

Artificial Intelligence for Text Analytics

Python for Natural Language Processing

1102AITA03

MBA, IM, NTPU (M5026) (Spring 2022)

Tue 2, 3, 4 (9:10-12:00) (B8F40)



<https://meet.google.com/paj-zhji-mya>



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	2022/02/22	Introduction to Artificial Intelligence for Text Analytics
2	2022/03/01	Foundations of Text Analytics: Natural Language Processing (NLP)
3	2022/03/08	Python for Natural Language Processing
4	2022/03/15	Natural Language Processing with Transformers
5	2022/03/22	Case Study on Artificial Intelligence for Text Analytics I
6	2022/03/29	Text Classification and Sentiment Analysis

Syllabus

Week	Date	Subject/Topics
7	2022/04/05	Tomb-Sweeping Day (Holiday, No Classes)
8	2022/04/12	Midterm Project Report
9	2022/04/19	Multilingual Named Entity Recognition (NER), Text Similarity and Clustering
10	2022/04/26	Text Summarization and Topic Models
11	2022/05/03	Text Generation
12	2022/05/10	Case Study on Artificial Intelligence for Text Analytics II

Syllabus

Week Date Subject/Topics

13 2022/05/17 Question Answering and Dialogue Systems

**14 2022/05/24 Deep Learning, Transfer Learning,
Zero-Shot, and Few-Shot Learning for Text Analytics**

15 2022/05/31 Final Project Report I

16 2022/06/07 Final Project Report II

17 2022/06/14 Self-learning

18 2022/06/21 Self-learning

Python for Natural Language Processing

Outline

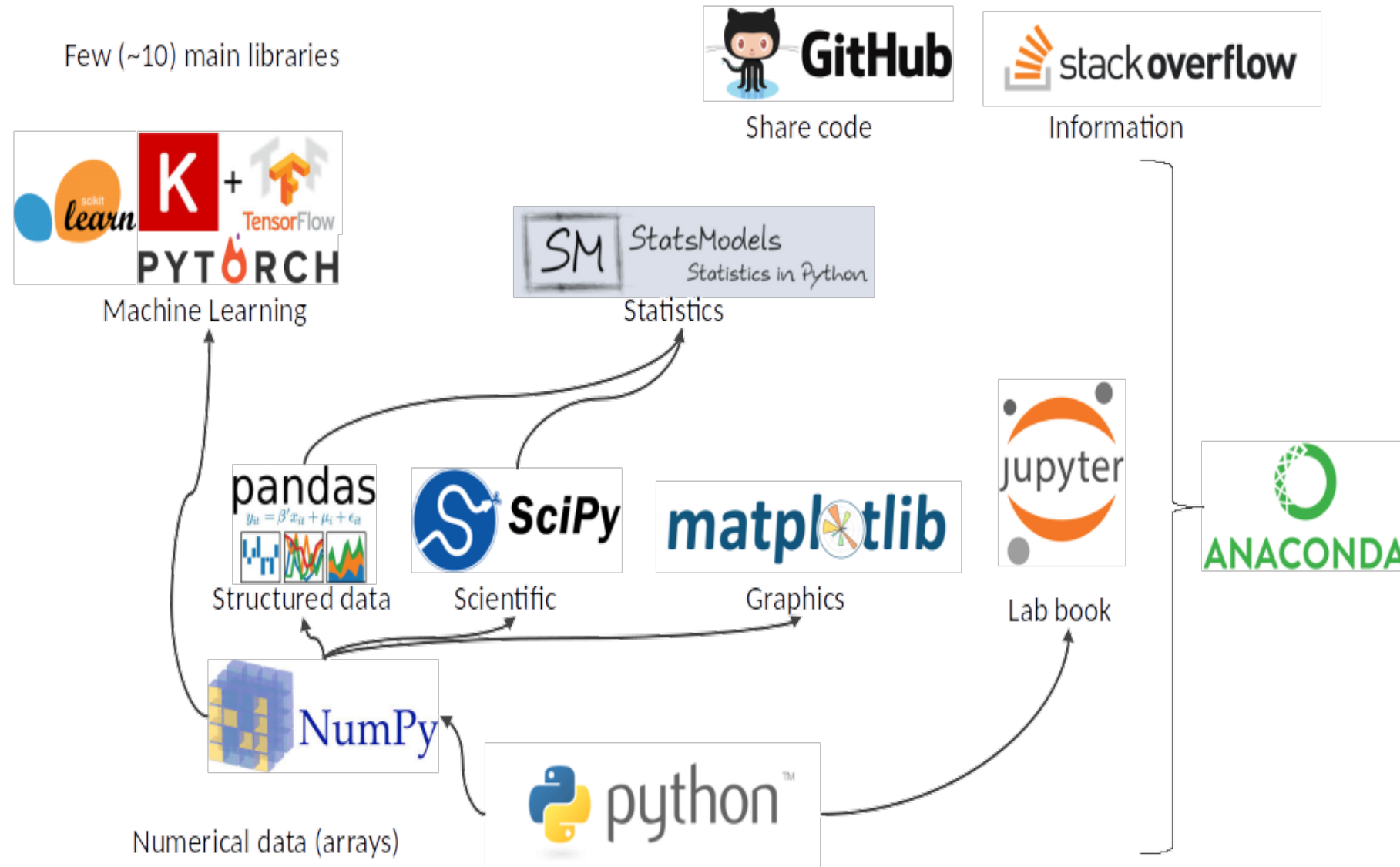
- **Python for Natural Language Processing**
 - **Python Ecosystem for Data Science**
 - **Python**
 - Programming language
 - **Numpy**
 - Scientific computing
 - **SpaCy**
 - Natural Language Processing



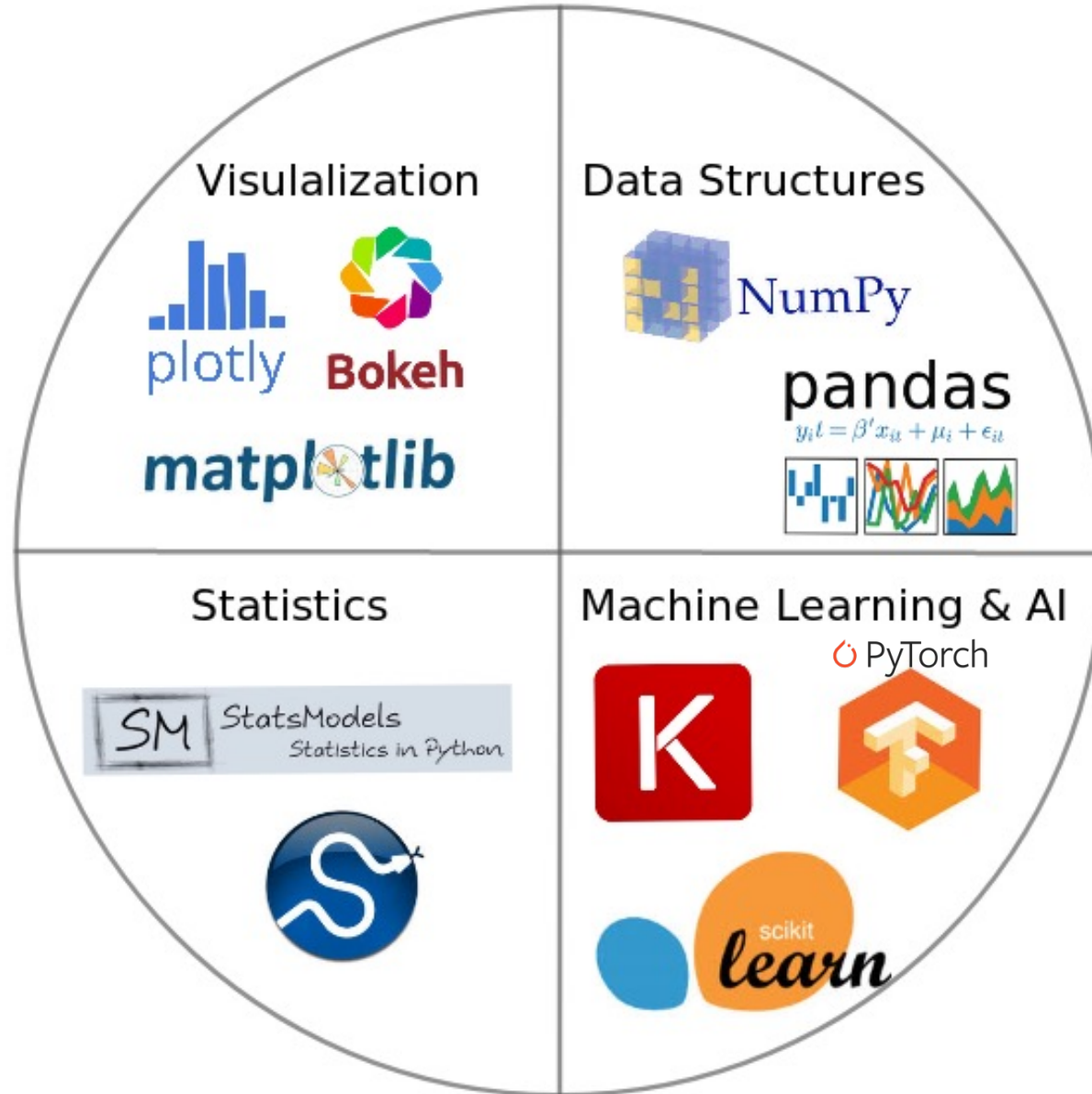
Python

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

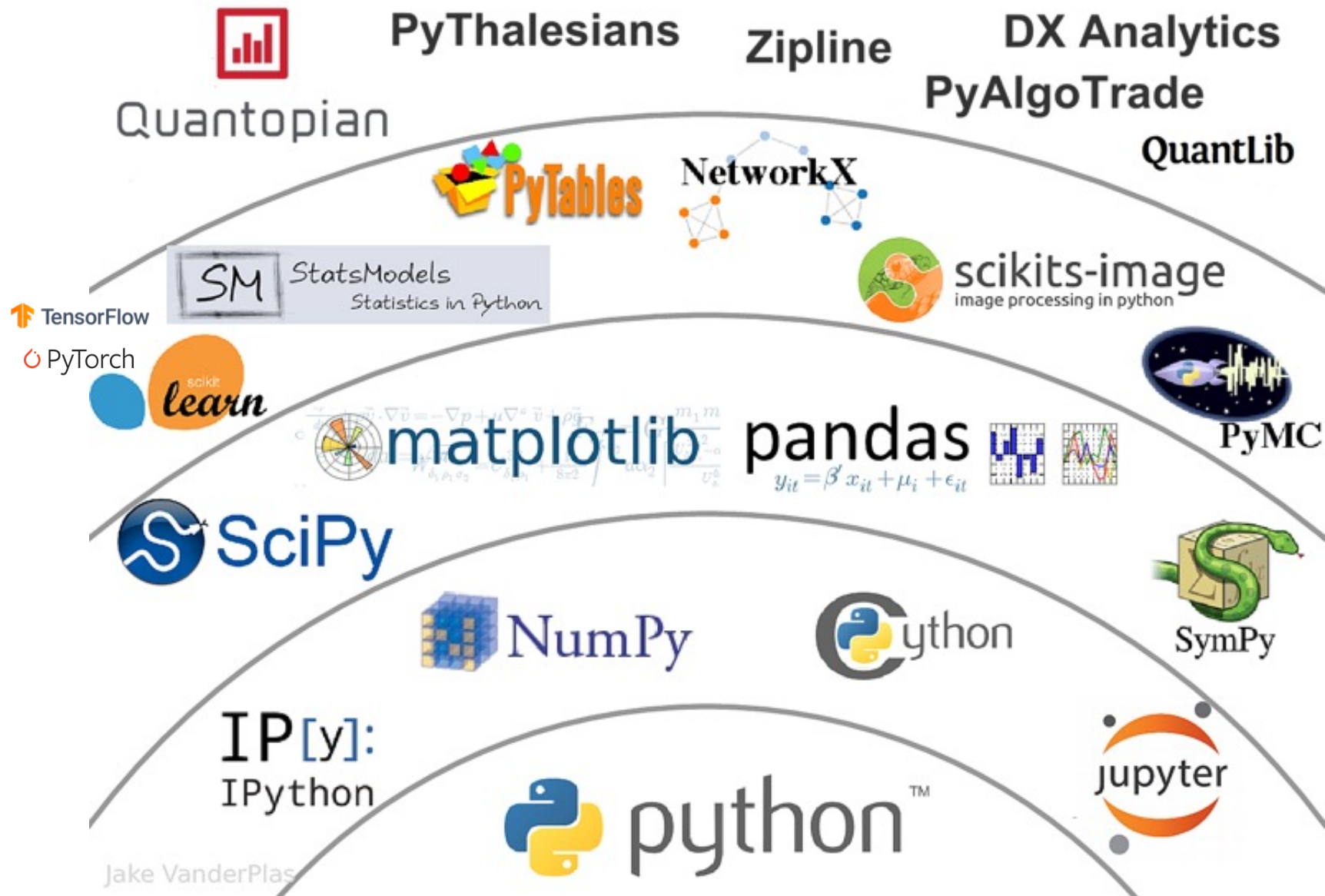
Python Ecosystem for Data Science



Python Ecosystem for Data Science



The Quant Finance PyData Stack



Google Colab

Table of contents

- 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.

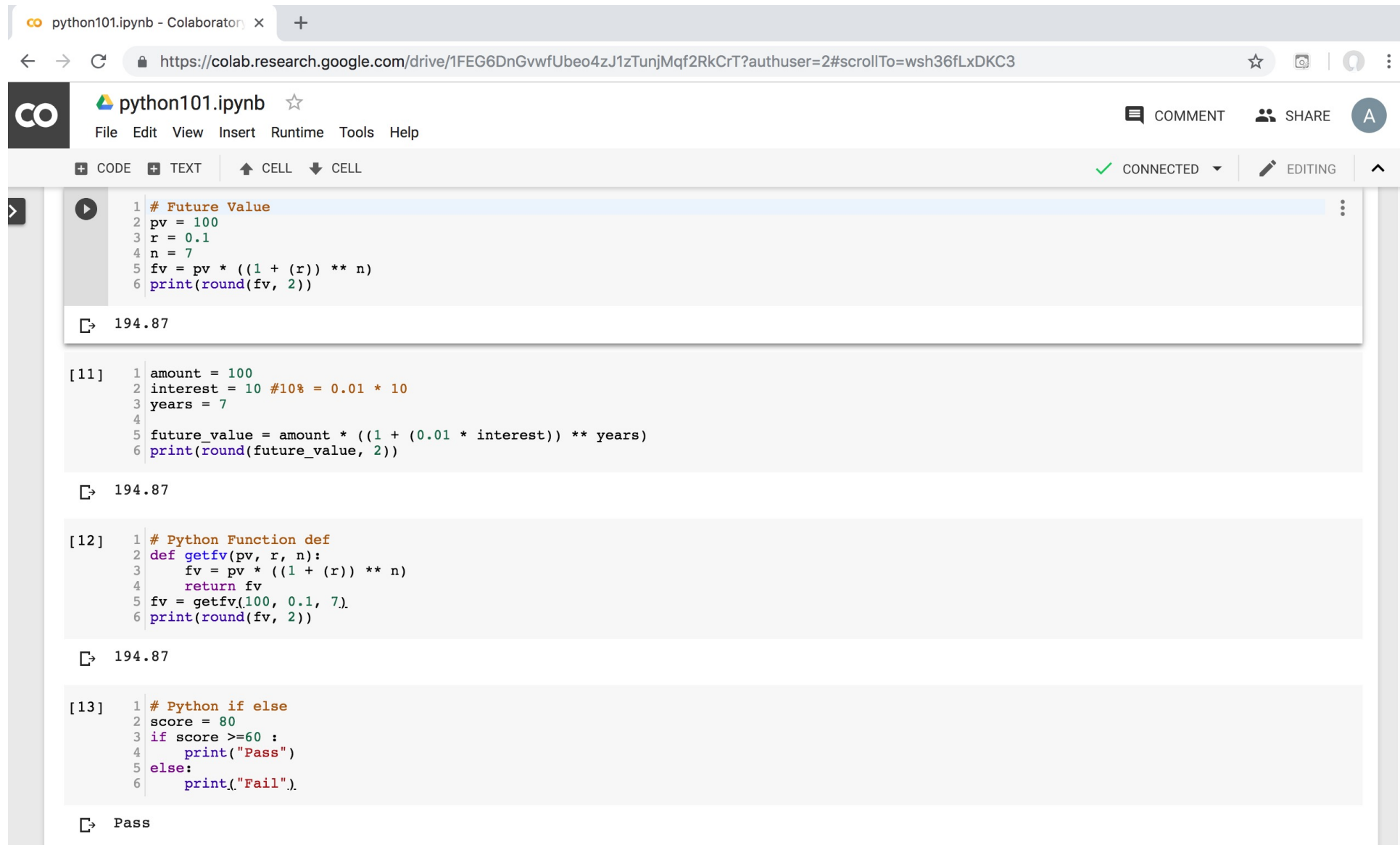
TensorFlow execution

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

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

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 browser address bar, a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), and a toolbar with options like CODE, TEXT, CELL, and a status indicator showing "CONNECTED" and "EDITING".

The notebook contains four code cells, each followed by its output:

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

194.87

```
[11] 1 amount = 100
     2 interest = 10 #10% = 0.01 * 10
     3 years = 7
     4
     5 future_value = amount * ((1 + (0.01 * interest)) ** years)
     6 print(round(future_value, 2))
```

194.87

```
[12] 1 # Python Function def
     2 def getfv(pv, r, n):
     3     fv = pv * ((1 + (r)) ** n)
     4     return fv
     5 fv = getfv(100, 0.1, 7).
     6 print(round(fv, 2))
```

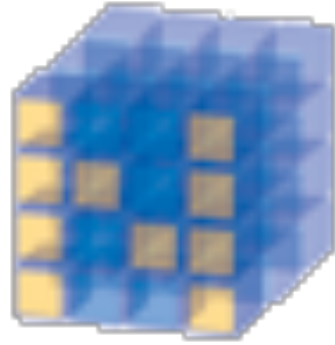
194.87

```
[13] 1 # Python if else
     2 score = 80
     3 if score >=60 :
     4     print("Pass")
     5 else:
     6     print("Fail").
```

Pass

<https://tinyurl.com/aintpupython101>

NumPy



NumPy

Base

N-dimensional array
package

Python

matplotlib

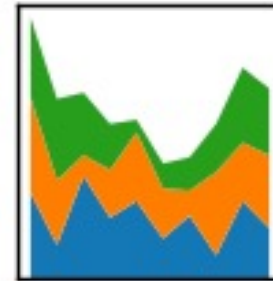
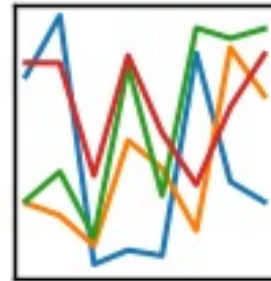
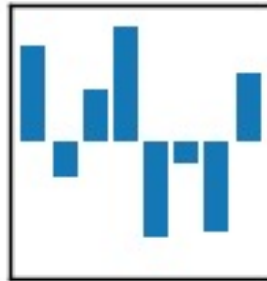
matplotlib

The logo for Matplotlib, which is a circular plot with a white background and a light gray grid. It features several colored wedges (orange, yellow, green, blue) radiating from the center, resembling a pie chart or a polar plot.

Python Pandas

pandas

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



Connect Google Colab in Google Drive

The image shows a browser window with the Google Drive interface. The address bar shows the URL `https://drive.google.com/drive/u/2/my-drive`. The main content area displays the 'My Drive' section with a 'Quick Access' sidebar on the left. The sidebar includes options like 'Computers', 'Shared with me', 'Recent', 'Starred', 'Trash', 'Backups', and 'Storage'. The 'Storage' section shows '0 bytes of 15 GB used' and a link to 'UPGRADE STORAGE'. A 'New' button is located at the top left of the main content area. A red dashed box highlights the 'New' button and the 'More' option in the dropdown menu. The 'More' option is expanded, showing a list of Google apps: Google Forms, Google Drawings, Google My Maps, Google Sites, and 'Connect more apps'. The 'Connect more apps' option is also highlighted with a red dashed box. The 'Files' section is visible at the bottom, showing a grid of cards for 'Store safely', 'Sync seamlessly', 'Access anywhere', and 'Share easily'. A notification banner at the bottom left says 'Get Backup and Sync for Mac'.

Google Colab

The screenshot shows the Google Drive web interface. A search bar at the top contains the text "colab". A dialog box titled "Connect apps to Drive" is open, displaying a grid of application cards. The search results are as follows:

App Name	User Count
ZIP Extractor	307,585 users
Lumin PDF - Beautiful PDF Editor	289,310 users
CloudConvert	373,161 users
Sejda	2,131,600 users
DocHub - Edit and Sign PDF Docu...	2,131,600 users
Google Forms	4,803,614 users

The "colab" search filter is highlighted with a red dashed border. The background interface shows the "My Drive" sidebar with options like "New", "Computers", "Shared with me", "Recent", "Starred", "Trash", "Backups", and "Storage".

Google Colab

The image shows a browser window with a Google Drive page. A modal dialog titled "Connect apps to Drive" is open in the center. The dialog has a search bar with "colab" entered. Below the search bar, a single app card for "Colaboratory" is displayed. The card includes the app's logo (two yellow circles), its name, the URL "https://colab.research.google.com", a description: "A data analysis tool that combines code, output, and descriptive text into one collaborative document.", and a rating of five stars with "(195)" reviews. A blue button with a white plus sign and the text "+ CONNECT" is positioned to the right of the app card. A red dashed border highlights the entire app card and the connect button. The background shows the Google Drive interface with a sidebar on the left containing navigation options like "My Drive", "Computers", "Shared with me", "Recent", "Starred", "Trash", "Backups", and "Storage". The top of the browser shows the address bar with the URL "https://drive.google.com/drive/u/2/my-drive".

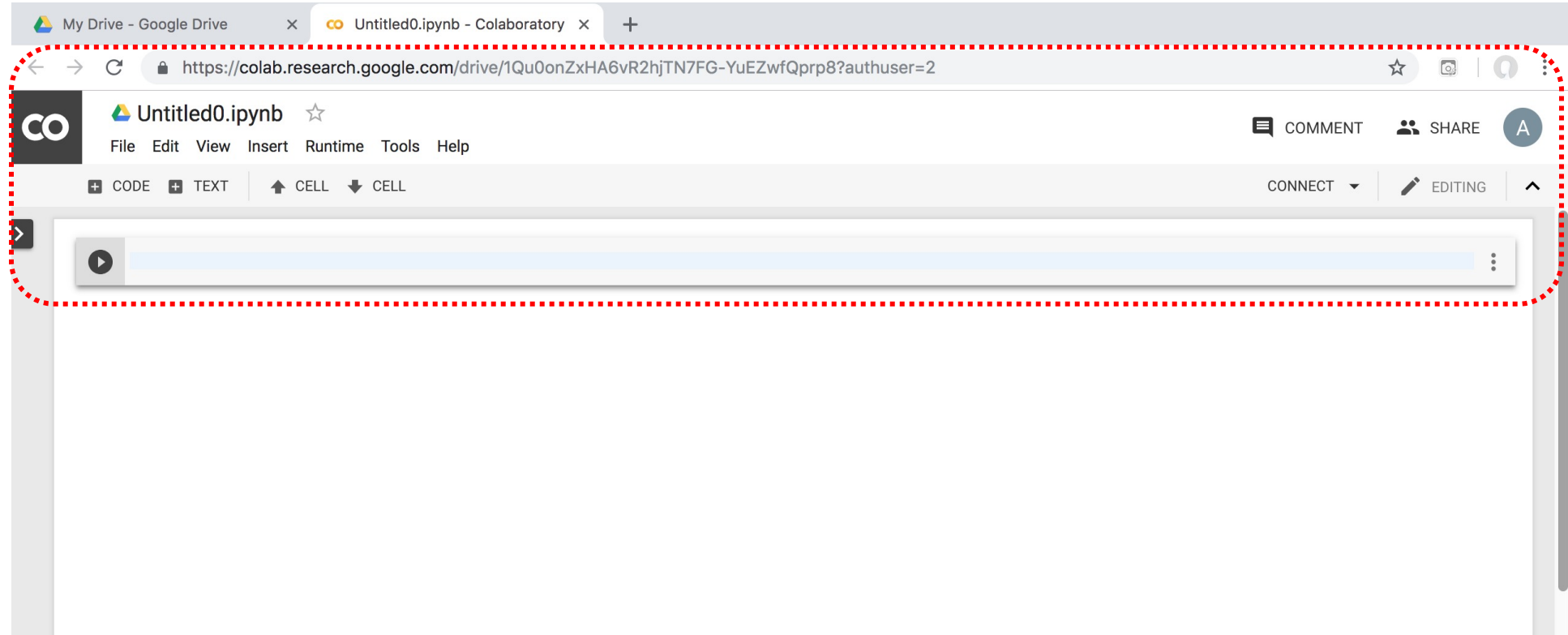
Connect Colaboratory to Google Drive

The screenshot shows the Google Drive web interface. A dialog box titled "Connect apps to Drive" is open, displaying a search for "colab". A confirmation message from Colaboratory is centered in the dialog, stating "Colaboratory was connected to Google Drive." and "Make Colaboratory the default app for files it can open" with a checked checkbox. An "OK" button is visible at the bottom right of the message. The background shows the Drive sidebar with categories like "My Drive", "Computers", "Shared with me", "Recent", "Starred", "Trash", "Backups", and "Storage". The storage status indicates "0 bytes of 15 GB used" with an "UPGRADE STORAGE" link. The top navigation bar includes the Drive logo, search bar, and various utility icons.

Google Colab

The image shows a browser window with the Google Drive interface. The address bar displays the URL `https://drive.google.com/drive/u/2/my-drive`. The main header includes the Drive logo, a search bar, and navigation icons. On the left sidebar, the 'New' button is highlighted with a red dashed box. A dropdown menu is open, listing various options: 'New folder...', 'Upload files...', 'Upload folder...', 'Google Docs', 'Google Sheets', 'Google Slides', 'More', 'Google Forms', 'Google Drawings', 'Google My Maps', 'Google Sites', 'Colaboratory', and 'Connect more apps'. The 'More' option in the first menu and the 'Colaboratory' option in the second menu are both highlighted with red dashed boxes. The background shows the 'Quick Access' section with 'My Drive' selected, and a 'Files' section with storage information (0 bytes of 15 GB used) and a 'Get Backup and Sync for Mac' notification.

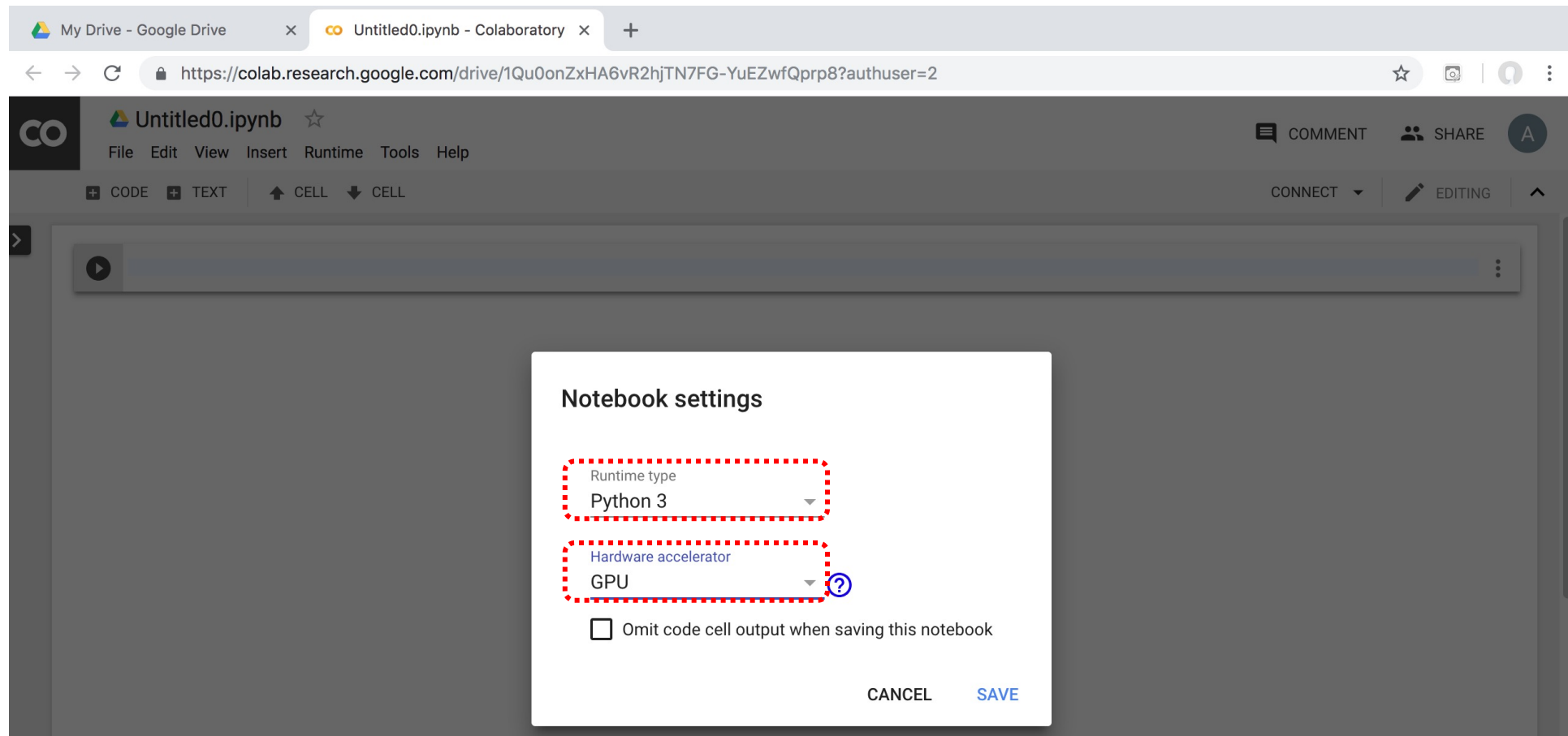
Google Colab



Google Colab

The image shows a browser window with two tabs: "My Drive - Google Drive" and "Untitled0.ipynb - Colaboratory". The address bar shows the URL: <https://colab.research.google.com/drive/1Qu0onZxHA6vR2hjTN7FG-YuEZwfQprp8?authuser=2>. The main interface displays the "Untitled0.ipynb" document with a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar (CODE, TEXT, CONNECT, EDITING). The "Runtime" menu is open, showing options such as "Run all", "Run before", "Run the focused cell", "Run selection", "Run after", "Interrupt execution", "Restart runtime...", "Restart and run all...", "Reset all runtimes...", "Change runtime type", and "Manage sessions". The "Runtime" menu title and the "Change runtime type" option are highlighted with red dashed boxes.

Run Jupyter Notebook Python3 GPU Google Colab



The screenshot shows the Google Colab web interface. The browser address bar displays the URL: `https://colab.research.google.com/drive/1Qu0onZxHA6vR2hjTN7FG-YuEZwfQprp8?authuser=2`. The notebook title is "Untitled0.ipynb". The top navigation bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". Below this, there are buttons for "+ CODE", "+ TEXT", "↑ CELL", and "↓ CELL". On the right side of the top bar, there are "COMMENT", "SHARE", and "CONNECT" options, along with an "EDITING" indicator.

The main content area is dimmed, and a "Notebook settings" dialog box is open in the center. The dialog has the following elements:

- Runtime type:** A dropdown menu currently set to "Python 3".
- Hardware accelerator:** A dropdown menu currently set to "GPU".
- Omit code cell output when saving this notebook
- Buttons for "CANCEL" and "SAVE".

Red dashed boxes highlight the "Runtime type" and "Hardware accelerator" dropdown menus. A blue question mark icon is visible next to the "Hardware accelerator" dropdown.

Google Colab Python Hello World

```
print('Hello World')
```





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>

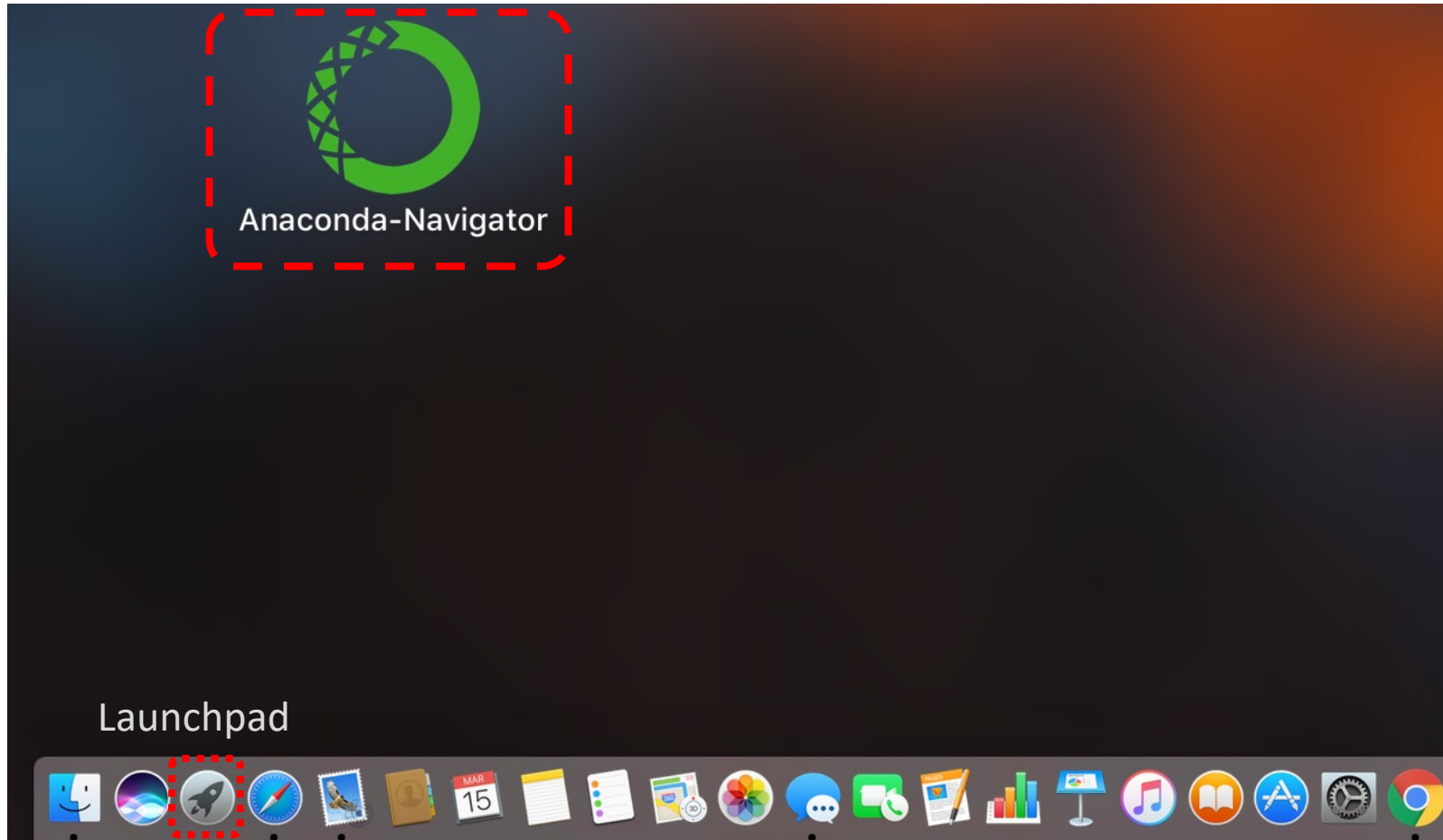




Python

HelloWorld

Anaconda-Navigator



Anaconda Navigator

The screenshot displays the Anaconda Navigator desktop application. At the top, the title bar reads "Anaconda Navigator". Below it, the application logo and name "ANACONDA NAVIGATOR" are on the left, and a "Sign in to Anaconda Cloud" button is on the right. A left-hand sidebar contains navigation options: Home, Environments, Learning, and Community. At the bottom of the sidebar are links for Documentation, Developer Blog, and Feedback, along with social media icons for Twitter, YouTube, and GitHub.

The main content area is titled "Applications on" and shows a dropdown menu set to "base (root)" and a "Channels" button. A "Refresh" button is in the top right of this section. The applications are arranged in a grid:

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

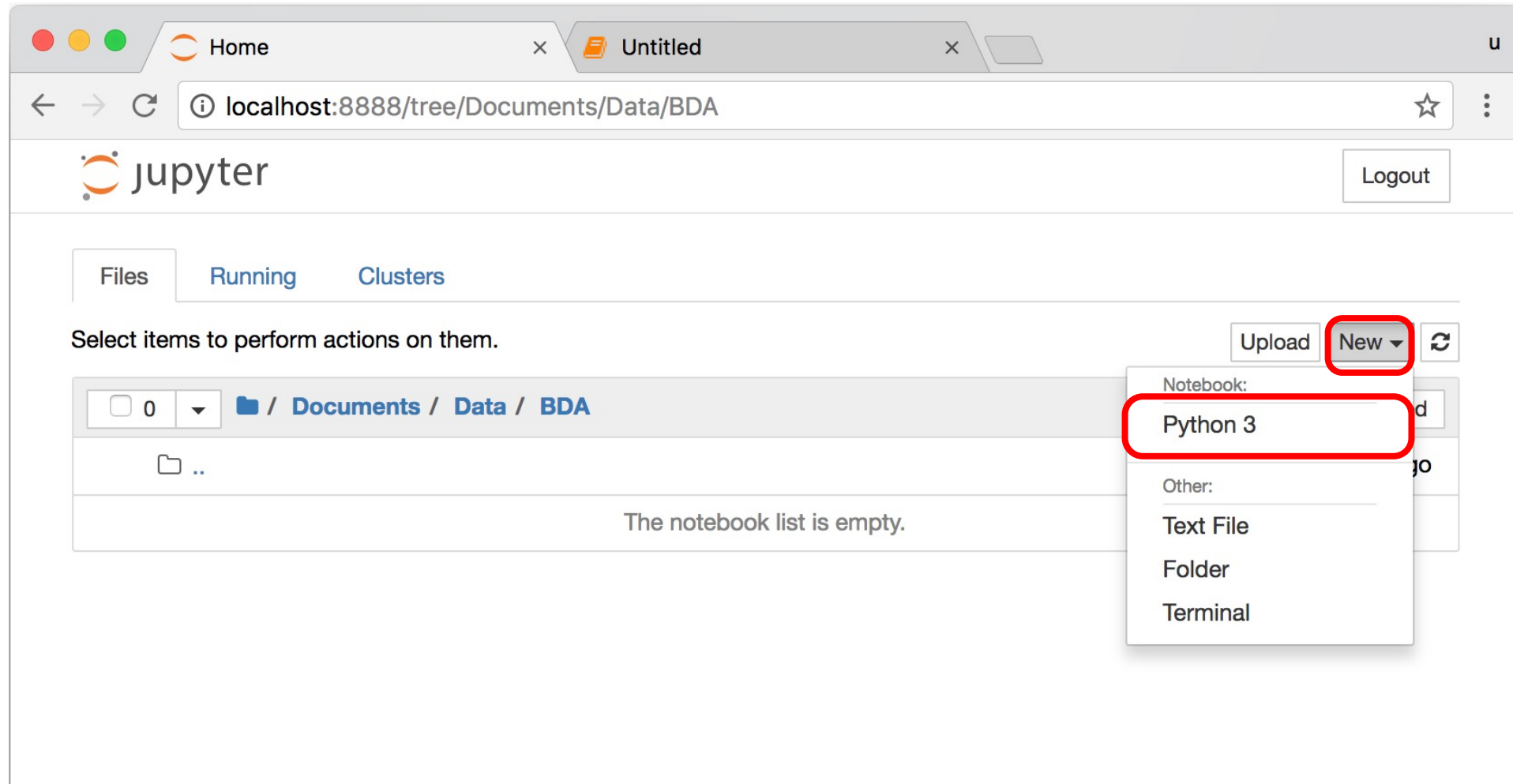
The "jupyter notebook" application card is highlighted with a red dashed border, and its "Launch" button is enclosed in a solid red box.

Jupyter Notebook

The screenshot shows a web browser window with the Jupyter Notebook interface. The browser's address bar displays the URL `localhost:8888/tree/Documents/Data/BDA`. The Jupyter logo and a "Logout" button are visible at the top. Below the navigation tabs ("Files", "Running", "Clusters"), there are buttons for "Upload", "New", and a refresh icon. A message "Select items to perform actions on them." is present. The main content area shows a file browser view for the path `/ Documents / Data / BDA`. A red dashed box highlights the file list area, which contains a single entry: a folder icon followed by `..`. To the right of the list are columns for "Name" and "Last Modified". The "Name" column has a downward arrow, and the "Last Modified" column shows "seconds ago". Below the list, a message states "The notebook list is empty."

Jupyter Notebook

New Python 3



The screenshot shows a web browser window with the Jupyter Notebook interface. The browser's address bar displays `localhost:8888/tree/Documents/Data/BDA`. The Jupyter logo and a "Logout" button are visible in the top left and right corners, respectively. Below the navigation tabs, there are buttons for "Files", "Running", and "Clusters". A message "Select items to perform actions on them." is displayed above a file browser area. The file browser shows a path of `/ Documents / Data / BDA` and a message "The notebook list is empty." A "New" dropdown menu is open, showing options for "Notebook:" (with "Python 3" selected) and "Other:" (with "Text File", "Folder", and "Terminal" listed). The "New" button and the "Python 3" option are highlighted with red boxes.

print("hello, world")

The screenshot shows a Jupyter Notebook interface in a web browser. The browser tabs include 'Home' and 'HelloWorld'. The address bar shows the URL 'localhost:8888/notebooks/Documents/Data/BDA/HelloWorld.ipynb'. The Jupyter logo and 'HelloWorld (autosaved)' are visible in the top left, along with a 'Logout' button and the Python logo. A menu bar contains 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. Below the menu is a toolbar with icons for file operations and a 'Run' button (a play icon) which is circled in red. The main area contains a code cell with the text 'In [1]: print("hello, world")' and its output 'hello, world'. The code cell is also circled in red. Below it is an empty code cell labeled 'In []:'.

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

The screenshot shows a Jupyter Notebook interface in a web browser. The browser tabs include 'Home' and 'HelloWorld'. The address bar shows 'localhost:8888/notebooks/Documents/Data/BDA/HelloWorld.ipynb'. The Jupyter logo and 'HelloWorld (unsaved changes)' are visible, along with a 'Logout' button. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. The toolbar contains icons for file operations and execution. The notebook content shows two code cells:

```
In [1]: print("hello, world")
hello, world
```

```
In [2]: from platform import python_version
print("Python Version:", python_version())
Python Version: 3.6.5
```

```
In [ ]:
```



Python

Programming



Python Fiddle

The screenshot shows the Python Fiddle web application interface. At the top, there is a browser window with the address bar showing "pythonfiddle.com". Below the browser, there is a navigation bar with buttons for "Run", "Reset", "Share", "Import", and "Login", along with a "Language" dropdown menu. The main content area is divided into three sections: a left sidebar with a list of "Examples" (including "Chaining comparison operators", "Decorators", "Creating generators objects", "Enumerate", "Function closure", "Lex tokenizer", "Step argument in slice operators", "For Else", "Verbose regular expressions", "In-place value swapping", and "Function argument unpacking"), a central code editor with the code `1 print("Hello Python Fiddle")` and `2`, and a right sidebar with form fields for "Title:", "Description:", and "Tags:", and a "Save" button. The "Python Fiddle" logo and "Python Cloud IDE" text are visible in the top right corner.

Hello Python Fiddle

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)
```

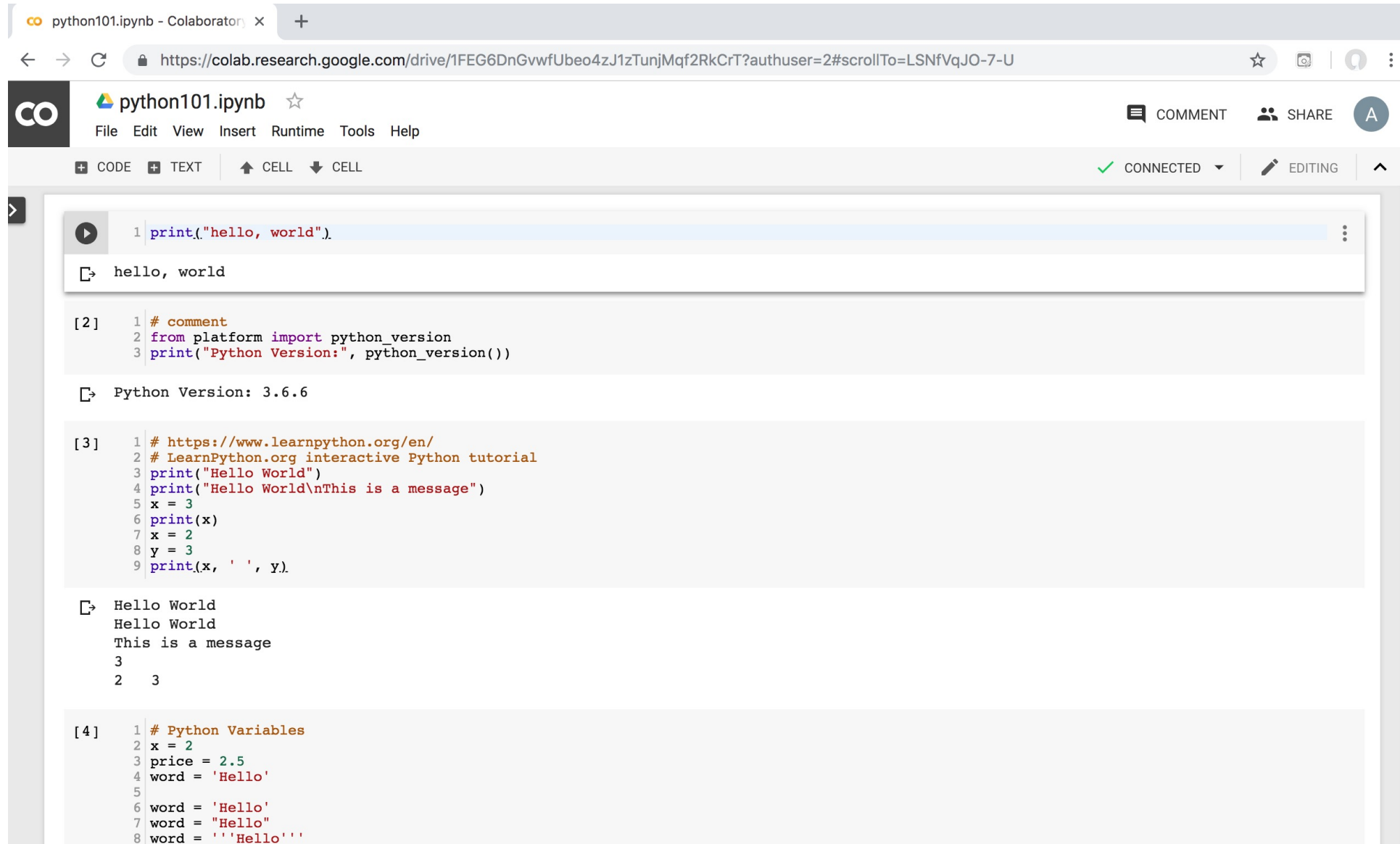
```
name = input("Enter a name: ")
```

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

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

Python in Google Colab

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



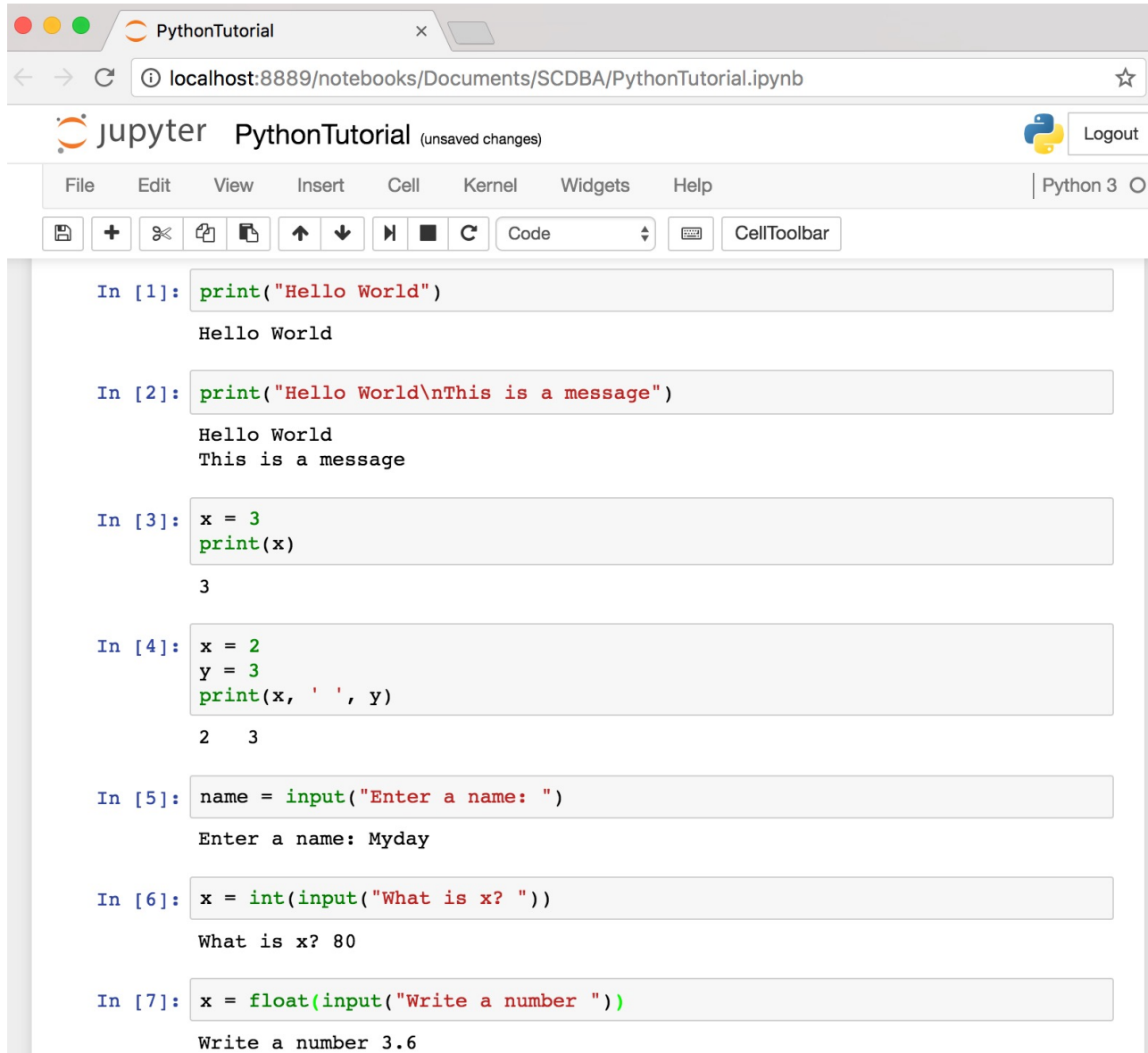
The screenshot shows a Google Colab notebook interface. The browser address bar displays the URL: <https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT?authuser=2#scrollTo=LSNfVqJO-7-U>. The notebook title is "python101.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a toolbar with options for CODE, TEXT, CELL, and a status bar showing "CONNECTED" and "EDITING".

The notebook contains four code cells:

- Cell 1:** `print("hello, world").` Output: `hello, world`
- Cell 2:** `# comment`
`from platform import python_version`
`print("Python Version:", python_version())` Output: `Python Version: 3.6.6`
- Cell 3:** `# https://www.learnpython.org/en/`
`# LearnPython.org interactive Python tutorial`
`print("Hello World")`
`print("Hello World\nThis is a message")`
`x = 3`
`print(x)`
`x = 2`
`y = 3`
`print(x, ' ', y).` Output: `Hello World`
`Hello World`
`This is a message`
`3`
`2 3`
- Cell 4:** `# Python Variables`
`x = 2`
`price = 2.5`
`word = 'Hello'`
`5`
`word = 'Hello'`
`word = "Hello"`
`8 word = '''Hello'''`

<https://tinyurl.com/aintpupython101>

Text input and output



The screenshot shows a Jupyter Notebook window titled "PythonTutorial" with the URL `localhost:8889/notebooks/Documents/SCDBA/PythonTutorial.ipynb`. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution. The notebook contains seven code cells, each with its 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

File Edit View Insert Cell Kernel Widgets Help

Python 3 

          Code   CellToolbar

```
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)
```

```
194.87171000000012
```

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)
```

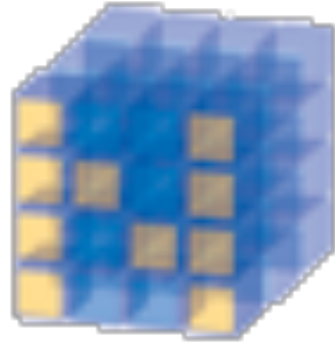
```
Hello my name is Alan  
Alan  
20
```

Big Data Analytics
with

Numpy

in Python

NumPy



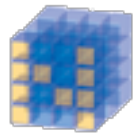
NumPy

Base

N-dimensional array

package

NumPy
is the
fundamental package
for
scientific computing
with **Python.**



NumPy

NumPy

- **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

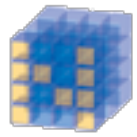
- [Getting NumPy](#)
- [Installing the SciPy Stack](#)
- [NumPy and SciPy documentation page](#)
- [NumPy Tutorial](#)
- [NumPy for MATLAB® Users](#)
- [NumPy functions by category](#)
- [NumPy Mailing List](#)

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

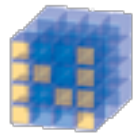
NumPy ndarray

One-dimensional Array (1-D Array)

0	1			n-1
1	2	3	4	5

Two-dimensional Array (2-D Array)

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



NumPy

NumPy

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

```
v
```

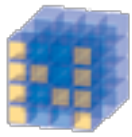
```
2 * v
```

```
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
```

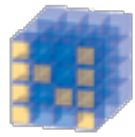
```
[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

```
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])
```



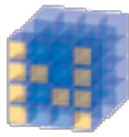
NumPy

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](#)[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
```

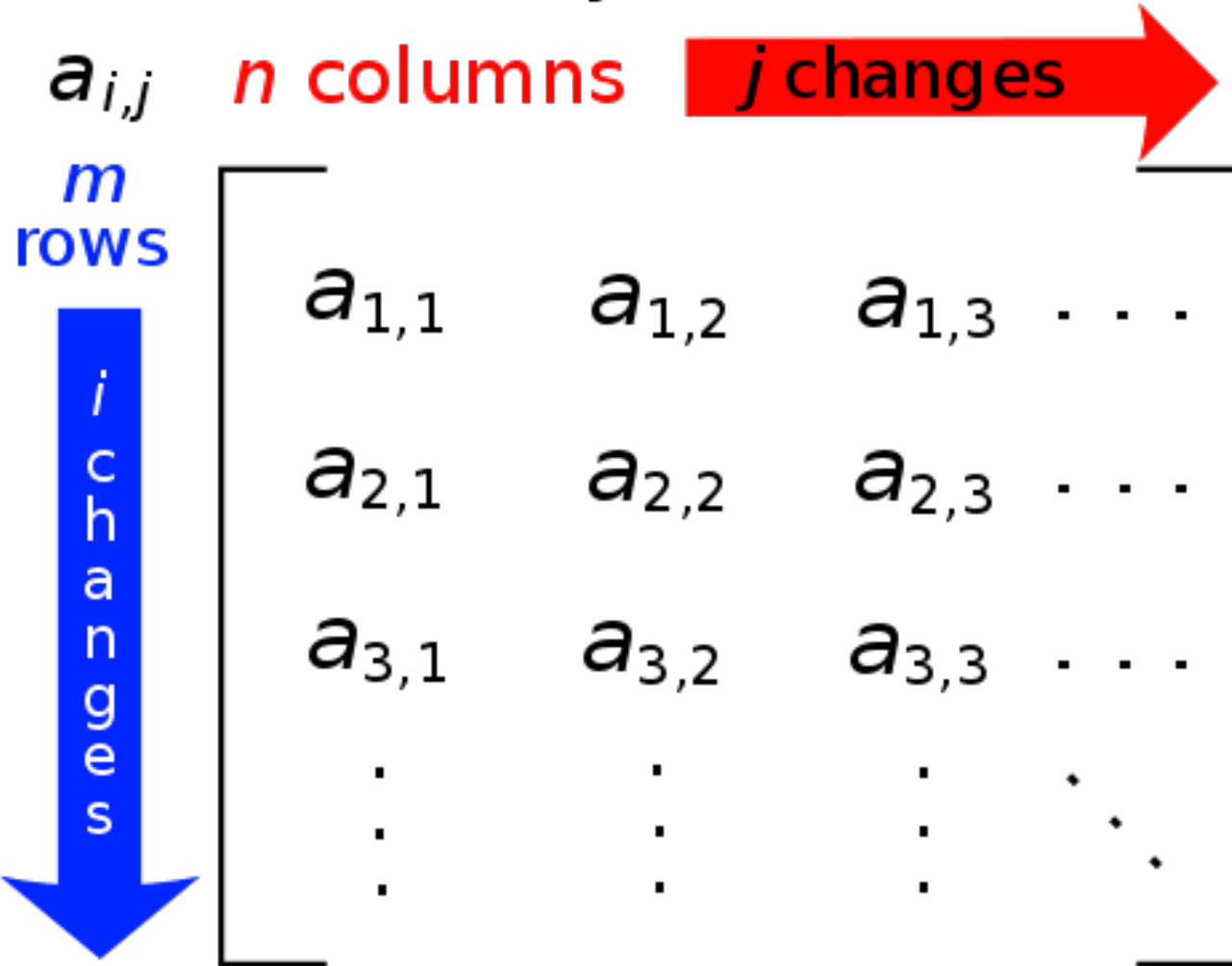
```
2
```

```
a.dtype.name
```

```
'int64'
```

Matrix

m -by- n matrix



NumPy ndarray: Multidimensional Array Object

NumPy ndarray

One-dimensional Array (1-D Array)

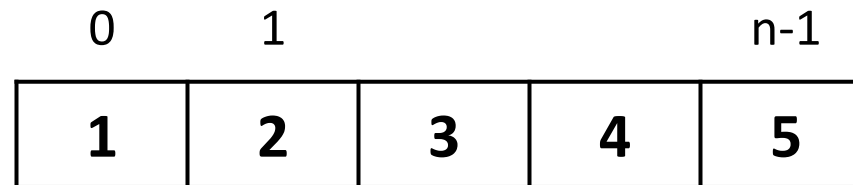
0	1			n-1
1	2	3	4	5

Two-dimensional Array (2-D Array)

	0	1		n-1	
0	1	2	3	4	5
1	6	7	8	9	10
	11	12	13	14	15
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)



```
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	5
1	6	7	8	9	10
	11	12	13	14	15
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

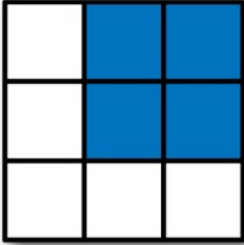
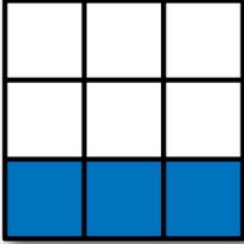
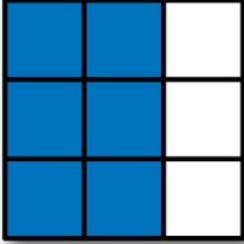
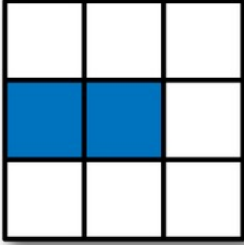
2,0

2,1

2,2

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

Numpy Array

	Expression	Shape
	<code>arr[:2, 1:]</code>	<code>(2, 2)</code>
	<code>arr[2]</code> <code>arr[2, :]</code> <code>arr[2:, :]</code>	<code>(3,)</code> <code>(3,)</code> <code>(1, 3)</code>
	<code>arr[:, :2]</code>	<code>(3, 2)</code>
	<code>arr[1, :2]</code> <code>arr[1:2, :2]</code>	<code>(2,)</code> <code>(1, 2)</code>

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 Watch 640 Star 4,031 Fork 4,348

[Code](#) [Issues 2](#) [Pull requests 0](#) [Projects 0](#) [Insights](#)

Branch: 2nd-edition **pydata-book / ch04.ipynb** Find file Copy path

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

Raw text

Sentence Segmentation

Tokenization

Part-of-Speech (POS)

Stop word removal

Stemming / **Lemmatization**

Dependency Parser

String Metrics & Matching

word's stem

am → am

having → hav

word's lemma

am → be

having → have

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 titled "python101.ipynb". The table of contents on the left lists various NLP topics. The main code cell contains the following Python code:

```
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)
```

The visual output of the code shows the sentence with named entities highlighted in colored boxes: "Steve Jobs" (PERSON), "Steve Wozniak" (PERSON), "Apple Computer" (ORG), "January 3, 1977" (DATE), "Cupertino" (GPE), and "California" (GPE).

```
[ ] 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
```

The output of the DataFrame is a table with the following columns: text, lemma, pos, tag, pos_explain, and stopwords.

	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 notebook titled "python101.ipynb". The interface includes a top menu bar with "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help". A "Table of contents" sidebar on the left lists various topics under "Text Analytics and Natural Language Processing (NLP)", including "Python for Natural Language Processing", "spaCy Chinese Model", "Open Chinese Convert", "Jieba", "NLTK", and "Stanza". The main content area shows a code cell with the following text:

```
+ Code + Text
RAM Disk
Editing
Text Analytics and Natural Language Processing (NLP)
Python for Natural Language Processing
spaCy
  • spaCy: Industrial-Strength Natural Language Processing in Python
  • Source: https://spacy.io/usage/spacy-101
[1] 1 !python -m spacy download en_core_web_sm
[3] 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 for token in doc:
    5     print(token.text, token.pos_, token.dep_)
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/aintpupython101>

Python in Google Colab (Python101)

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



python101.ipynb ☆

File Edit View Insert Runtime Tools Help [All changes saved](#)

Comment

Share



A

+ Code + Text

RAM
Disk

Editing

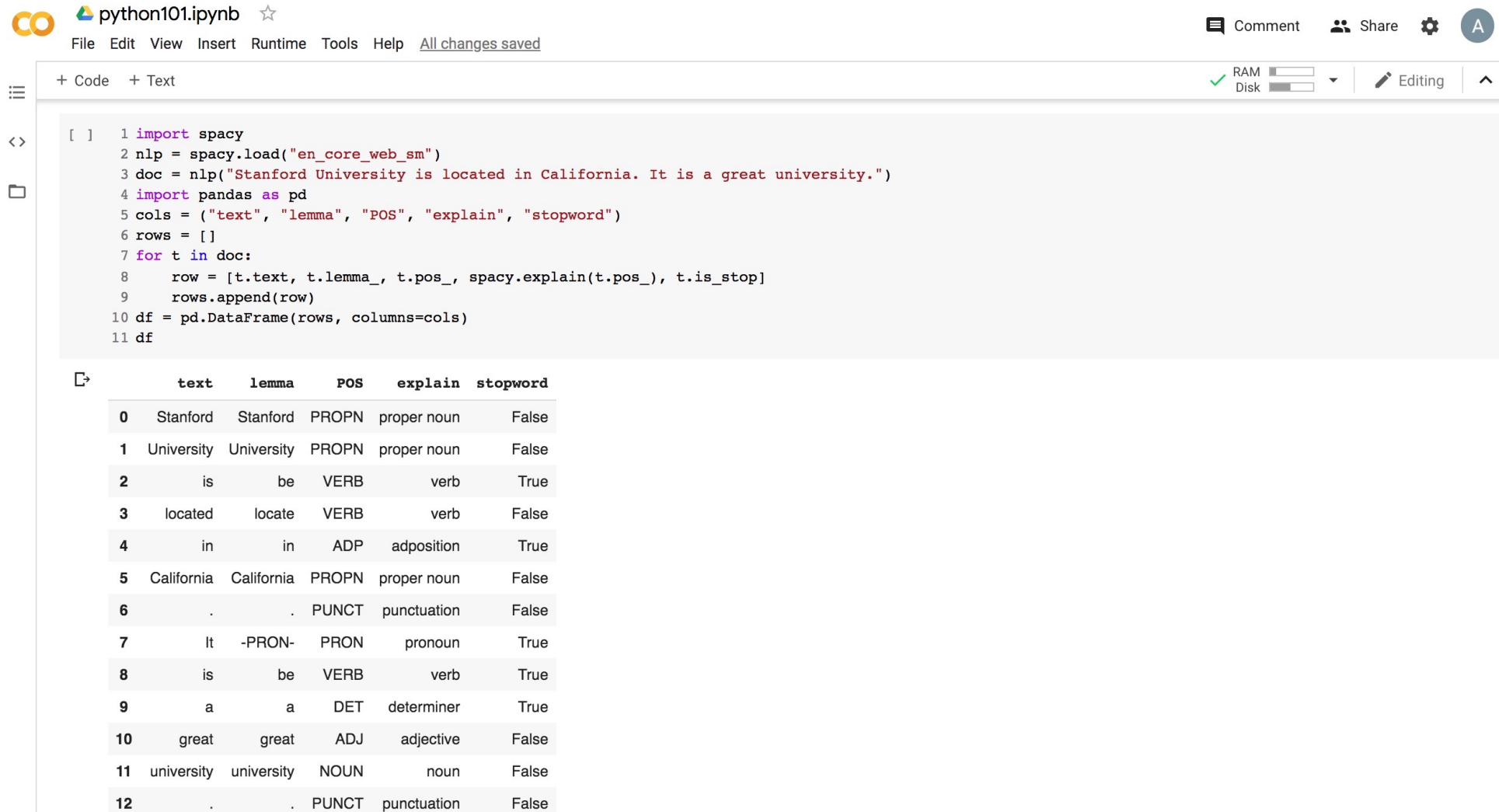
```
[ ] 1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
4 import pandas as pd
5 cols = ("text", "lemma", "POS", "explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_, spacy.explain(t.pos_), t.is_stop]
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/aintpupython101>

Python in Google Colab (Python101)

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



The screenshot shows a Google Colab notebook interface. At the top, the notebook is titled "python101.ipynb" and has a star icon. The menu bar includes "File", "Edit", "View", "Insert", "Runtime", "Tools", and "Help", with a status indicator "All changes saved". On the right, there are icons for "Comment", "Share", "Settings", and a user profile "A". Below the menu, there are tabs for "+ Code" and "+ Text", and a status bar showing "RAM" and "Disk" usage, a green checkmark, and "Editing" mode.

```
[ ] 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
```

Below the code, a table is displayed with the following columns: text, lemma, POS, explain, and stopwords. The table contains 12 rows of data, indexed from 0 to 12.

	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>



python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] 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_)
```

☞ Stanford University ORG
California GPE

```
[ ] 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)
```

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

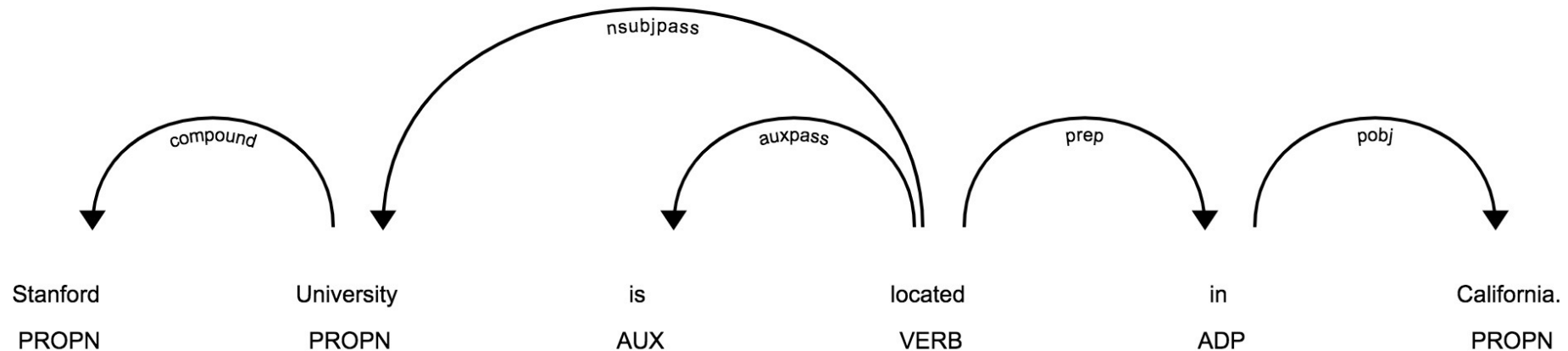
<https://tinyurl.com/aintpupython101>

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.



<https://tinyurl.com/aintpupython101>

Python in Google Colab

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



python101.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[ ] 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_)
```

↳ Stanford University ORG
California GPE

```
[ ] 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)
```

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

<https://tinyurl.com/aintpupython101>

Python in Google Colab (Python101)

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

The screenshot shows a Google Colab notebook titled "python101.ipynb". The left sidebar contains a "Table of contents" with various NLP topics. The main area displays a code cell with Python code for NLP processing using spaCy. The code defines a text string, processes it with spaCy, and renders the output with named entity recognition. Below the code, the output shows the original sentence with colored boxes around named entities: "Steve Jobs" (PERSON), "Steve Wozniak" (PERSON), "Apple Computer" (ORG), "January 3, 1977" (DATE), "Cupertino" (GPE), and "California" (GPE). A second code cell shows the creation of a pandas DataFrame from the spaCy output, which is then displayed as a table.

```
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)
```

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

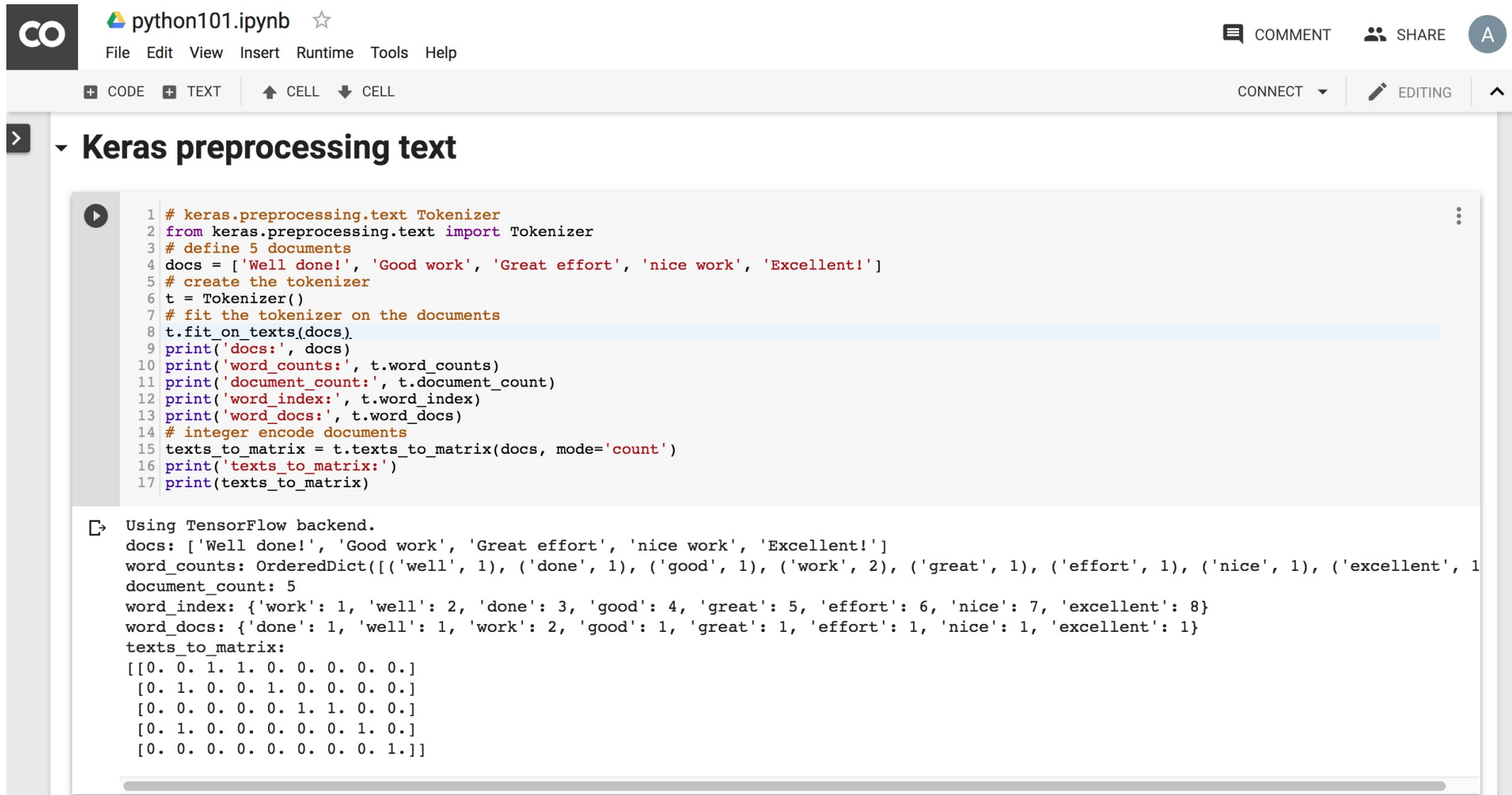
```
[ ] 1 import spacy
2 nlp = spacy.load("en_core_web_sm")
3 doc = nlp("Stanford University is located in California. It is a great university.")
4 import pandas as pd
5 cols = ("text", "lemma", "pos", "tag", "pos_explain", "stopword")
6 rows = []
7 for t in doc:
8     row = [t.text, t.lemma_, t.pos_,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

	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	VRB	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 notebook interface. At the top, the notebook is titled 'python101.ipynb'. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. On the right, there are buttons for 'COMMENT', 'SHARE', and a user profile icon. Below the menu bar, there are tabs for '+ CODE', '+ TEXT', and cell navigation arrows. The main content area is titled 'Keras preprocessing text' and contains a code cell with the following Python code:

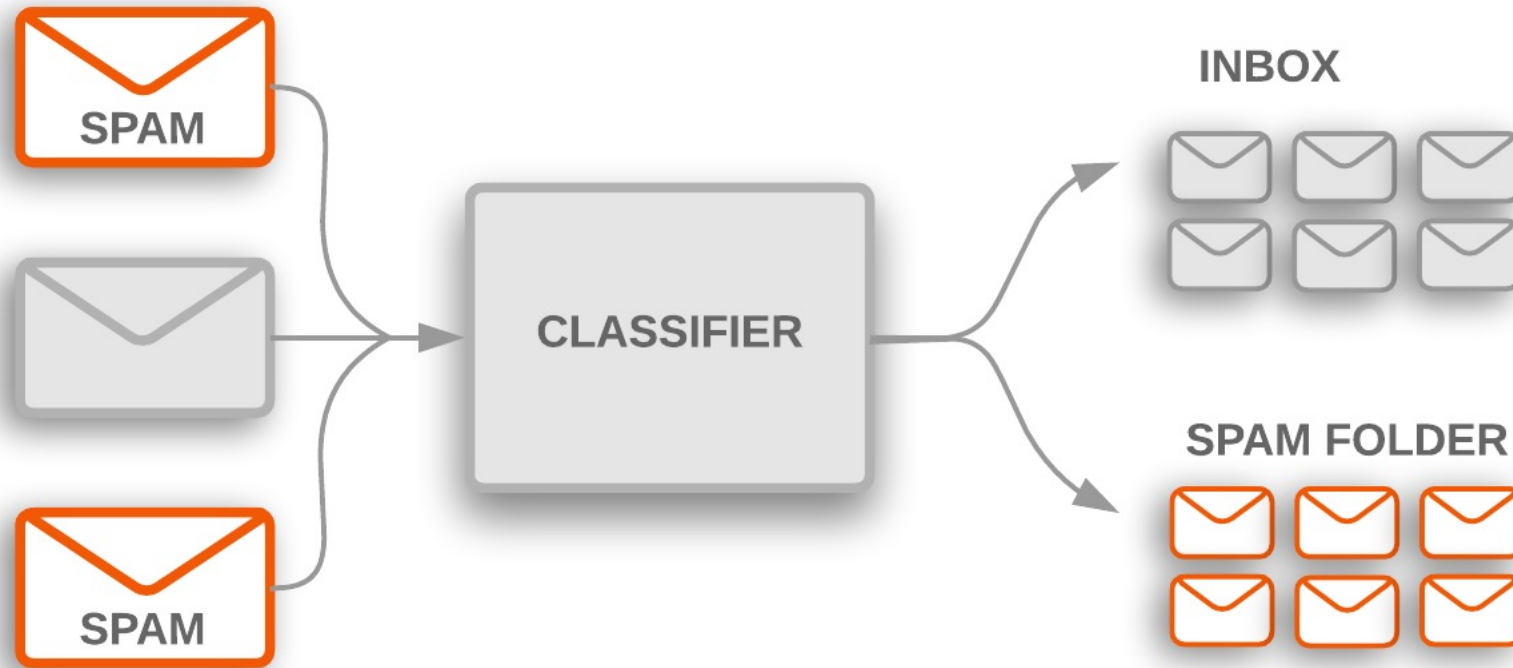
```
1 # keras.preprocessing.text Tokenizer
2 from keras.preprocessing.text import Tokenizer
3 # define 5 documents
4 docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
5 # create the tokenizer
6 t = Tokenizer()
7 # fit the tokenizer on the documents
8 t.fit_on_texts(docs)
9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

The output of the code cell is as follows:

```
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.]
 [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.]]
```

<https://tinyurl.com/aintpupython101>

Text Classification

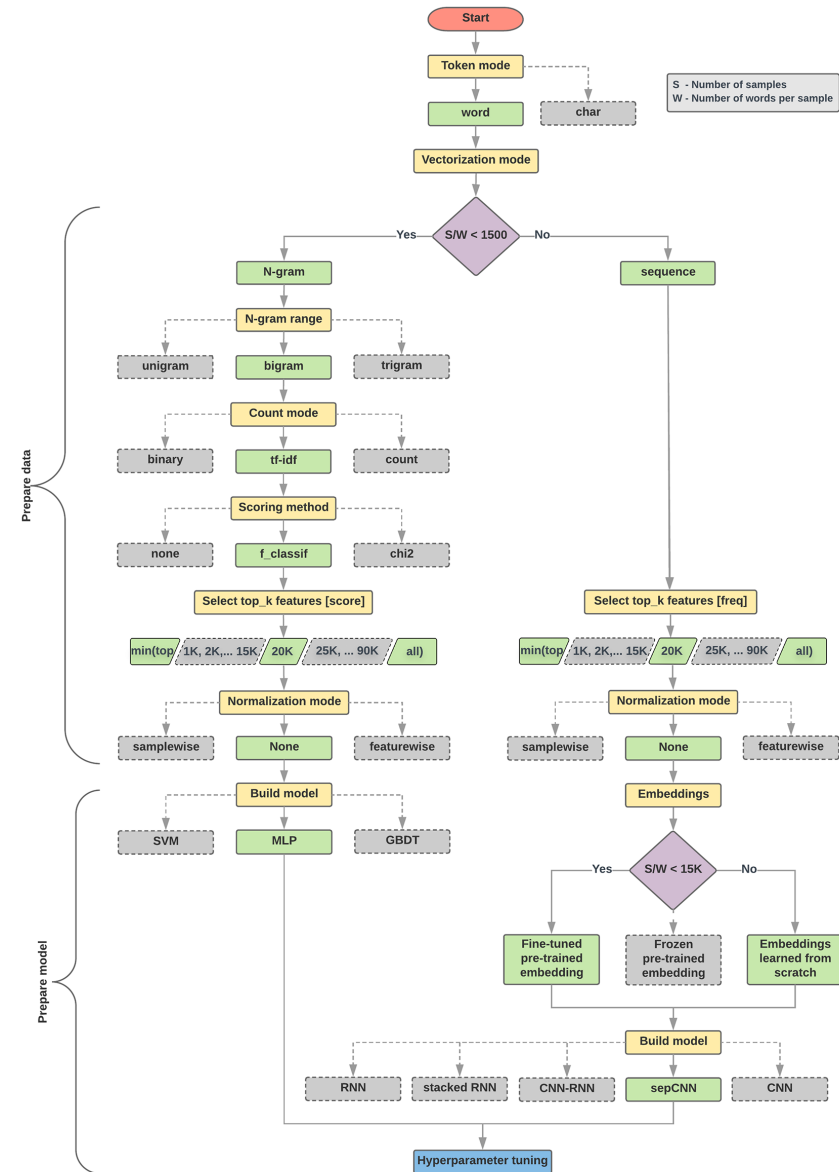


Text Classification Workflow

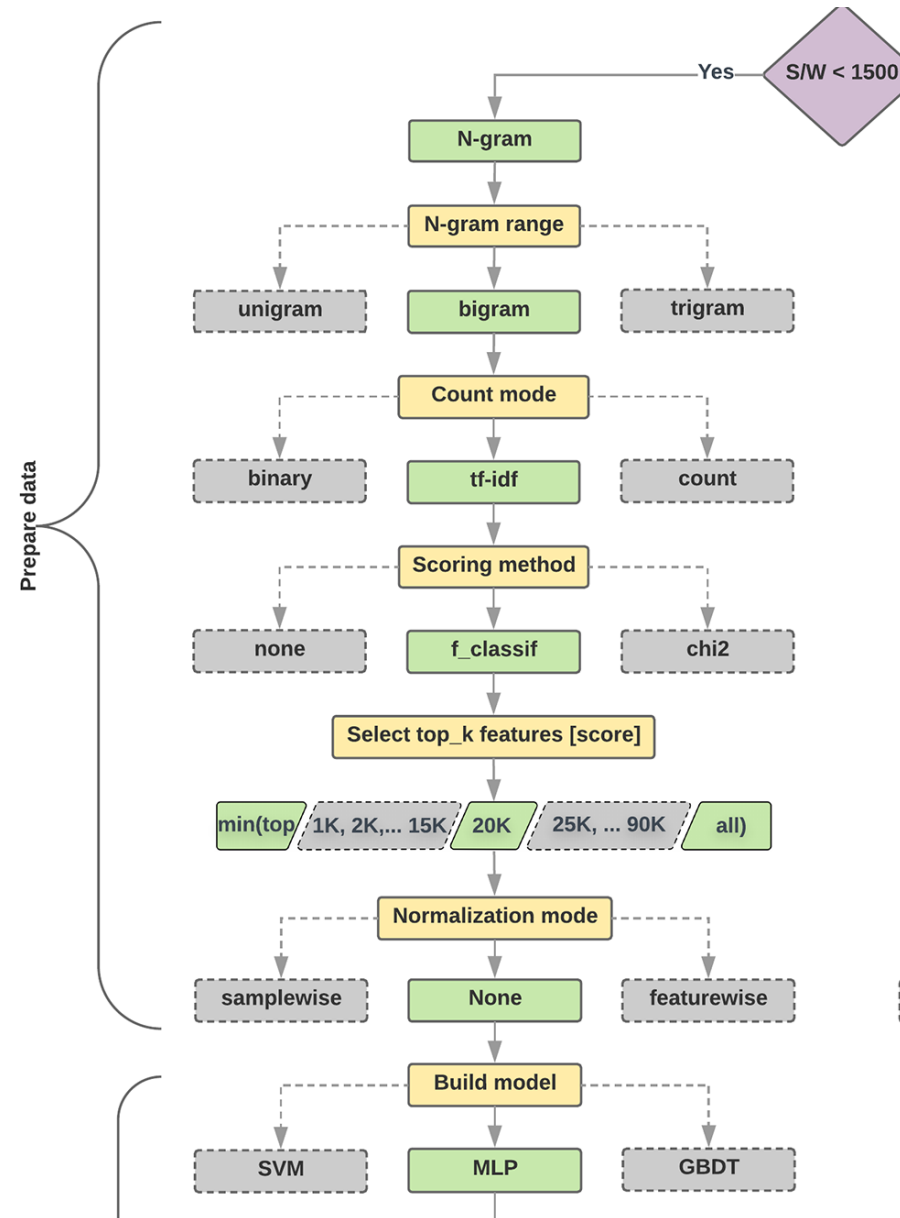
- **Step 1: Gather Data**
- **Step 2: Explore Your Data**
- **Step 2.5: Choose a Model***
- **Step 3: Prepare Your Data**
- **Step 4: Build, Train, and Evaluate Your Model**
- **Step 5: Tune Hyperparameters**
- **Step 6: Deploy Your Model**



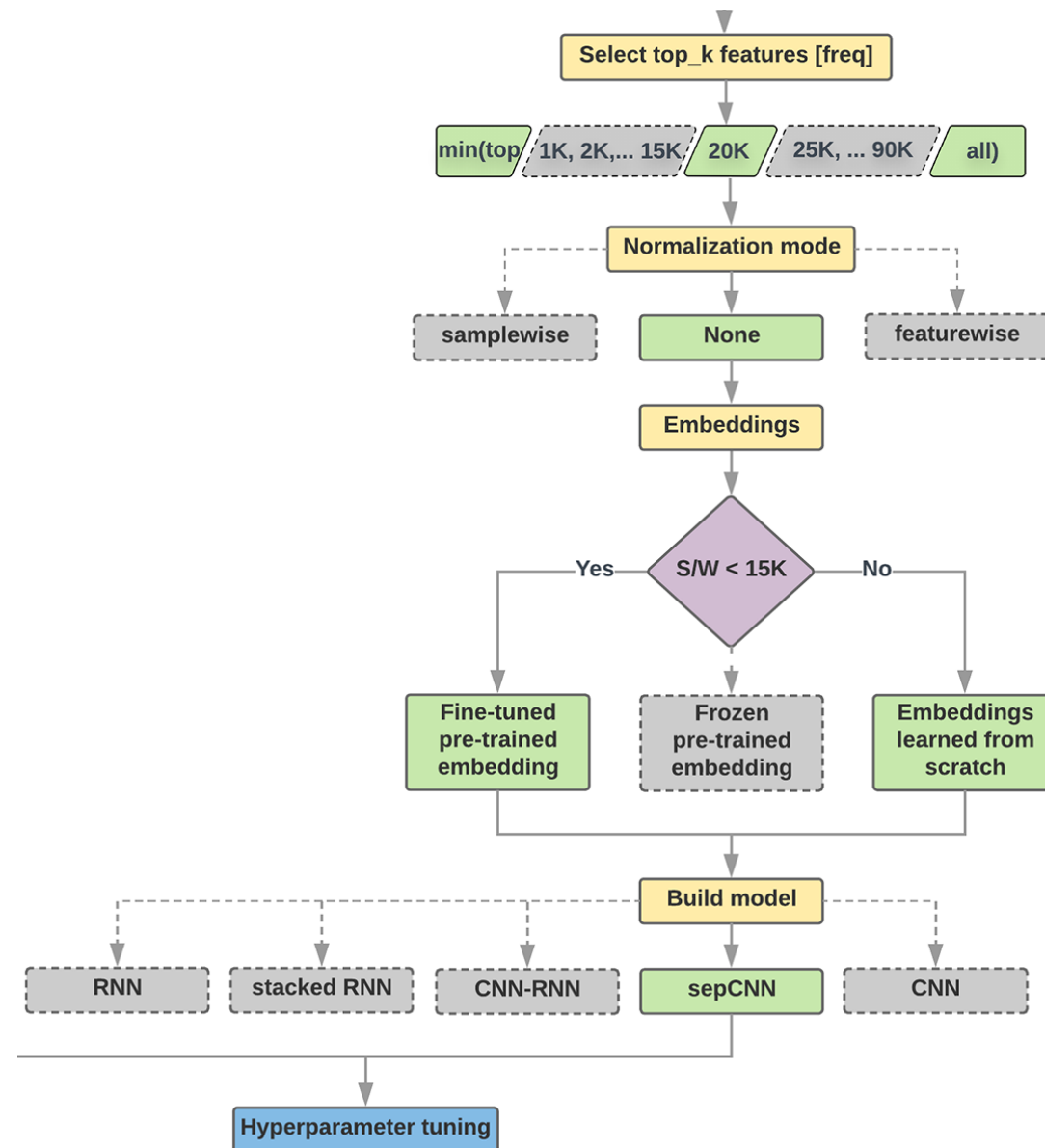
Text Classification Flowchart



Text Classification $S/W < 1500$: N-gram



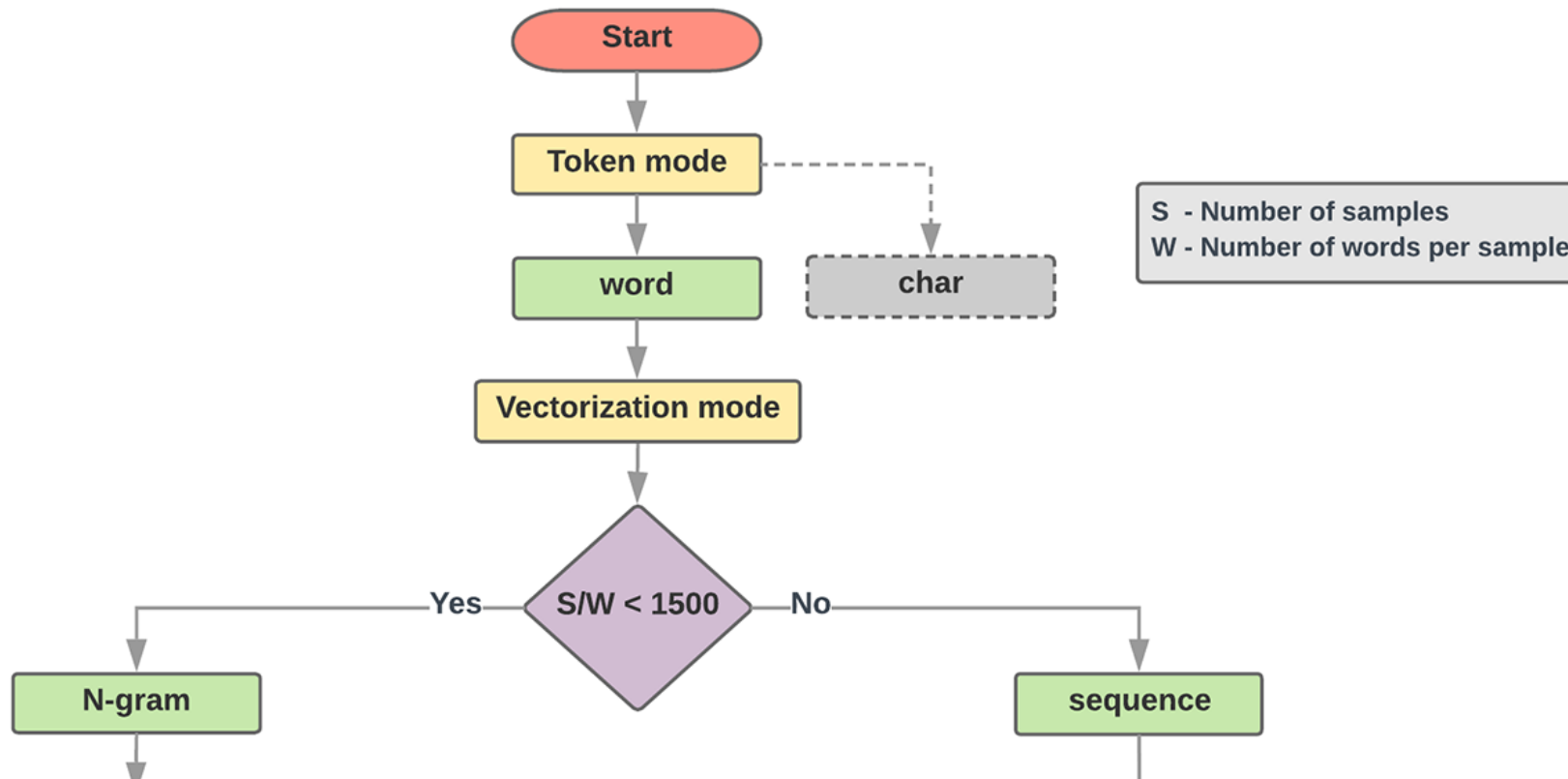
Text Classification $S/W \geq 1500$: Sequence



Step 2.5: Choose a Model

Samples/Words < 1500

150,000/100 = 1500

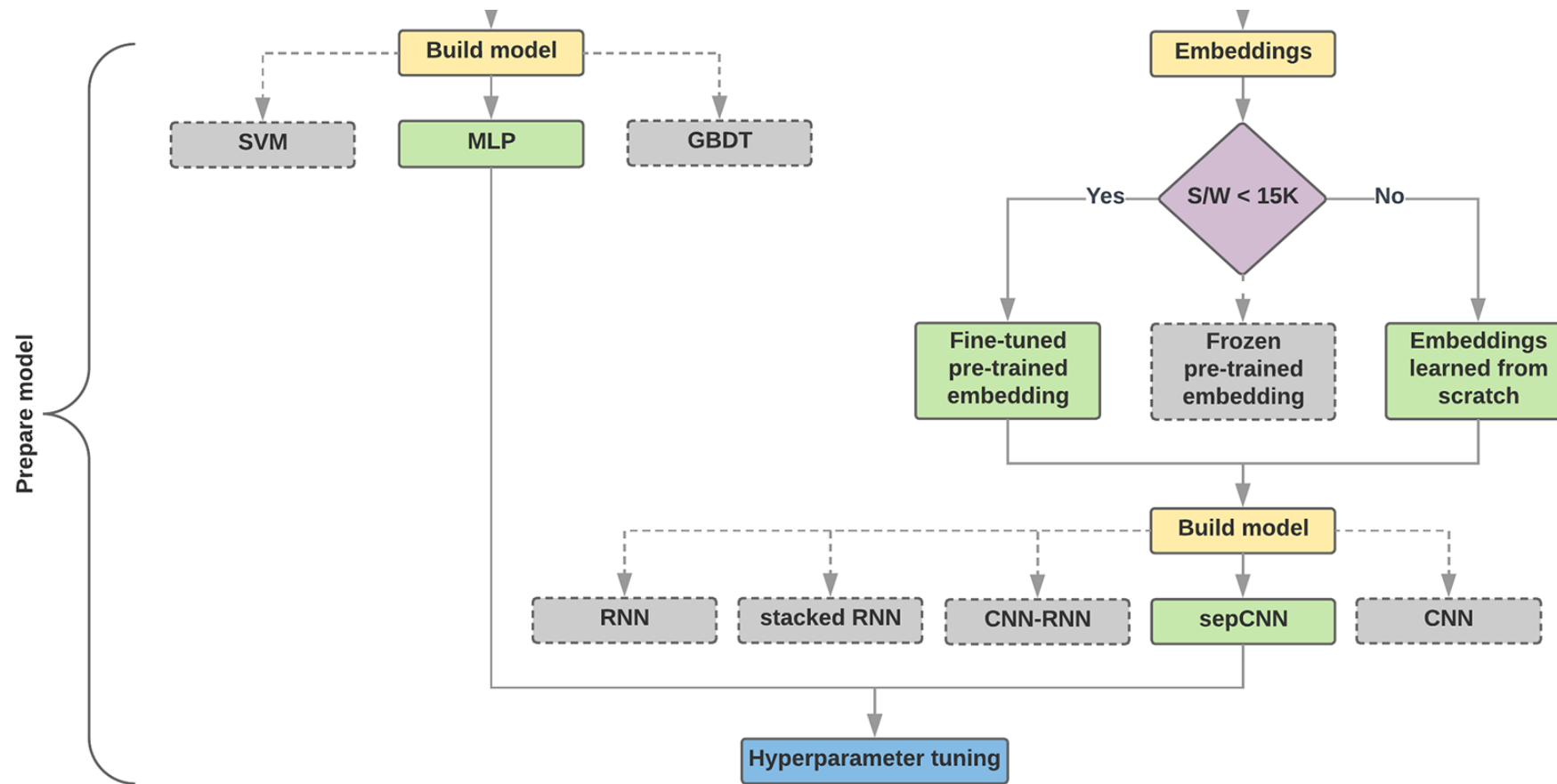


IMDb review dataset,
the samples/words-per-sample ratio is ~ 144

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]

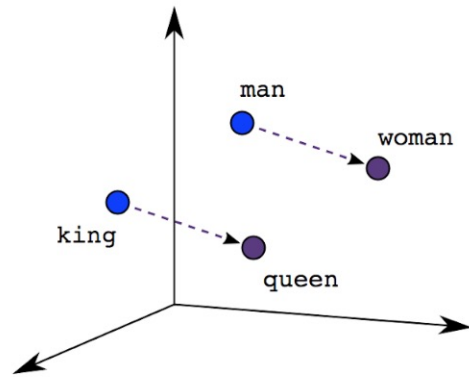
T2: 'The mouse ran down' =
[1, 2, 3, 6]

One-hot encoding

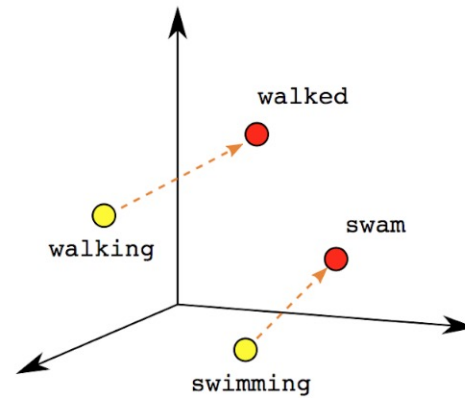
'The mouse ran up the clock' =

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]

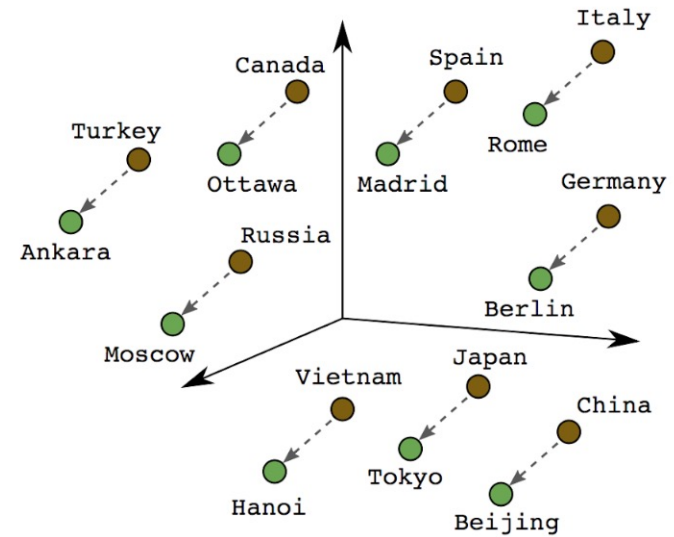
Word embeddings



Male-Female

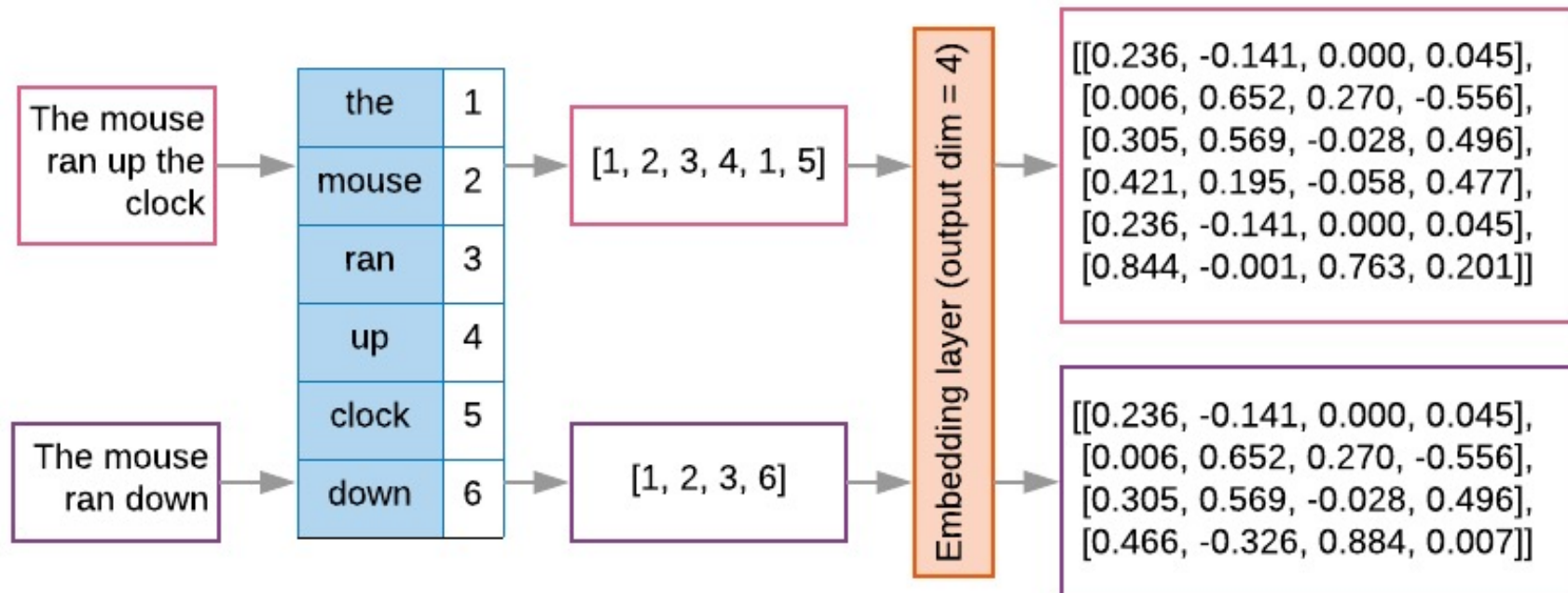


Verb Tense



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.]
 [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.]]
```

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.  0.98082925  0.  0.  1.25276297  0.  0.  0.  0. ]
 [0.  0.  0.  0.  0.  1.25276297  1.25276297  0.  0. ]
 [0.  0.98082925  0.  0.  0.  0.  0.  1.25276297  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

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

NLTK

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

Fork me on GitHub

 **gensim**
topic modelling for humans

 **Download**
latest version from the Python Package Index

 **Direct install with:**
easy_install -U gensim

Home | Tutorials | Install | Support | API | About

```
>>> 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

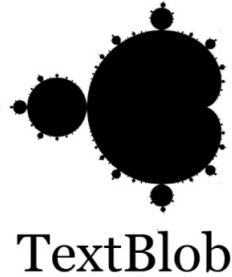
- ✓ Scalable statistical semantics
- ✓ Analyze plain-text documents for semantic structure
- ✓ Retrieve semantically similar documents


TextBlob

TextBlob: 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.



 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,

```
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

🏠 polyglot
latest

- Installation
- Language Detection
- Tokenization
- Command Line Interface
- Downloading Models
- Word Embeddings
- Part of Speech Tagging
- Named Entity Extraction
- Morphological Analysis
- Transliteration
- Sentiment
- polyglot

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



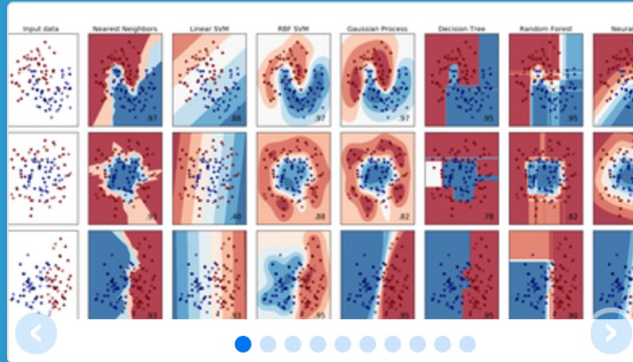
powered by Google

[Home](#) [Installation](#) [Documentation](#) [Examples](#)

Google Custom Search

Search

Fork me on GitHub



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

tf02_basic-text-classification.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CONNECT EDITING

Table of contents Code snippets Files

- Copyright 2018 The TensorFlow Authors.
 - Licensed under the Apache License, Version 2.0 (the "License");
 - MIT License
- Text classification with movie reviews**
 - Download the IMDB dataset
 - Explore the data
 - Convert the integers back to words
 - Prepare the data
 - Build the model
 - Hidden units
 - Loss function and optimizer
 - Create a validation set
 - Train the model
 - Evaluate the model

▶ Copyright 2018 The TensorFlow Authors.

↳ 2 cells hidden

Text classification with movie reviews

[View on TensorFlow.org](#) [Run in Google Colab](#) [View source on GitHub](#)

This notebook classifies movie reviews as *positive* or *negative* using the text of the review. This is an example of *binary*—or two-class—classification, an important and widely applicable kind of machine learning problem.

We'll use the [IMDB dataset](#) that contains the text of 50,000 movie reviews from the [Internet Movie Database](#). These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are *balanced*, meaning they contain an equal number of positive and negative reviews.

This notebook uses [tf.keras](#), a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using `tf.keras`, see the [MLCC Text Classification Guide](#).

```
1 # memory footprint support libraries/code
2 !ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
3 !pip install gputil
4 !pip install psutil
5 !pip install humanize
6 import psutil
7 import humanize
8 import os
9 import GPUtil as GPU
10 GPUs = GPU.getGPUs()
11 gpu = GPUs[0]
12 def printm():
13     process = psutil.Process(os.getpid())
```

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

NLP with Transformers Github

The screenshot shows the GitHub repository page for 'nlp-with-transformers/notebooks'. The repository is public and has 170 forks and 1.1k stars. The repository contains several Jupyter notebooks and scripts, including '01_introduction.ipynb', '02_classification.ipynb', '03_transformer-anatomy.ipynb', '04_multilingual-ner.ipynb', and '05_text-generation.ipynb'. The repository is licensed under Apache-2.0 and has 33 watchers and 170 forks. The repository is about the book 'Natural Language Processing with Transformers' by Lewis Tunstall, Leandro von Werra, and Thomas Wolf.

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

lewtun 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

Releases

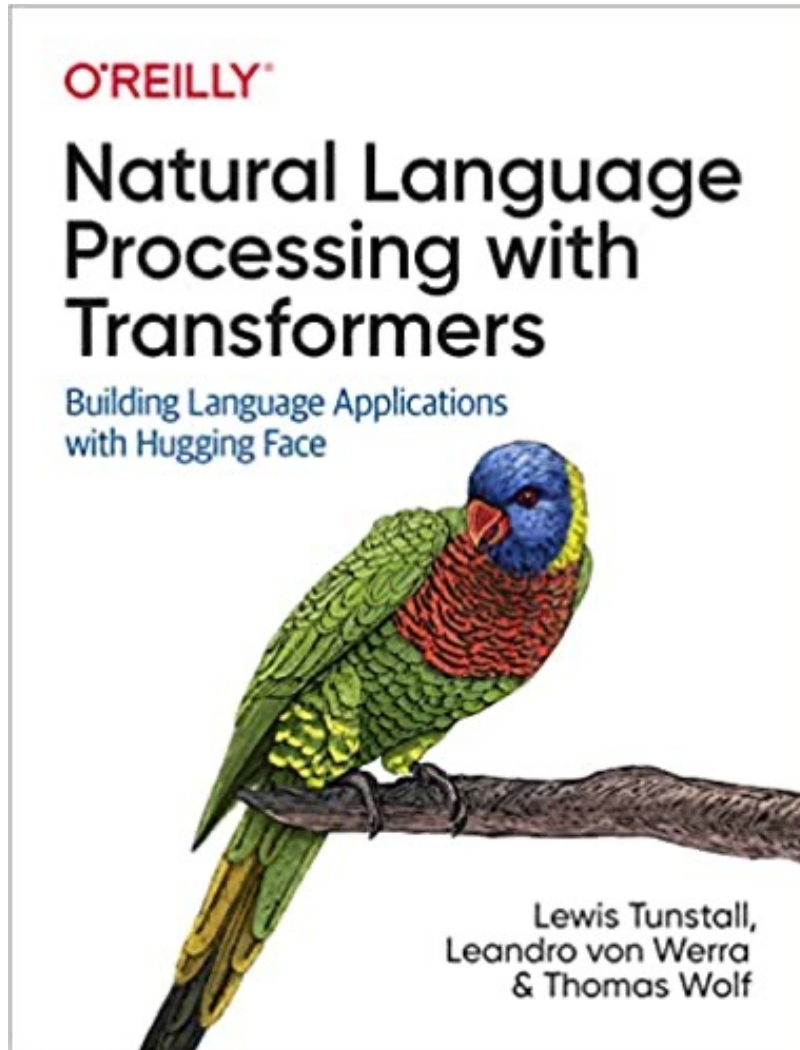
No releases published

Packages

O'REILLY
Natural Language Processing with Transformers
Building Language Applications with Hugging Face
Lewis Tunstall, Leandro von Werra & Thomas Wolf

<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 notebook titled "python101.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a "Table of contents" sidebar on the left. The main workspace displays a code cell with the following Python code:

```
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)
```

The output of the code cell shows the text with named entities highlighted in colored boxes: "Steve Jobs" (PERSON), "Steve Wozniak" (PERSON), "Apple Computer" (ORG), "January 3, 1977" (DATE), "Cupertino" (GPE), and "California" (GPE).

Below the code cell, there is another code cell that imports spaCy and processes the text "Stanford University is located in California. It is a great university." The output is a DataFrame with the following columns: text, lemma, pos, tag, pos_explain, and stopwords.

	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	VRB	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/>
- Min-Yuh Day (2022), Python 101, <https://tinyurl.com/aintpupython101>