-of-speech (POS) tagging information for each word in the sentence.

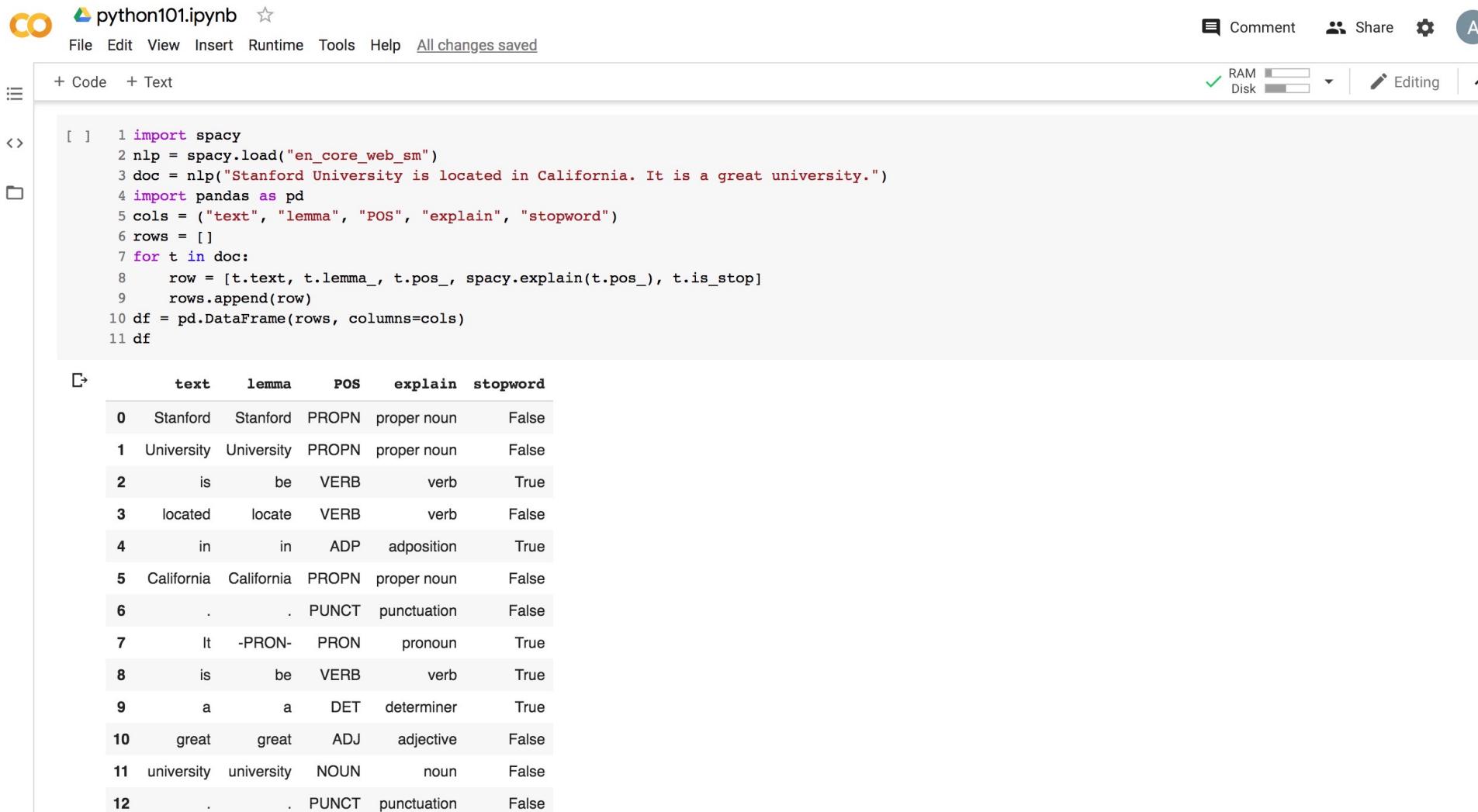
```
[ ] 1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
4 import pandas as pd
5 cols = ("text", "lemma", "POS", "explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, spacy.explain(t.pos_), t.is_stop]
9     rows.append(row)
10 df = pd.DataFrame(rows, columns=cols)
11 df
```

	text	lemma	POS	explain	stopword
0	Apple	Apple	PROPN	proper noun	False
1	is	be	VERB	verb	True
2	looking	look	VERB	verb	False
3	at	at	ADP	adposition	True
4	buying	buy	VERB	verb	False
5	U.K.	U.K.	PROPN	proper noun	False
6	startup	startup	NOUN	noun	False
7	for	for	ADP	adposition	True
8	\$	\$	SYM	symbol	False
9	1	1	NUM	numeral	False
10	billion	billion	NUM	numeral	False

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

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



The screenshot shows a Google Colab notebook titled "python101.ipynb". The code cell contains the following Python script:

```
[ ] 1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Stanford University is located in California. It is a great university.")
4 import pandas as pd
5 cols = ("text", "lemma", "POS", "explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, spacy.explain(t.pos_), t.is_stop]
9     rows.append(row)
10 df = pd.DataFrame(rows, columns=cols)
11 df
```

The output cell displays a DataFrame with the following data:

	text	lemma	POS	explain	stopword
0	Stanford	Stanford	PROPN	proper noun	False
1	University	University	PROPN	proper noun	False
2	is	be	VERB	verb	True
3	located	locate	VERB	verb	False
4	in	in	ADP	adposition	True
5	California	California	PROPN	proper noun	False
6	.	.	PUNCT	punctuation	False
7	It	-PRON-	PRON	pronoun	True
8	is	be	VERB	verb	True
9	a	a	DET	determiner	True
10	great	great	ADJ	adjective	False
11	university	university	NOUN	noun	False
12	.	.	PUNCT	punctuation	False

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot shows the Google Colab interface with two code cells and their corresponding outputs.

Code Cell 1:

```
[ ] 1 import spacy  
2 nlp = spacy.load("en_core_web_sm")  
3 text = "Stanford University is located in California. It is a great university."  
4 doc = nlp(text)  
5 for ent in doc.ents:  
6     print(ent.text, ent.label_)
```

Output 1:

```
Stanford University ORG  
California GPE
```

Code Cell 2:

```
[ ] 1 from spacy import displacy  
2 text = "Stanford University is located in California. It is a great university."  
3 doc = nlp(text)  
4 displacy.render(doc, style="ent", jupyter=True)
```

Output 2:

```
Stanford University ORG is located in California GPE . It is a great university.
```

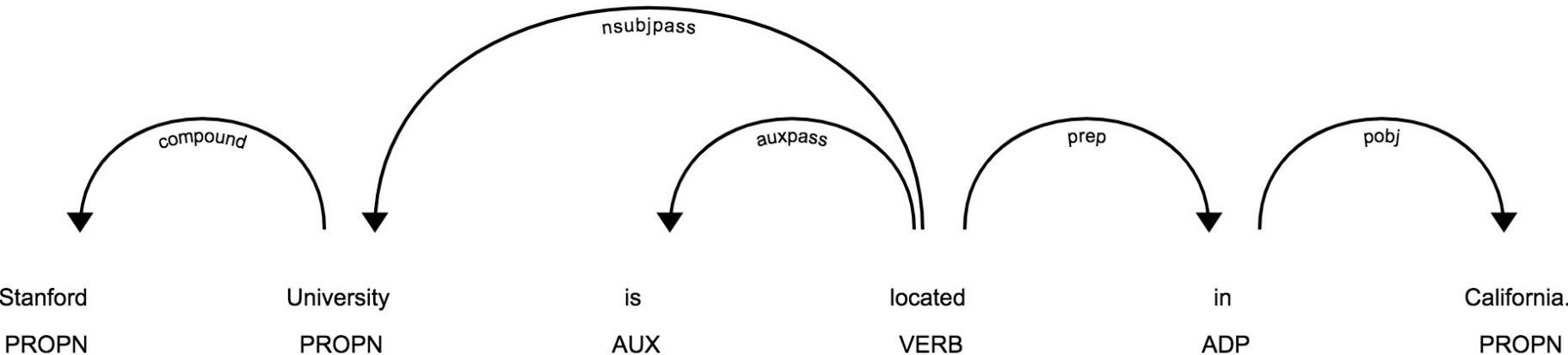
<https://tinyurl.com/aintpython101>

Python in Google Colab (Python101)

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

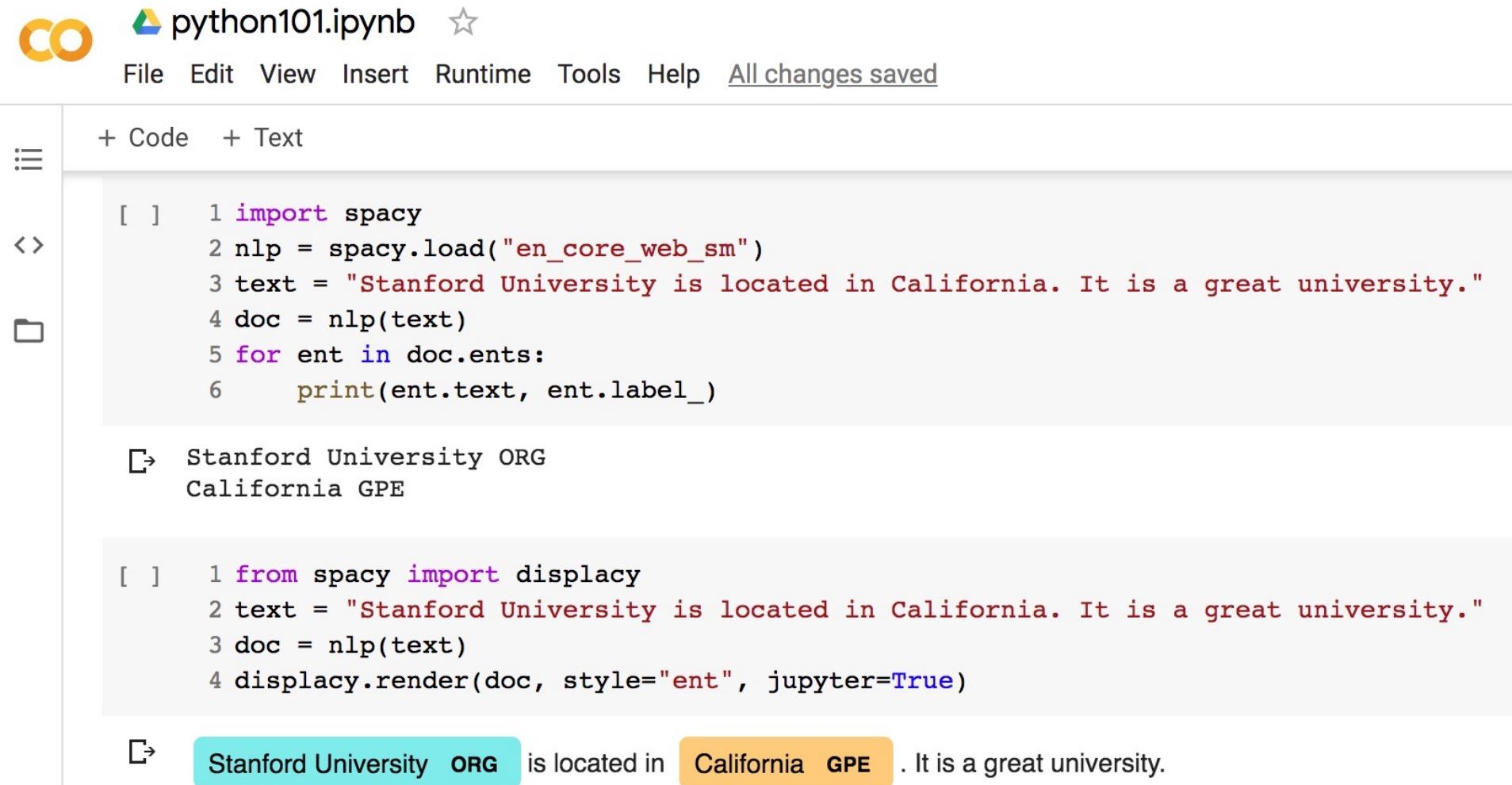
```
1 from spacy import displacy
2 text = "Stanford University is located in California. It is a great university."
3 doc = nlp(text)
4 displacy.render(doc, style="ent", jupyter=True)
5 displacy.render(doc, style="dep", jupyter=True)
```

Stanford University **ORG** is located in **California GPE**. It is a great university.



Python in Google Colab

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



The screenshot shows a Google Colab notebook titled "python101.ipynb". The interface includes a menu bar with File, Edit, View, Insert, Runtime, Tools, Help, and a status bar indicating "All changes saved". On the left, there are icons for file operations like + Code, + Text, and a sidebar with a list icon. The main area contains two code cells. The first cell displays Python code for loading a Spacy model and extracting entities from a text about Stanford University:

```
[ ] 1 import spacy  
2 nlp = spacy.load("en_core_web_sm")  
3 text = "Stanford University is located in California. It is a great university."  
4 doc = nlp(text)  
5 for ent in doc.ents:  
6     print(ent.text, ent.label_)
```

The output of this cell shows the extracted entities:

```
Stanford University ORG  
California GPE
```

The second cell contains code for rendering the entities using displacy:

```
[ ] 1 from spacy import displacy  
2 text = "Stanford University is located in California. It is a great university."  
3 doc = nlp(text)  
4 displacy.render(doc, style="ent", jupyter=True)
```

The output of this cell is a visual representation of the entities:

```
Stanford University ORG is located in California GPE . It is a great university.
```

<https://tinyurl.com/aintpuppython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab interface with the following details:

- Title:** python101.ipynb
- Table of contents:**
 - Text Analytics and Natural Language Processing (NLP)
 - Python for Natural Language Processing**
 - spaCy Chinese Model
 - Open Chinese Convert (OpenCC, 開放中文轉換)
 - Jieba 結巴中文分詞
 - Natural Language Toolkit (NLTK)
 - Stanza: A Python NLP Library for Many Human Languages
 - Text Processing and Understanding
 - NLTK (Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit)
 - NLP Zero to Hero
 - Natural Language Processing - Tokenization (NLP Zero to Hero, part 1)
 - Natural Language Processing - Sequencing - Turning sentence into data (NLP Zero to Hero, part 2)
 - Natural Language Processing - Training a model to recognize sentiment in text (NLP Zero to Hero, part 3)
 - Keras preprocessing text
 - JSON File
- Code and Output:**
 - Entity extraction example:

```
1 text = "Steve Jobs and Steve Wozniak incorporated Apple Computer on January 3, 1977, in Cupertino, California."
2 doc = nlp(text)
3 displacy.render(doc, style="ent", jupyter=True)
```

Output: Steve Jobs PERSON and Steve Wozniak PERSON incorporated Apple Computer ORG on January 3, 1977 DATE , in Cupertino GPE , California GPE .
 - Part-of-speech tagging example:

```
1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Stanford University is located in California. It is a great university.")
4 import pandas as pd
5 cols = ("text", "lemma", "pos", "tag", "pos_explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, t.tag_, spacy.explain(t.pos_), t.is_stop]
9     rows.append(row)
10 df = pd.DataFrame(rows, columns=cols)
11 df
```

Output: A Pandas DataFrame showing POS tags for the sentence "Stanford University is located in California. It is a great university.".

	text	lemma	pos	tag	pos_explain	stopword
0	Stanford	Stanford	PROPN	NNP	proper noun	False
1	University	University	PROPN	NNP	proper noun	False
2	is	be	VERB	VBZ	verb	True
3	located	locate	VERB	VBN	verb	False
4	in	in	ADP	IN	adposition	True
5	California	California	PROPN	NNP	proper noun	False
6	.	.	PUNCT	.	punctuation	False
7	It	-PRON-	PRON	PRP	pronoun	True

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

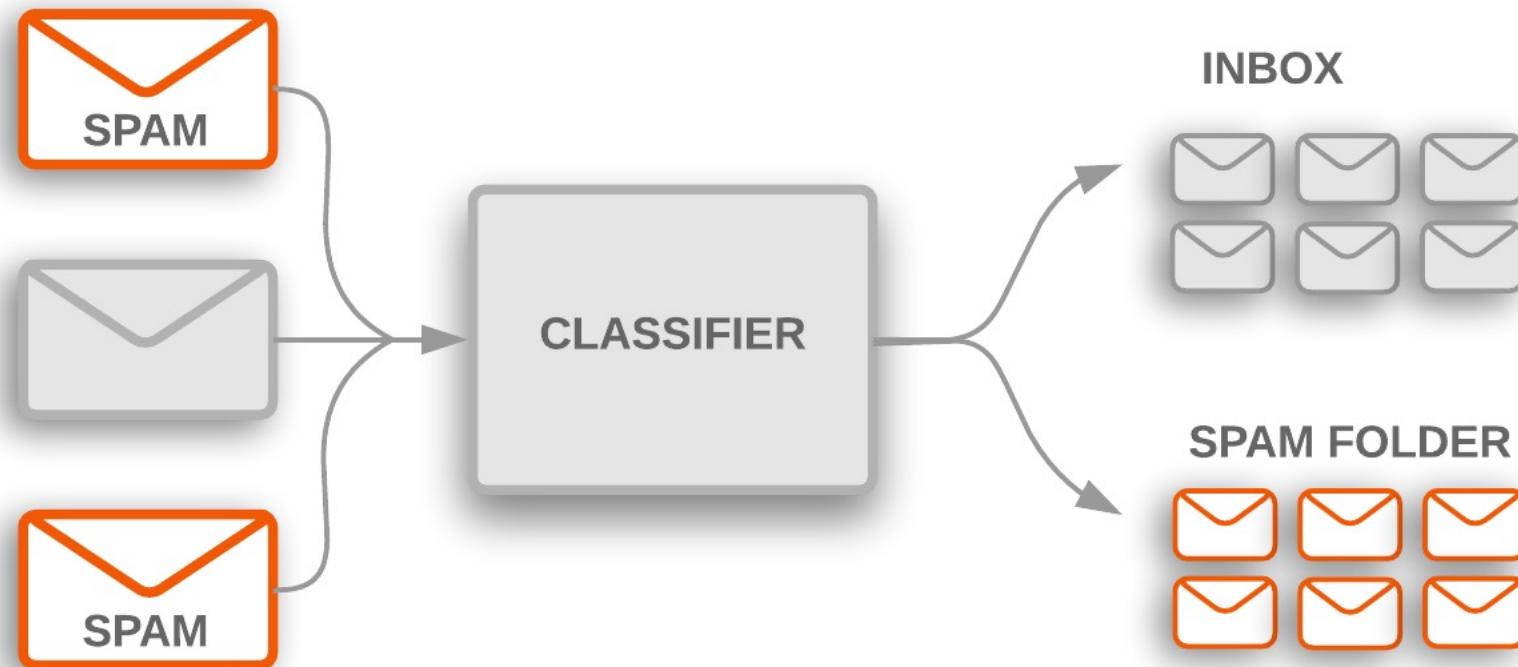
The screenshot shows a Google Colab interface with a Jupyter notebook titled "python101.ipynb". The notebook has a single cell containing Python code for text preprocessing using Keras. The output of the cell shows the results of fitting a Tokenizer to five documents and creating a word matrix.

```
# keras.preprocessing.text Tokenizer
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='count')
print('texts_to_matrix:')
print(texts_to_matrix)
```

```
Using TensorFlow backend.
docs: ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
word_counts: OrderedDict([('well', 1), ('done', 1), ('good', 1), ('work', 2), ('great', 1), ('effort', 1), ('nice', 1), ('excellent', 1)])
document_count: 5
word_index: {'work': 1, 'well': 2, 'done': 3, 'good': 4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1, 'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:
[[0. 0. 1. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1.]]
```

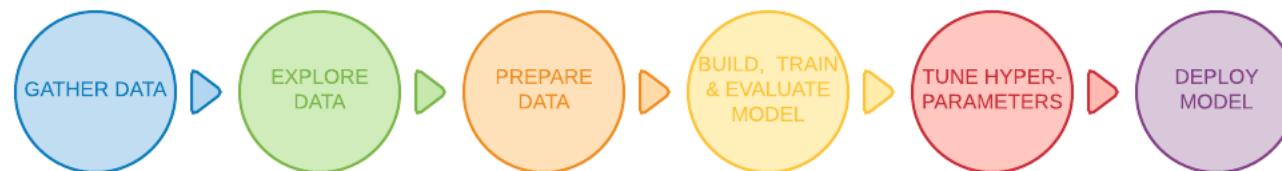
<https://tinyurl.com/aintpuppython101>

Text Classification

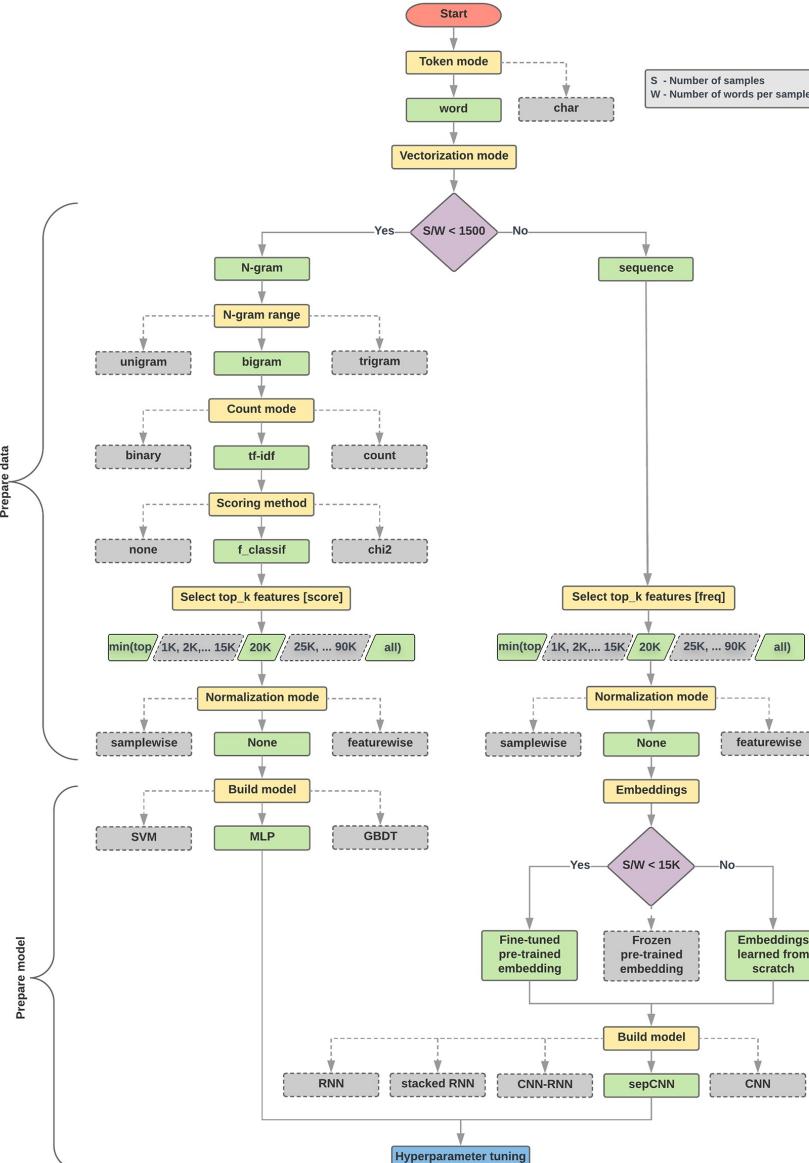


Text Classification Workflow

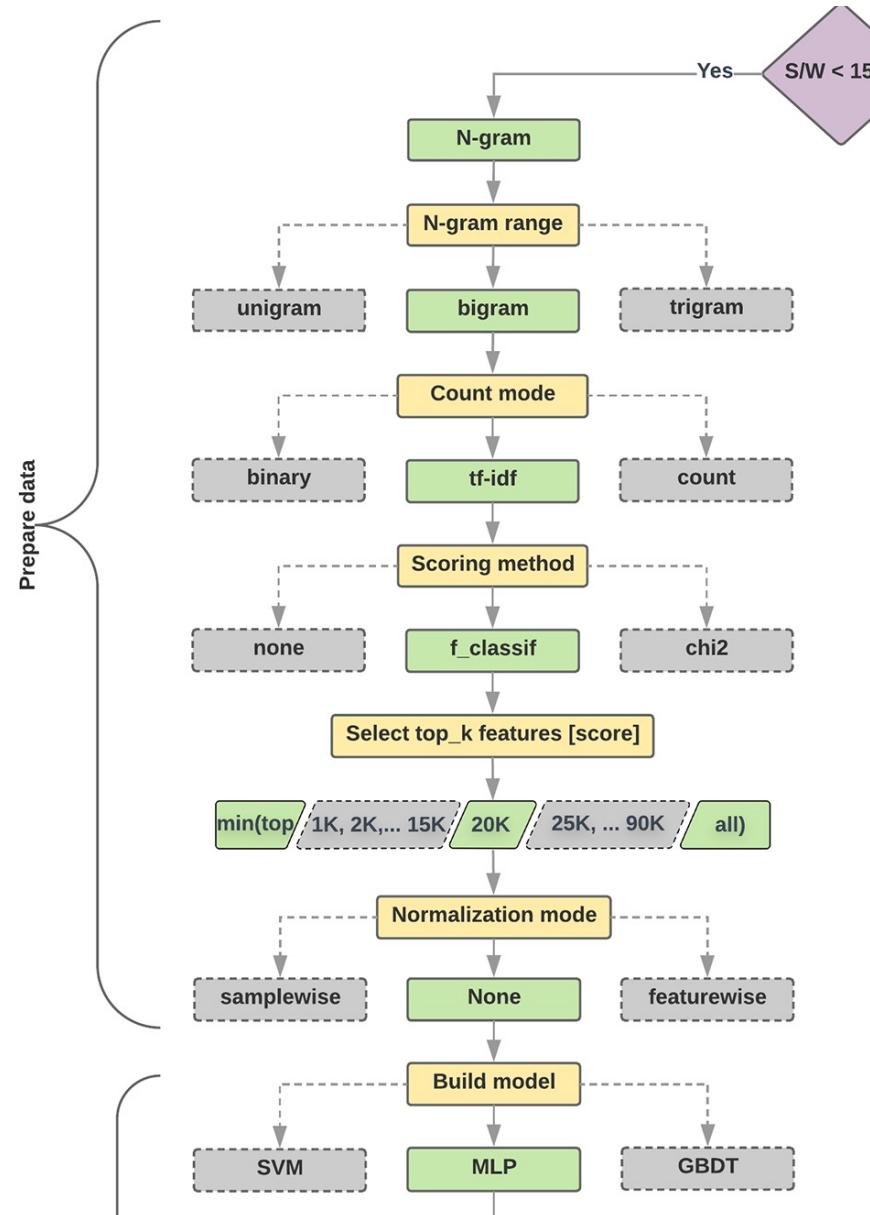
- Step 1: Gather Data
- Step 2: Explore Your Data
- Step 2.5: Choose a Model*
- Step 3: Prepare Your Data
- Step 4: Build, Train, and Evaluate Your Model
- Step 5: Tune Hyperparameters
- Step 6: Deploy Your Model



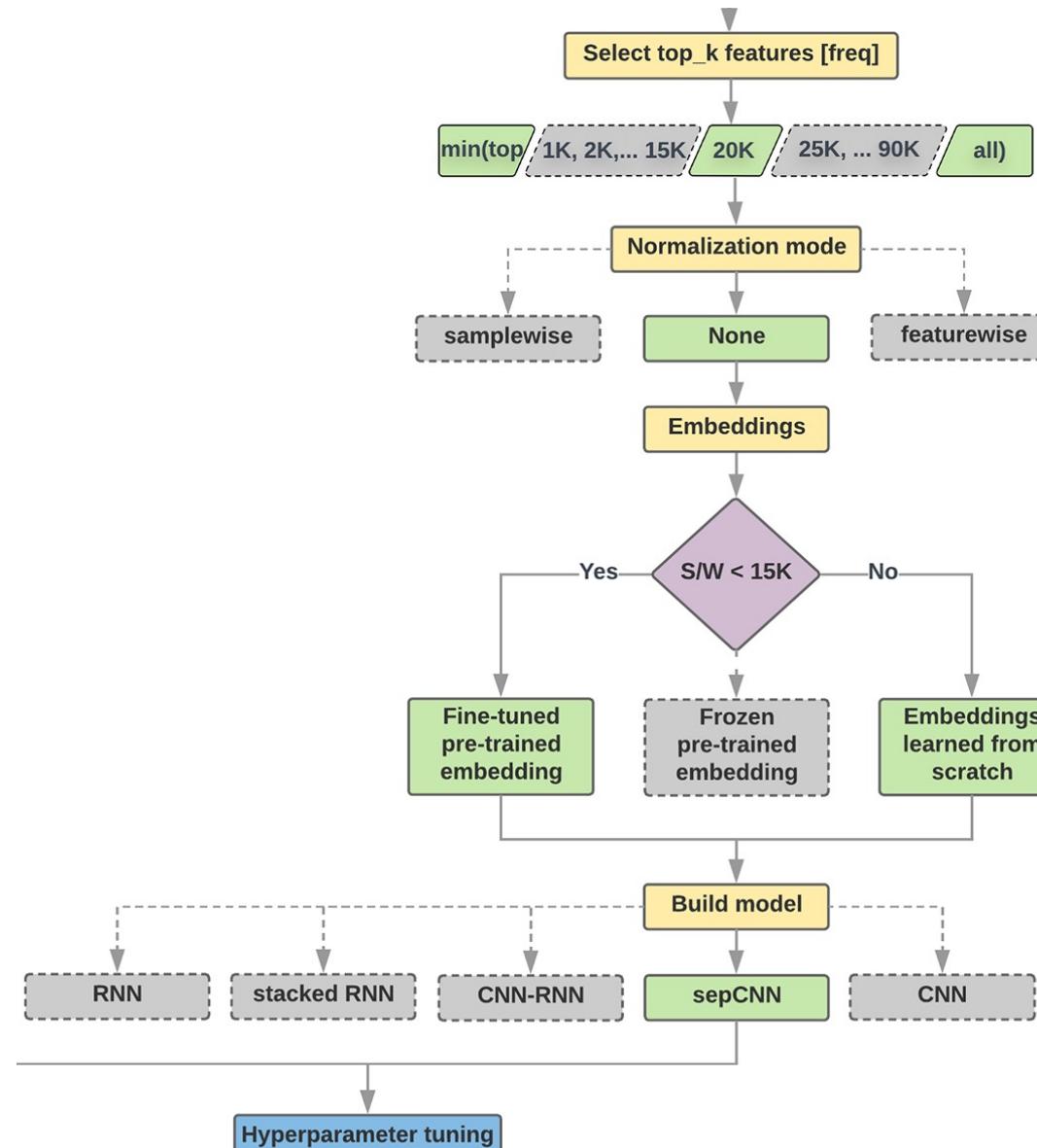
Text Classification Flowchart



Text Classification S/W<1500: N-gram



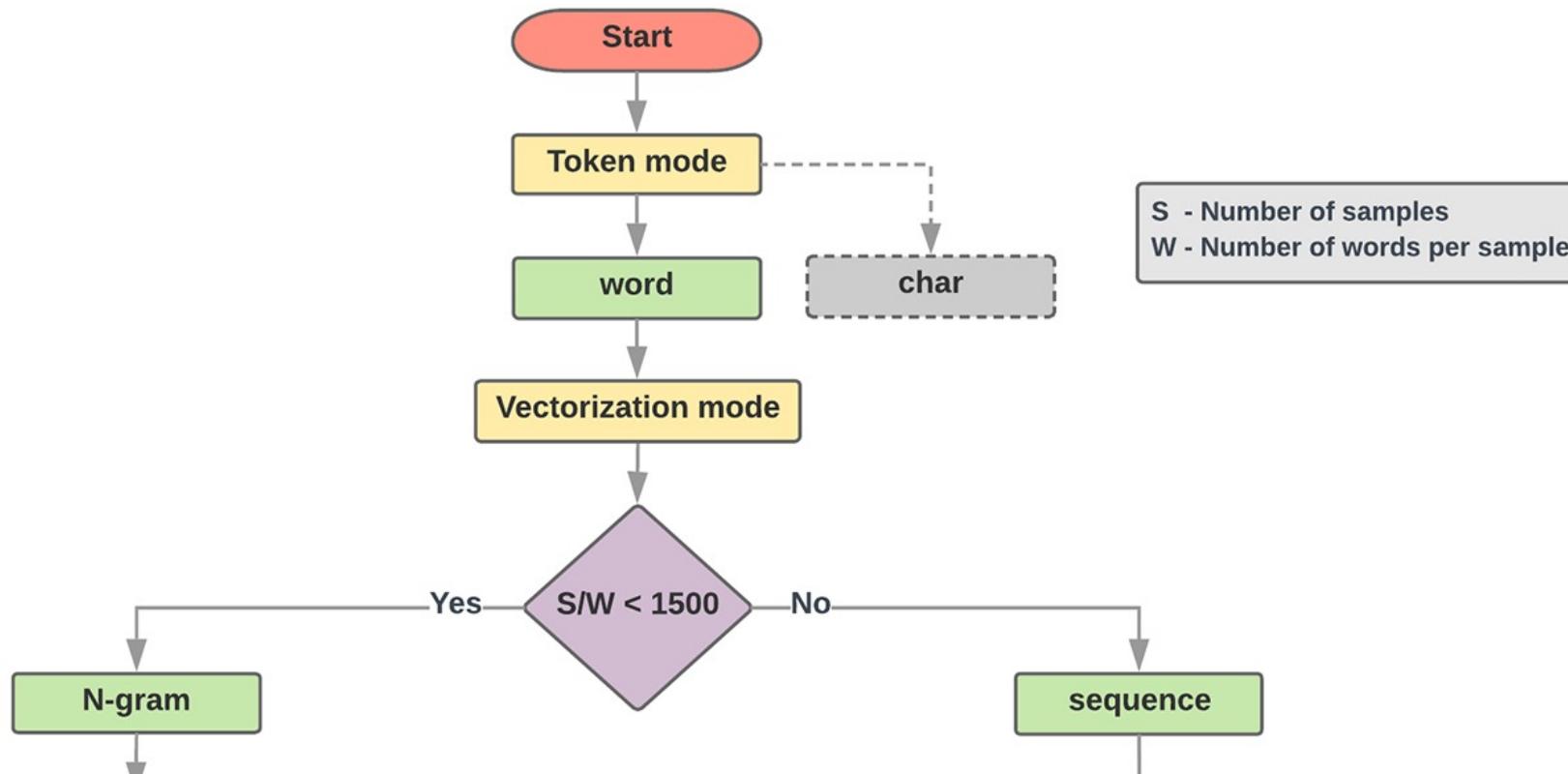
Text Classification S/W>=1500: Sequence



Step 2.5: Choose a Model

Samples/Words < 1500

$$150,000 / 100 = 1500$$

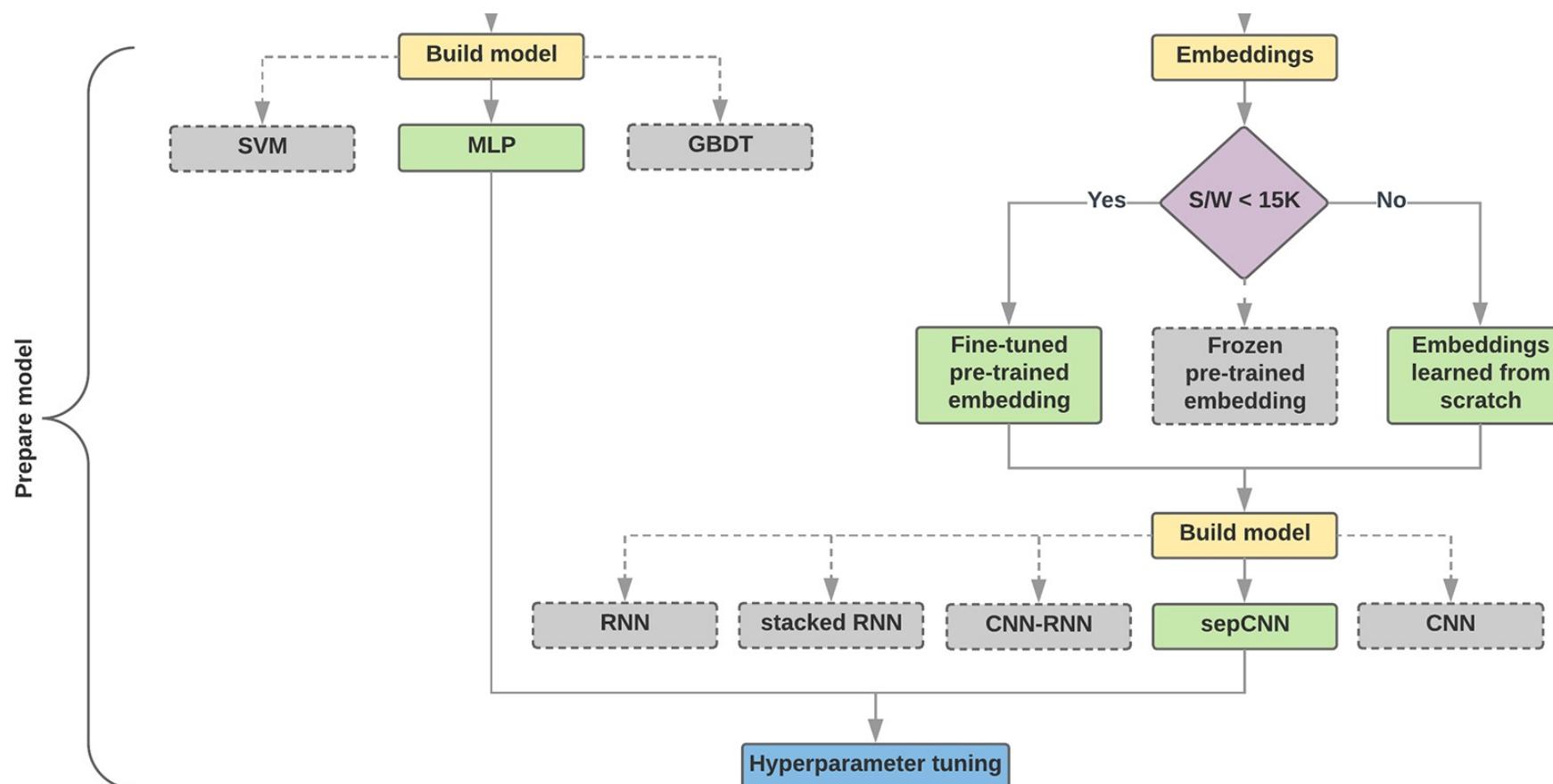


IMDb review dataset,
the samples/words-per-sample ratio is ~ 144

Step 2.5: Choose a Model

Samples/Words < 15,000

$$1,500,000 / 100 = 15,000$$



Step 3: Prepare Your Data

Texts:

T1: 'The mouse ran up the clock'

T2: 'The mouse ran down'

Token Index:

```
{'the': 1, 'mouse': 2, 'ran': 3, 'up': 4, 'clock': 5, 'down': 6,}.
```

NOTE: 'the' occurs most frequently,
so the index value of 1 is assigned to it.
Some libraries reserve index 0 for unknown tokens,
as is the case here.

Sequence of token indexes:

T1: 'The mouse ran up the clock' =

```
[1, 2, 3, 4, 1, 5]
```

T2: 'The mouse ran down' =

```
[1, 2, 3, 6]
```

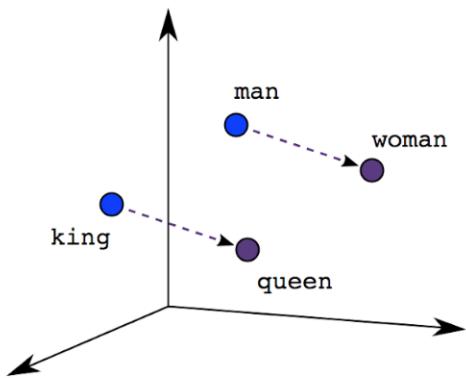
One-hot encoding

'The mouse ran up the clock' =

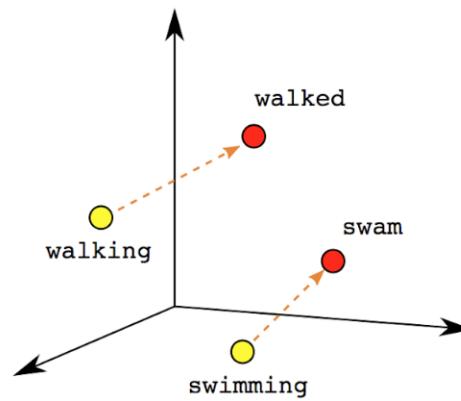
The	1	[[0, 1, 0, 0, 0, 0, 0],
mouse	2	[0, 0, 1, 0, 0, 0, 0],
ran	3	[0, 0, 0, 1, 0, 0, 0],
up	4	[0, 0, 0, 0, 1, 0, 0],
the	1	[0, 1, 0, 0, 0, 0, 0],
clock	5	[0, 0, 0, 0, 0, 1, 0]]

[0, 1, 2, 3, 4, 5, 6]

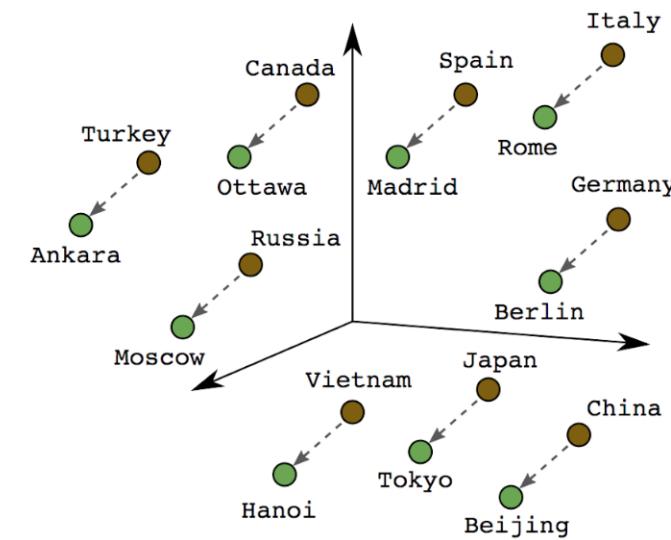
Word embeddings



Male-Female

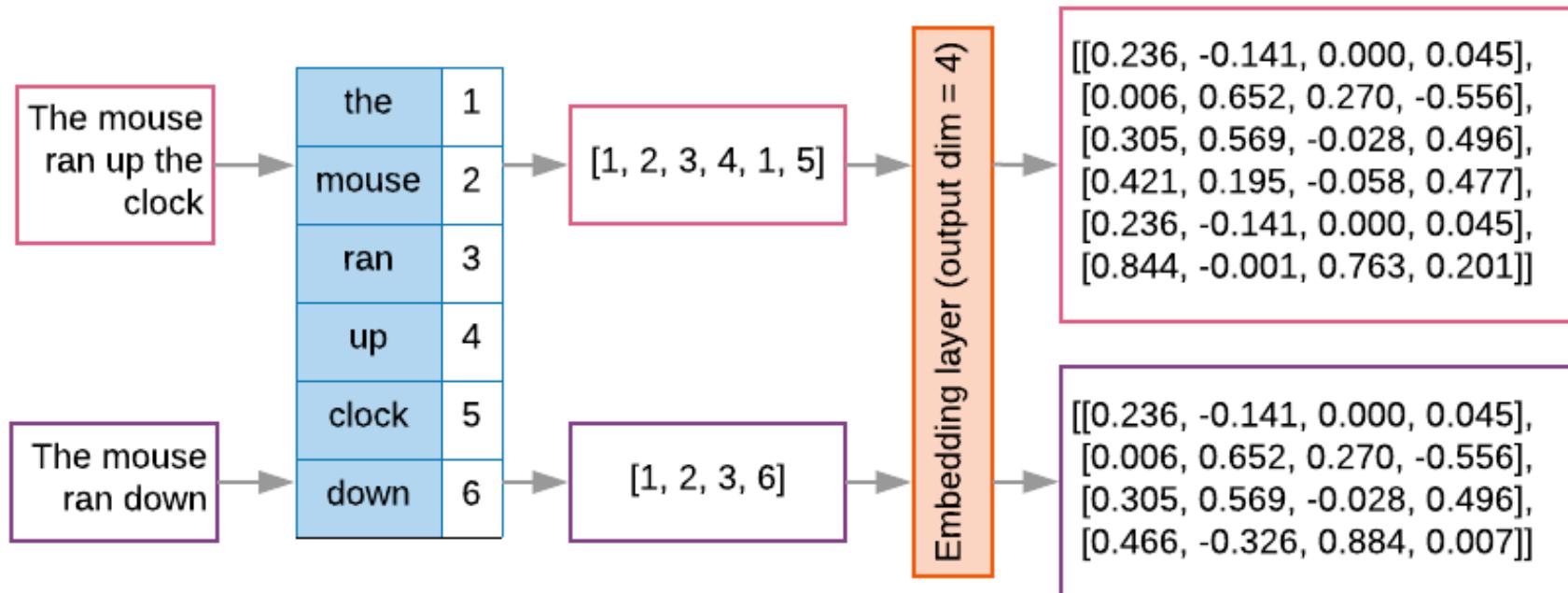


Verb Tense



Country-Capital

Word embeddings



```
t1 = 'The mouse ran up the clock'  
t2 = 'The mouse ran down'  
s1 = t1.lower().split(' ')  
s2 = t2.lower().split(' ')  
terms = s1 + s2  
sortedset = sorted(set(terms))  
print('terms =', terms)  
print('sortedset =', sortedset)
```

```
1 t1 = 'The mouse ran up the clock'  
2 t2 = 'The mouse ran down'  
3 s1 = t1.lower().split(' ')  
4 s2 = t2.lower().split(' ')  
5 terms = s1 + s2  
6 sortedset = sorted(set(terms))  
7 print('terms =', terms)  
8 print('sortedset =', sortedset)
```

```
terms = ['the', 'mouse', 'ran', 'up', 'the', 'clock', 'the', 'mouse', 'ran', 'down']  
sortedset = ['clock', 'down', 'mouse', 'ran', 'the', 'up']
```

```
t1 = 'The mouse ran up the clock'  
t2 = 'The mouse ran down'  
s1 = t1.lower().split(' ')  
s2 = t2.lower().split(' ')  
terms = s1 + s2  
print(terms)  
  
tfdict = {}  
for term in terms:  
    if term not in tfdict:  
        tfdict[term] = 1  
    else:  
        tfdict[term] += 1  
  
a = []  
for k,v in tfdict.items():  
    a.append('{} , {}'.format(k,v))  
print(a)
```

```
['the', 'mouse', 'ran', 'up', 'the', 'clock', 'the', 'mouse', 'ran', 'down']  
['the', 3, 'mouse', 2, 'ran', 2, 'up', 1, 'clock', 1, 'down', 1]
```

```
sorted_by_value_reverse = sorted(tfdict.items(),
key=lambda kv: kv[1], reverse=True)
```

```
sorted_by_value_reverse_dict =
dict(sorted_by_value_reverse)
```

```
id2word = {id: word for id, word in
enumerate(sorted_by_value_reverse_dict)}
```

```
word2id = dict([(v, k) for (k, v) in
id2word.items()])
```

```
sorted_by_value: [('up', 1), ('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3)]
sorted_by_value2: ['the', 'mouse', 'ran', 'up', 'clock', 'down']
sorted_by_value_reverse: [('the', 3), ('mouse', 2), ('ran', 2), ('up', 1), ('clock', 1), ('down', 1)]
sorted_by_value_reverse_dict {'the': 3, 'mouse': 2, 'ran': 2, 'up': 1, 'clock': 1, 'down': 1}
id2word {0: 'the', 1: 'mouse', 2: 'ran', 3: 'up', 4: 'clock', 5: 'down'}
word2id {'the': 0, 'mouse': 1, 'ran': 2, 'up': 3, 'clock': 4, 'down': 5}
len_words: 6
sorted_by_key: [('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3), ('up', 1)]
the, 3
mouse, 2
ran, 2
up, 1
clock, 1
down, 1
```

```

sorted_by_value = sorted(tfdict.items(), key=lambda kv: kv[1])
print('sorted_by_value: ', sorted_by_value)
sorted_by_value2 = sorted(tfdict, key=tfdict.get, reverse=True)
print('sorted_by_value2: ', sorted_by_value2)
sorted_by_value_reverse = sorted(tfdict.items(), key=lambda kv: kv[1], reverse=True)
print('sorted_by_value_reverse: ', sorted_by_value_reverse)
sorted_by_value_reverse_dict = dict(sorted_by_value_reverse)
print('sorted_by_value_reverse_dict', sorted_by_value_reverse_dict)
id2word = {id: word for id, word in enumerate(sorted_by_value_reverse_dict)}
print('id2word', id2word)
word2id = dict([(v, k) for (k, v) in id2word.items()])
print('word2id', word2id)
print('len_words:', len(word2id))

sorted_by_key = sorted(tfdict.items(), key=lambda kv: kv[0])
print('sorted_by_key: ', sorted_by_key)

tfstring = '\n'.join(a)
print(tfstring)
tf = tfdict.get('mouse')
print(tf)

```

```

sorted_by_value:  [('up', 1), ('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3)]
sorted_by_value2:  ['the', 'mouse', 'ran', 'up', 'clock', 'down']
sorted_by_value_reverse:  [('the', 3), ('mouse', 2), ('ran', 2), ('up', 1), ('clock', 1), ('down', 1)]
sorted_by_value_reverse_dict {'the': 3, 'mouse': 2, 'ran': 2, 'up': 1, 'clock': 1, 'down': 1}
id2word {0: 'the', 1: 'mouse', 2: 'ran', 3: 'up', 4: 'clock', 5: 'down'}
word2id {'the': 0, 'mouse': 1, 'ran': 2, 'up': 3, 'clock': 4, 'down': 5}
len_words: 6
sorted_by_key:  [('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3), ('up', 1)]
the, 3
mouse, 2
ran, 2
up, 1
clock, 1
down, 1

```

from keras.preprocessing.text import Tokenizer

```
1 from keras.preprocessing.text import Tokenizer
2 # define 5 documents
3 docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
4 # create the tokenizer
5 t = Tokenizer()
6 # fit the tokenizer on the documents
7 t.fit_on_texts(docs)
8 print('docs:', docs)
9 print('word_counts:', t.word_counts)
10 print('document_count:', t.document_count)
11 print('word_index:', t.word_index)
12 print('word_docs:', t.word_docs)
13 # integer encode documents
14 texts_to_matrix = t.texts_to_matrix(docs, mode='count')
15 print('texts_to_matrix:')
16 print(texts_to_matrix)

docs: ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
word_counts: OrderedDict([('well', 1), ('done', 1), ('good', 1), ('work', 2), ('great', 1), ('effort', 1), ('ni
document_count: 5
word_index: {'work': 1, 'well': 2, 'done': 3, 'good': 4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1, 'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:
[[0. 0. 1. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1.]]
```

from keras.preprocessing.text import Tokenizer

```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice
work', 'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='count')
print('texts_to_matrix:')
print(texts_to_matrix)
```

```
texts_to_matrix =  
t.texts_to_matrix(docs, mode='count')
```

```
docs: ['Well done!', 'Good work', 'Great effort',  
'nice work', 'Excellent!']  
word_counts: OrderedDict([('well', 1), ('done', 1),  
(('good', 1), ('work', 2), ('great', 1), ('effort', 1),  
(('nice', 1), ('excellent', 1))  
document_count: 5  
word_index: {'work': 1, 'well': 2, 'done': 3, 'good':  
4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}  
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1,  
'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}  
texts_to_matrix:  
[[0. 0. 1. 1. 0. 0. 0. 0. 0.]  
[0. 1. 0. 0. 1. 0. 0. 0. 0.]  
[0. 0. 0. 0. 0. 1. 1. 0. 0.]  
[0. 1. 0. 0. 0. 0. 0. 1. 0.]  
[0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

t.texts_to_matrix(docs, mode='tfidf')

```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work',
'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='tfidf')
print('texts_to_matrix:')
print(texts_to_matrix)
```

```
texts_to_matrix:
[[0. 0. 1.25276297 1.25276297 0. 0. 0. 0. 0. 0. ]
 [0. 0.98082925 0. 0. 1.25276297 0. 0. 0. 0. 0. ]
 [0. 0. 0. 0. 0. 1.25276297 1.25276297 0. 0. 0. ]
 [0. 0.98082925 0. 0. 0. 0. 0. 1.25276297 0. 0. ]
 [0. 0. 0. 0. 0. 0. 0. 0. 1.25276297]]
```

NLP Libraries and Tools

spaCy: Natural Language Processing

spaCy

USAGE

MODELS

API

UNIVERSE



Search docs

Industrial-Strength Natural Language Processing

IN PYTHON

Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive. We like to think of spaCy as the Ruby on Rails of Natural Language Processing.

Blazing fast

spaCy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. Independent research in 2015 found spaCy to be the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

Deep learning

spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with TensorFlow, PyTorch, scikit-learn, Gensim and the rest of Python's awesome AI ecosystem. With spaCy, you can easily construct linguistically sophisticated statistical models for a variety of NLP problems.

<https://spacy.io/>

NLTK (Natural Language Toolkit)

NLTK 3.0 documentation

[NEXT](#) | [MODULES](#) | [INDEX](#)

Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to [over 50 corpora and lexical resources](#) such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active [discussion forum](#).

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called “a wonderful tool for teaching, and working in, computational linguistics using Python,” and “an amazing library to play with natural language.”

[Natural Language Processing with Python](#) provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The book is being updated for Python 3 and NLTK 3. (The original Python 2 version is still available at http://nltk.org/book_1ed.)

Some simple things you can do with NLTK

Tokenize and tag some text:

```
>>> import nltk
```

TABLE OF CONTENTS

NLTK News
Installing NLTK
Installing NLTK Data
Contribute to NLTK
FAQ
Wiki
API
HOWTO

SEARCH

 Go
Enter search terms or a module, class or function name.

Natural Language Processing with Python

– Analyzing Text with the Natural Language Toolkit

← → ⌂ www.nltk.org/book/

Natural Language Processing with Python

– Analyzing Text with the Natural Language Toolkit

Steven Bird, Ewan Klein, and Edward Loper

This version of the NLTK book is updated for Python 3 and NLTK 3. The first edition of the book, published by O'Reilly, is available at http://nltk.org/book_1ed/. (There are currently no plans for a second edition of the book.)



- 0. [Preface](#)
- 1. [Language Processing and Python](#)
- 2. [Accessing Text Corpora and Lexical Resources](#)
- 3. [Processing Raw Text](#)
- 4. [Writing Structured Programs](#)
- 5. [Categorizing and Tagging Words](#) (minor fixes still required)
- 6. [Learning to Classify Text](#)
- 7. [Extracting Information from Text](#)
- 8. [Analyzing Sentence Structure](#)
- 9. [Building Feature Based Grammars](#)
- 10. [Analyzing the Meaning of Sentences](#) (minor fixes still required)
- 11. [Managing Linguistic Data](#) (minor fixes still required)
- 12. [Afterword: Facing the Language Challenge](#)

[Bibliography](#)

[Term Index](#)

This book is made available under the terms of the [Creative Commons Attribution Noncommercial No-Derivative-Works 3.0 US License](#). Please post any questions about the materials to the [nltk-users](#) mailing list. Please report any errors on the [issue tracker](#).

<http://www.nltk.org/book/>

gensim

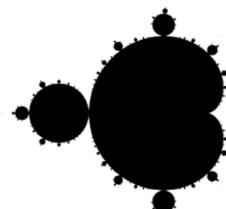
The screenshot shows the official website for the gensim library. At the top left is a GitHub fork button labeled "Fork me on GitHub". To the right is the gensim logo, which is a blue circular icon containing a stylized "g" and the text "SSPIW 2.0". The main title "gensim" is in large blue letters, with the subtitle "topic modelling for humans" in smaller gray text below it. To the right of the title are two green call-to-action buttons: "Download" (with a download arrow icon) and "Direct install with: easy_install -U gensim" (with a lightbulb icon). Below the title is a navigation bar with links for Home, Tutorials, Install, Support, API, and About. The "Home" link is highlighted with a blue background. A large blue box at the bottom contains a Python code snippet demonstrating the library's usage:

```
>>> from gensim import corpora, models, similarities
>>>
>>> # Load corpus iterator from a Matrix Market file on disk.
>>> corpus = corpora.MmCorpus('/path/to/corpus.mm')
>>>
>>> # Initialize Latent Semantic Indexing with 200 dimensions.
>>> lsi = models.LsiModel(corpus, num_topics=200)
>>>
>>> # Convert another corpus to the Latent space and index it.
>>> index = similarities.MatrixSimilarity(lsi[another_corpus])
>>>
>>> # Compute similarity of a query vs. indexed documents
>>> sims = index[query]
```

Gensim is a FREE Python library

- Scalable statistical semantics
- Analyze plain-text documents for semantic structure
- Retrieve semantically similar documents

TextBlob



TextBlob

Star 3,777

TextBlob is a Python (2 and 3) library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more.

Useful Links

[TextBlob @ PyPI](#)

[TextBlob @ GitHub](#)

[Issue Tracker](#)

Stay Informed

Follow @sloria

Donate

If you find TextBlob useful,

TextBlob: Simplified Text Processing

Release v0.12.0. ([Changelog](#))

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

```
from textblob import TextBlob

text = """
The titular threat of The Blob has always struck me as the ultimate movie
monster: an insatiably hungry, amoeba-like mass able to penetrate
virtually any safeguard, capable of--as a doomed doctor chillingly
describes it--"assimilating flesh on contact.
Snide comparisons to gelatin be damned, it's a concept with the most
devastating of potential consequences, not unlike the grey goo scenario
proposed by technological theorists fearful of
artificial intelligence run rampant.
"""

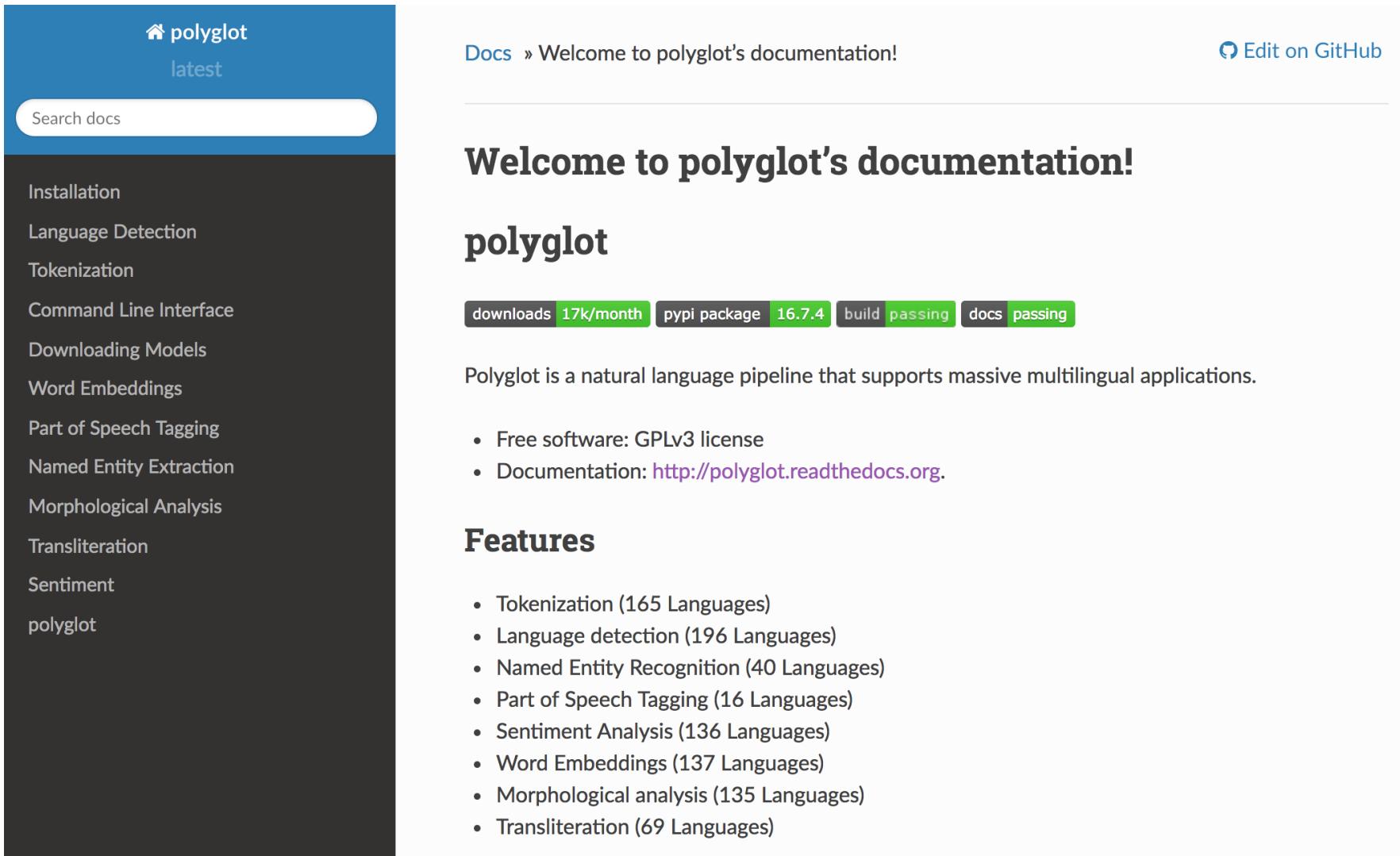
blob = TextBlob(text)
blob.tags          # [('The', 'DT'), ('titular', 'JJ'),
# ('threat', 'NN'), ('of', 'IN'), ...]

blob.noun_phrases # WordList(['titular threat', 'blob',
#                      'ultimate movie monster',
#                      'amoeba-like mass', ...])

for sentence in blob.sentences:
    print(sentence.sentiment.polarity)
# 0.060
```

<https://textblob.readthedocs.io>

Polyglot



The screenshot shows the homepage of the Polyglot documentation. The top navigation bar is blue, featuring the project name "polyglot" and the word "latest". A search bar is located below the navigation. The main content area has a white background. At the top right, there is a link to "Edit on GitHub". The main heading "Welcome to polyglot's documentation!" is displayed in large, bold, dark font. Below it, the word "polyglot" is shown in a larger, bold, dark font. A row of four green rectangular buttons provides metrics: "downloads 17k/month", "pypi package 16.7.4", "build passing", and "docs passing". A descriptive paragraph follows, stating "Polyglot is a natural language pipeline that supports massive multilingual applications." A bulleted list details the software's features: "Free software: GPLv3 license" and "Documentation: <http://polyglot.readthedocs.org>". The sidebar on the left lists various documentation topics: Installation, Language Detection, Tokenization, Command Line Interface, Downloading Models, Word Embeddings, Part of Speech Tagging, Named Entity Extraction, Morphological Analysis, Transliteration, Sentiment, and the "polyglot" module itself.

Docs » Welcome to polyglot's documentation!

[Edit on GitHub](#)

Welcome to polyglot's documentation!

polyglot

downloads 17k/month pypi package 16.7.4 build passing docs passing

Polyglot is a natural language pipeline that supports massive multilingual applications.

- Free software: GPLv3 license
- Documentation: <http://polyglot.readthedocs.org>.

Features

- Tokenization (165 Languages)
- Language detection (196 Languages)
- Named Entity Recognition (40 Languages)
- Part of Speech Tagging (16 Languages)
- Sentiment Analysis (136 Languages)
- Word Embeddings (137 Languages)
- Morphological analysis (135 Languages)
- Transliteration (69 Languages)

scikit-learn



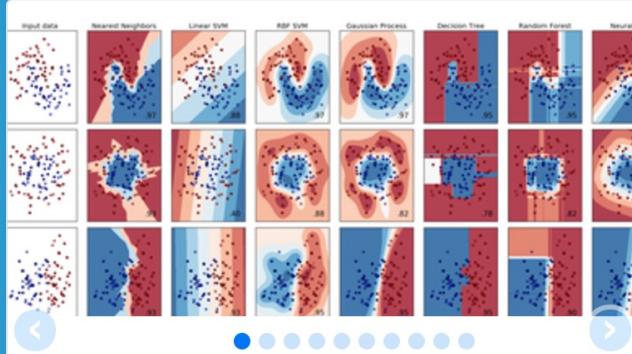
Home Installation Documentation Examples

Google Custom Search

Search

Fork me on GitHub

powered by Google



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.

Algorithms: SVM, nearest neighbors, random forest, ...

— Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, ridge regression, Lasso, ...

— Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes

Algorithms: k-Means, spectral clustering, mean-shift, ...

— Examples

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.

Modules: preprocessing, feature extraction.

<http://scikit-learn.org/>

TensorFlow NLP Examples

- **Basic Text Classification
(Text Classification) (46 Seconds)**

- https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb

- **NMT with Attention
(20-30 minutes)**

- https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/nmt_with_attention/nmt_with_attention.ipynb

Text Classification

IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsLlrLYtPCvCHaoO1W-i_gror

The screenshot shows a Google Colab notebook interface. The title bar says 'tf02_basic-text-classification.ipynb'. The left sidebar contains a 'Table of contents' with sections like 'Copyright 2018 The TensorFlow Authors.', 'Text classification with movie reviews', and 'Evaluate the model'. The main content area has a heading 'Text classification with movie reviews' with sub-sections for 'View on TensorFlow.org', 'Run in Google Colab', and 'View source on GitHub'. It describes the task of classifying movie reviews as positive or negative. Below this is a code cell containing Python code for setting up GPU support and importing libraries.

```
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
    processes = psutil.Process(os.getcwd()).children(recursive=True)
```

NLP with Transformers Github

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search / Sign in Sign up

[nlp-with-transformers / notebooks](#) Public

Notifications Fork 170 Star 1.1k

Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags Go to file Code

lewtn Merge pull request #21 from JingchaoZhang/patch-3 ... ae5b7c1 15 days ago 71 commits

.github/ISSUE_TEMPLATE Update issue templates 25 days ago

data Move dataset to data directory 4 months ago

images Add README last month

scripts Update issue templates 25 days ago

.gitignore Initial commit 4 months ago

01_introduction.ipynb Remove Colab badges & fastdoc refs 27 days ago

02_classification.ipynb Merge pull request #8 from nlp-with-transformers/remove-display-df 26 days ago

03_transformer-anatomy.ipynb [Transformers Anatomy] Remove cells with figure references 22 days ago

04_multilingual-ner.ipynb Merge pull request #8 from nlp-with-transformers/remove-display-df 26 days ago

05_text-generation.ipynb Merge pull request #8 from nlp-with-transformers/remove-display-df 26 days ago

About

Jupyter notebooks for the Natural Language Processing with Transformers book

[transformersbook.com/](#)

Readme Apache-2.0 License 1.1k stars 33 watching 170 forks

O'REILLY® Natural Language Processing with Transformers Building Language Applications with Hugging Face

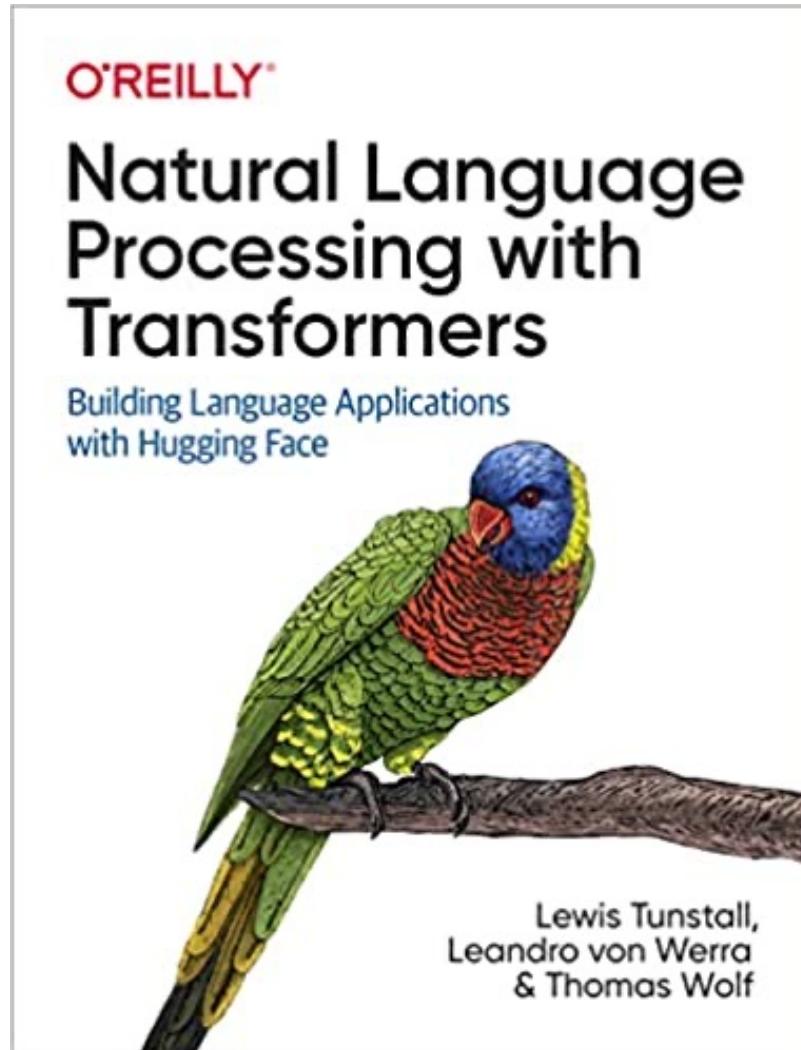
Releases

No releases published

Packages

<https://github.com/nlp-with-transformers/notebooks>

NLP with Transformers Github Notebooks



Running on a cloud platform

To run these notebooks on a cloud platform, just click on one of the badges in the table below:

Chapter	Colab	Kaggle	Gradient	Studio Lab
Introduction	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Classification	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Transformer Anatomy	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Multilingual Named Entity Recognition	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Text Generation	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Summarization	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Question Answering	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Making Transformers Efficient in Production	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Dealing with Few to No Labels	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Training Transformers from Scratch	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab
Future Directions	Open in Colab	Open in Kaggle	Run on Gradient	Open Studio Lab

Nowadays, the GPUs on Colab tend to be K80s (which have limited memory), so we recommend using [Kaggle](#), [Gradient](#), or [SageMaker Studio Lab](#). These platforms tend to provide more performant GPUs like P100s, all for free!

<https://github.com/nlp-with-transformers/notebooks>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab interface with the following details:

- Title:** python101.ipynb
- Table of contents:**
 - Text Analytics and Natural Language Processing (NLP)
 - Python for Natural Language Processing**
 - spaCy Chinese Model
 - Open Chinese Convert (OpenCC, 開放中文轉換)
 - Jieba 結巴中文分詞
 - Natural Language Toolkit (NLTK)
 - Stanza: A Python NLP Library for Many Human Languages
 - Text Processing and Understanding
 - NLTK (Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit)
 - NLP Zero to Hero
 - Natural Language Processing - Tokenization (NLP Zero to Hero, part 1)
 - Natural Language Processing - Sequencing - Turning sentence into data (NLP Zero to Hero, part 2)
 - Natural Language Processing - Training a model to recognize sentiment in text (NLP Zero to Hero, part 3)
 - Keras preprocessing text
 - JSON File
- Code and Output:**
 - Code cell 1:

```
1 text = "Steve Jobs and Steve Wozniak incorporated Apple Computer on January 3, 1977, in Cupertino, California."
2 doc = nlp(text)
3 displacy.render(doc, style="ent", jupyter=True)
```

Output:

Steve Jobs PERSON and Steve Wozniak PERSON incorporated Apple Computer ORG on January 3, 1977 DATE , in Cupertino GPE , California GPE .
 - Code cell 2:

```
1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Stanford University is located in California. It is a great university.")
4 import pandas as pd
5 cols = ("text", "lemma", "pos", "tag", "pos_explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, t.tag_, spacy.explain(t.pos_), t.is_stop]
9     rows.append(row)
10 df = pd.DataFrame(rows, columns=cols)
11 df
```

Output:

	text	lemma	pos	tag	pos_explain	stopword
0	Stanford	Stanford	PROPN	NNP	proper noun	False
1	University	University	PROPN	NNP	proper noun	False
2	is	be	VERB	VBZ	verb	True
3	located	locate	VERB	VBN	verb	False
4	in	in	ADP	IN	adposition	True
5	California	California	PROPN	NNP	proper noun	False
6	.	.	PUNCT	.	punctuation	False
7	It	-PRON-	PRON	PRP	pronoun	True

<https://tinyurl.com/aintpupython101>

Summary

- Python for Natural Language Processing
 - Python Ecosystem for Data Science
 - Python
 - Programming language
 - Numpy
 - Scientific computing
 - SpaCy
 - Natural Language Processing

References

- Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Denis Rothman (2021), Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more, Packt Publishing.
- Savaş Yıldırım and Meysam Asgari-Chenaghlu (2021), Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques, Packt Publishing.
- Sowmya Vajjala, Bodhisattwa Majumder, Anuj Gupta (2020), Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems, O'Reilly Media.
- Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson.
- Dipanjan Sarkar (2019), Text Analytics with Python: A Practitioner's Guide to Natural Language Processing, Second Edition. APress.
- Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018), Applied Text Analysis with Python:
Enabling Language-Aware Data Products with Machine Learning, O'Reilly.
- Charu C. Aggarwal (2018), Machine Learning for Text, Springer.
- Gabe Ignatow and Rada F. Mihalcea (2017), An Introduction to Text Mining: Research Design, Data Collection, and Analysis, SAGE Publications.
- Rajesh Arumugam (2018), Hands-On Natural Language Processing with Python: A practical guide to applying deep learning architectures to your NLP applications, Packt.
- Jake VanderPlas (2016), Python Data Science Handbook: Essential Tools for Working with Data, O'Reilly Media.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805.
- Steven Bird, Ewan Klein and Edward Loper (2009), Natural Language Processing with Python, O'Reilly Media, <http://www.nltk.org/book/>
- The Super Duper NLP Repo, <https://notebooks.quantumstat.com/>
- Avinash Jain (2017), Introduction To Python Programming, Udemy, <https://www.udemy.com/pythonforbeginnersintro/>
- 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/>
- W3Schools Python, <https://www.w3schools.com/python/>
- Learn Python, <https://www.learnpython.org/>
- Google's Python Class, <https://developers.google.com/edu/python>
- Min-Yuh Day (2023), Python 101, <https://tinyurl.com/aintpupython101>