

# Python for Accounting Applications

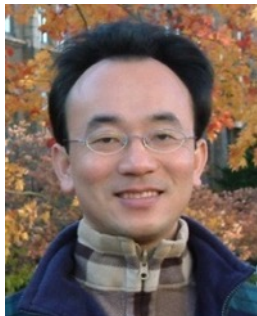
## Control Logic and Loops

1131PAA04

ACC2, NTPU (U2004) (Fall 2024)

Wed 6, 7, 8, (14:10-17:00) (9:10-12:00) (B3F10)

aws  
educate | Cloud  
Ambassador  
2020 Cohort



Min-Yuh Day, Ph.D,  
Professor

Institute of Information Management, National Taipei University

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

2024-10-09



# Syllabus

**Week Date Subject/Topics**

**1 2024/09/11 Introduction to Python for Accounting Applications**

**2 2024/09/18 Python Programming and Data Science**

**3 2024/09/25 Foundations of Python Programming**

**4 2024/10/02 Data Structures**

**5 2024/10/09 Control Logic and Loops**

**6 2024/10/16 Functions and Modules; Files and Exception Handling**

**7 2024/10/23 Data Analytics and Visualization with Python**

**8 2024/10/30 Midterm Project Report**

# Syllabus

**Week Date Subject/Topics**

9 2024/11/06 Self-Learning

10 2024/11/13 Obtaining Data From the Web with Python

11 2024/11/20 Statistical Analysis with Python

12 2024/11/27 Machine Learning with Python

13 2024/12/04 Text Analytics with Generative AI and Python

14 2024/12/11 Applications of Accounting Data Analytics with Python

15 2024/12/18 Applications of ESG Data Analytics with Python

16 2024/12/25 Final Project Report

# Outline

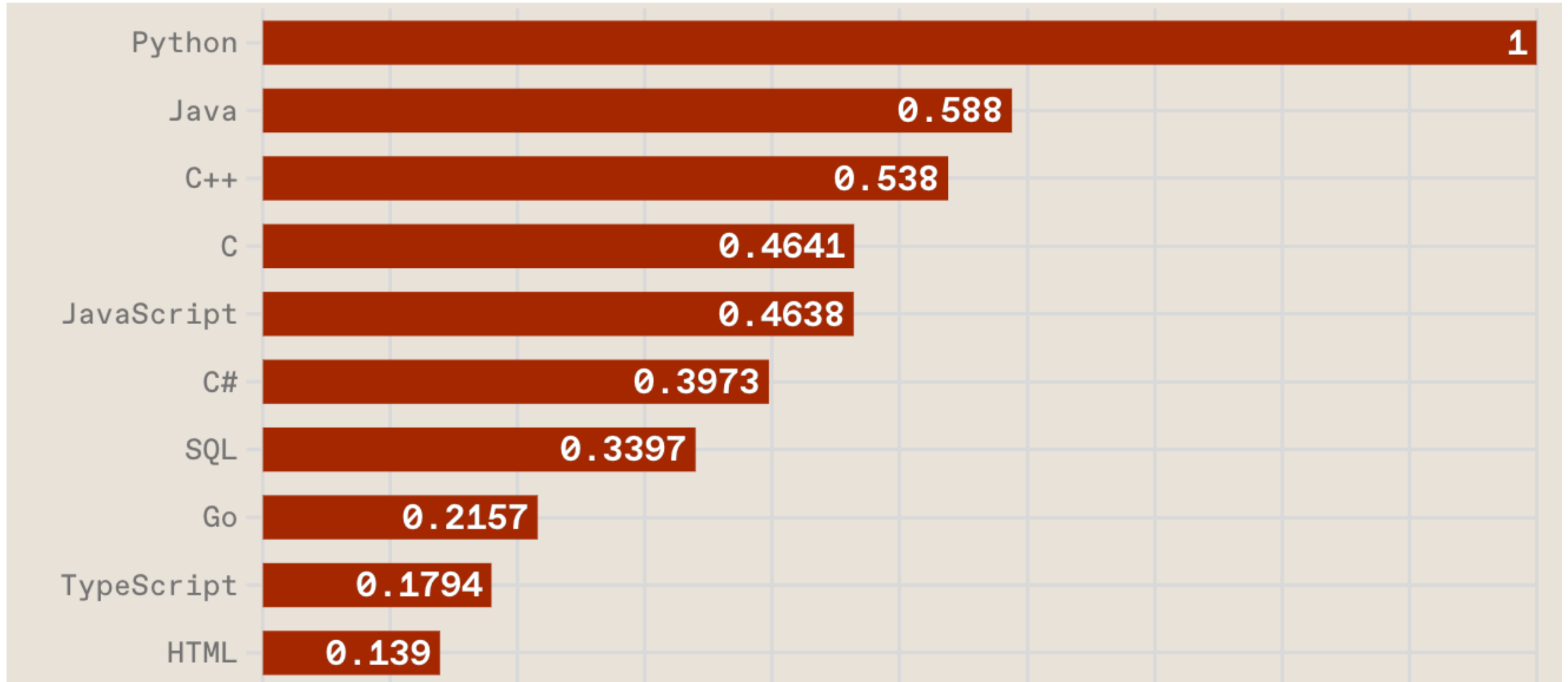
- Python **if else**
  - **if elif else**
  - Booleans: True, False
  - Operators: ==, !=, >, <, >=, <=, and, or, not
- Python **for** Loops
  - **for**
- Python **while** Loops
  - **While**
    - break
    - continue



# Python

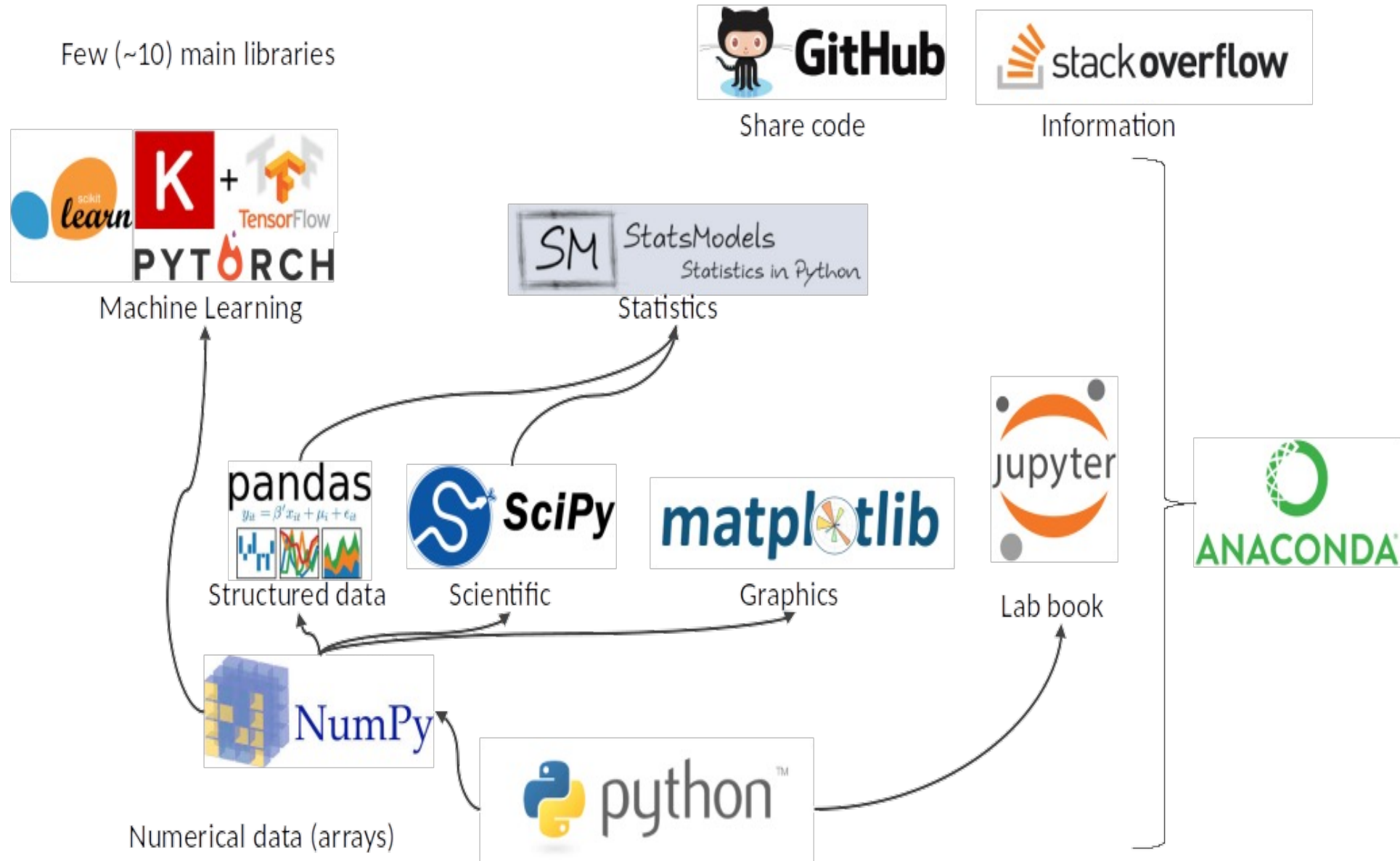
# Programming

# Top Programming Languages

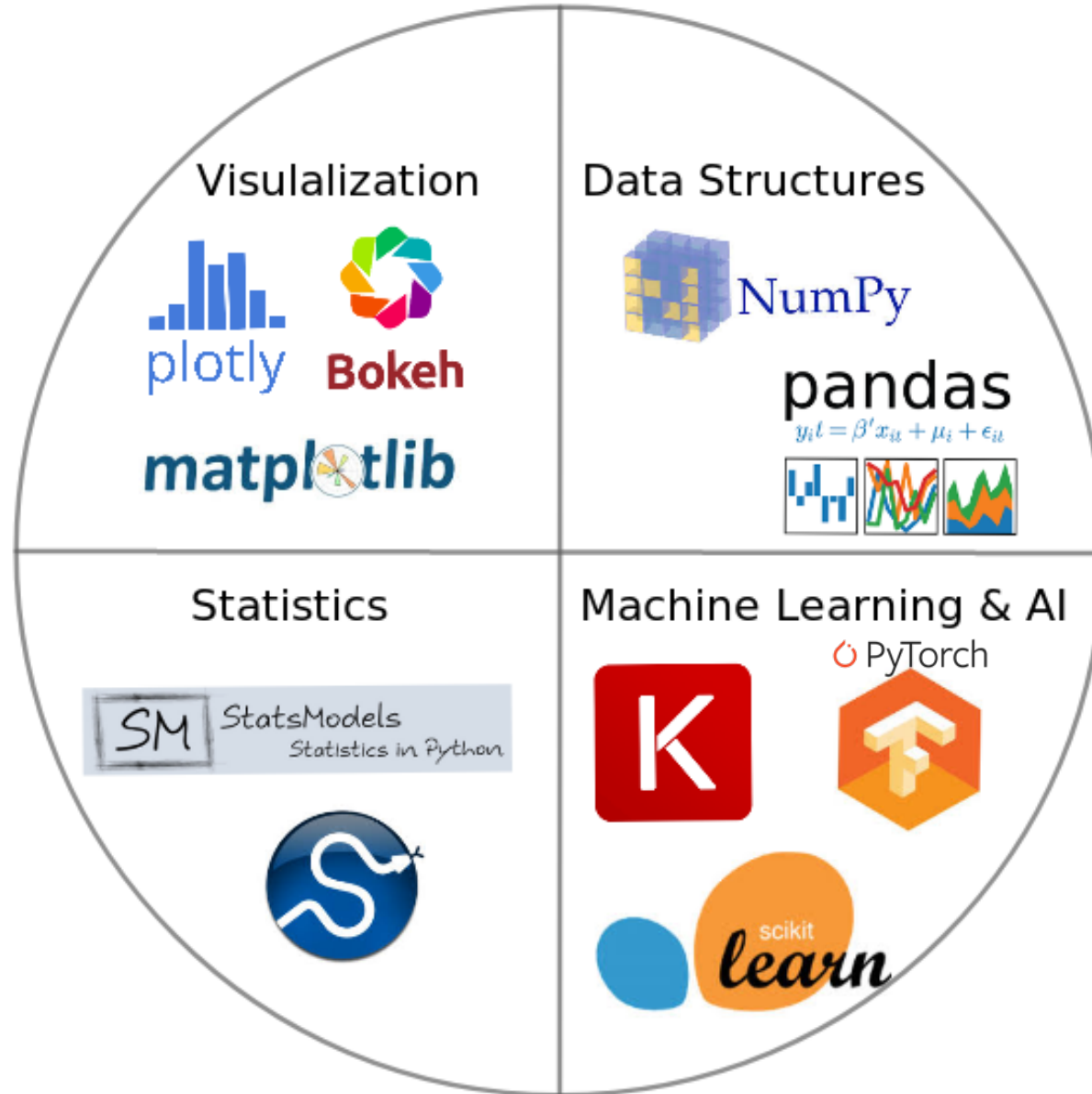


**Python** is an  
interpreted,  
object-oriented,  
high-level  
programming language  
with  
dynamic semantics.

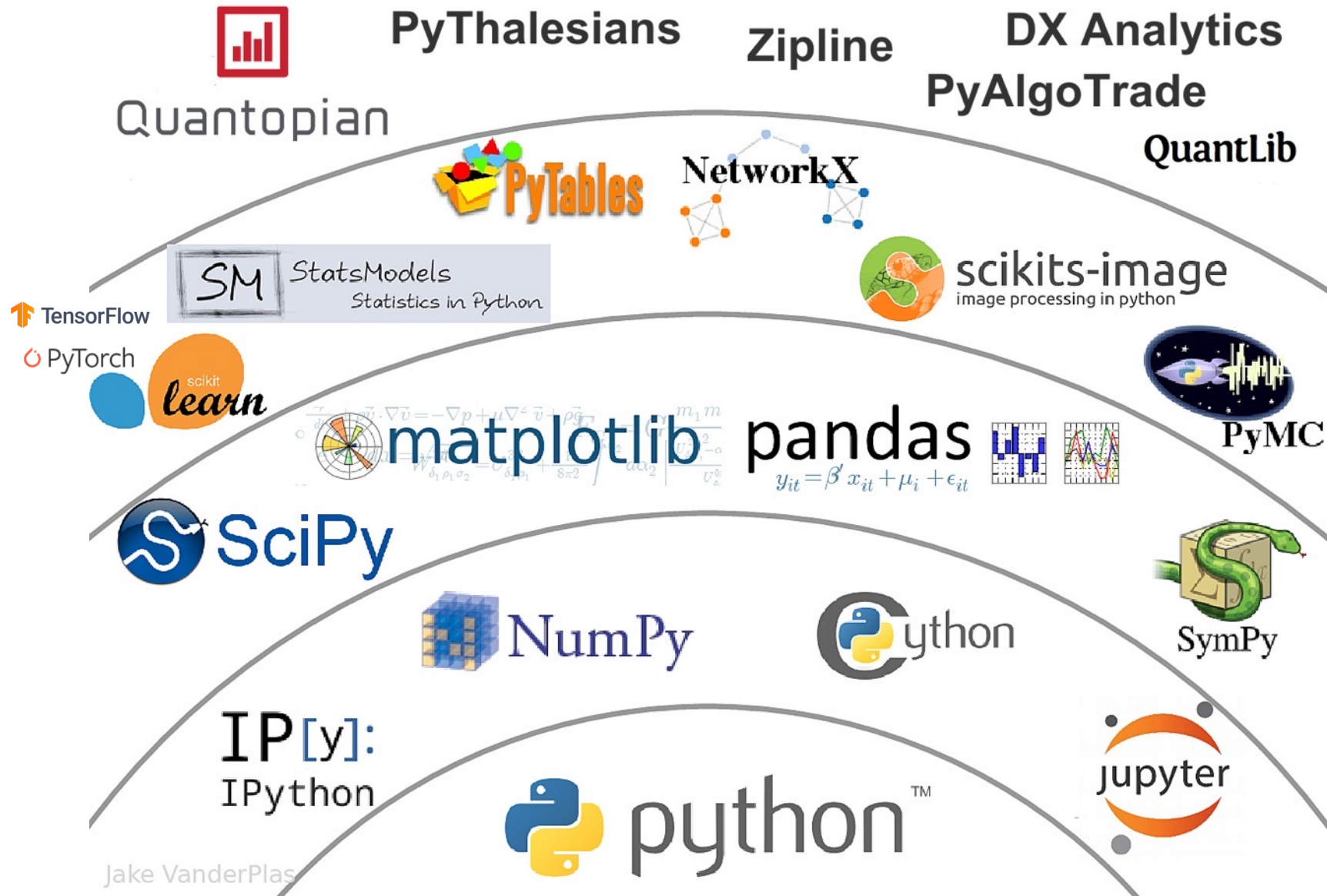
# Python Ecosystem for Data Science



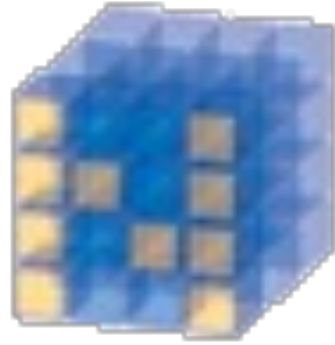
# Python Ecosystem for Data Science



# The Quant Finance PyData Stack



# NumPy



NumPy

Base

**N-dimensional array**

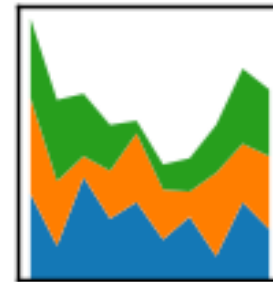
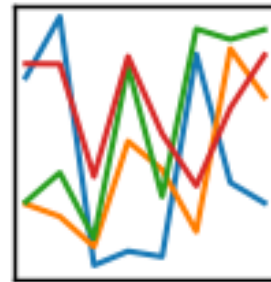
package

**Python**  
**matplotlib**  
**matplotlib**

# Python Pandas

pandas

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



- 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

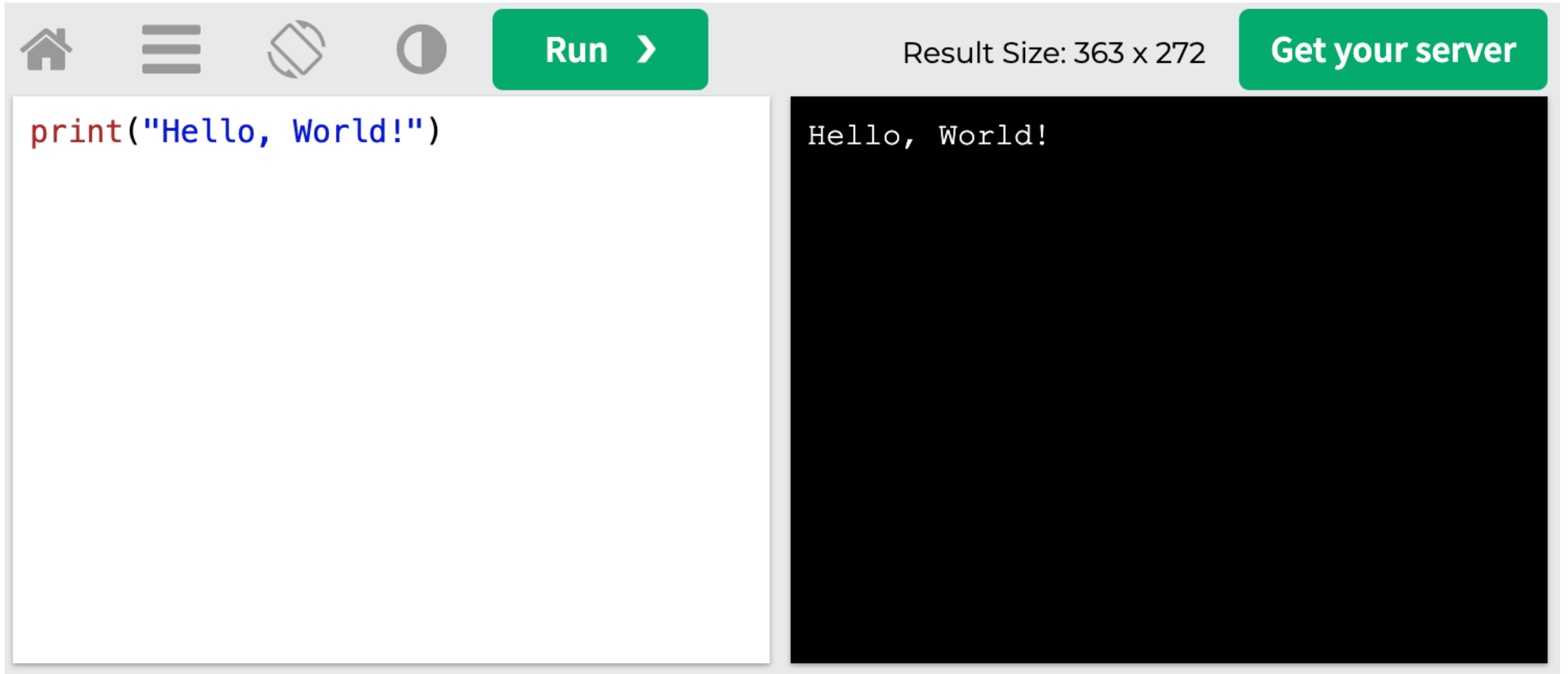
Python is a popular programming language.  
Python can be used on a server to create web applications.

[Start learning Python now »](#)

### Learning by Examples

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

# W3Schools Python: Try Python

A screenshot of the W3Schools Python Try Python interface. The interface has a light gray header with navigation icons (home, menu, refresh, moon) on the left, a green 'Run >' button in the center, and 'Result Size: 363 x 272' and a green 'Get your server' button on the right. Below the header, there is a white code editor on the left containing the Python code `print("Hello, World!")` and a black output window on the right displaying the result 'Hello, World!' in white text.

Run >

Result Size: 363 x 272

Get your server

```
print("Hello, World!")
```

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

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.

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

# 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 includes a search bar, a 'My Drive' dropdown, and a 'Quick Access' section. On the left sidebar, the 'New' button is highlighted with a red dashed box. A dropdown menu is open from 'New', listing options like 'New folder...', 'Upload files...', 'Upload folder...', 'Google Docs', 'Google Sheets', 'Google Slides', and 'More'. The 'More' option is also highlighted with a red dashed box. A second dropdown menu is open from 'More', listing 'Google Forms', 'Google Drawings', 'Google My Maps', 'Google Sites', and 'Connect more apps'. The 'Connect more apps' option is highlighted with a red dashed box. The right side of the interface shows a 'Name' column header and a vertical scrollbar.

# Google Colab

The screenshot shows the Google Drive web interface. A dialog box titled "Connect apps to Drive" is open, displaying a search for "colab". The search results are as follows:

App Name	Description	User Count
ZIP Extractor	Extract ZIP files to Google Drive	307,585 users
Lumin PDF	Beautiful PDF Editor	289,310 users
CloudConvert	Cloud-based conversion tool	373,161 users
Sejda	Merge PDF - Split PDF - Sejda.com	2,131,600 users
DocHub	Edit and Sign PDF Documents	2,131,600 users
Google Forms	Form creation tool	4,803,614 users

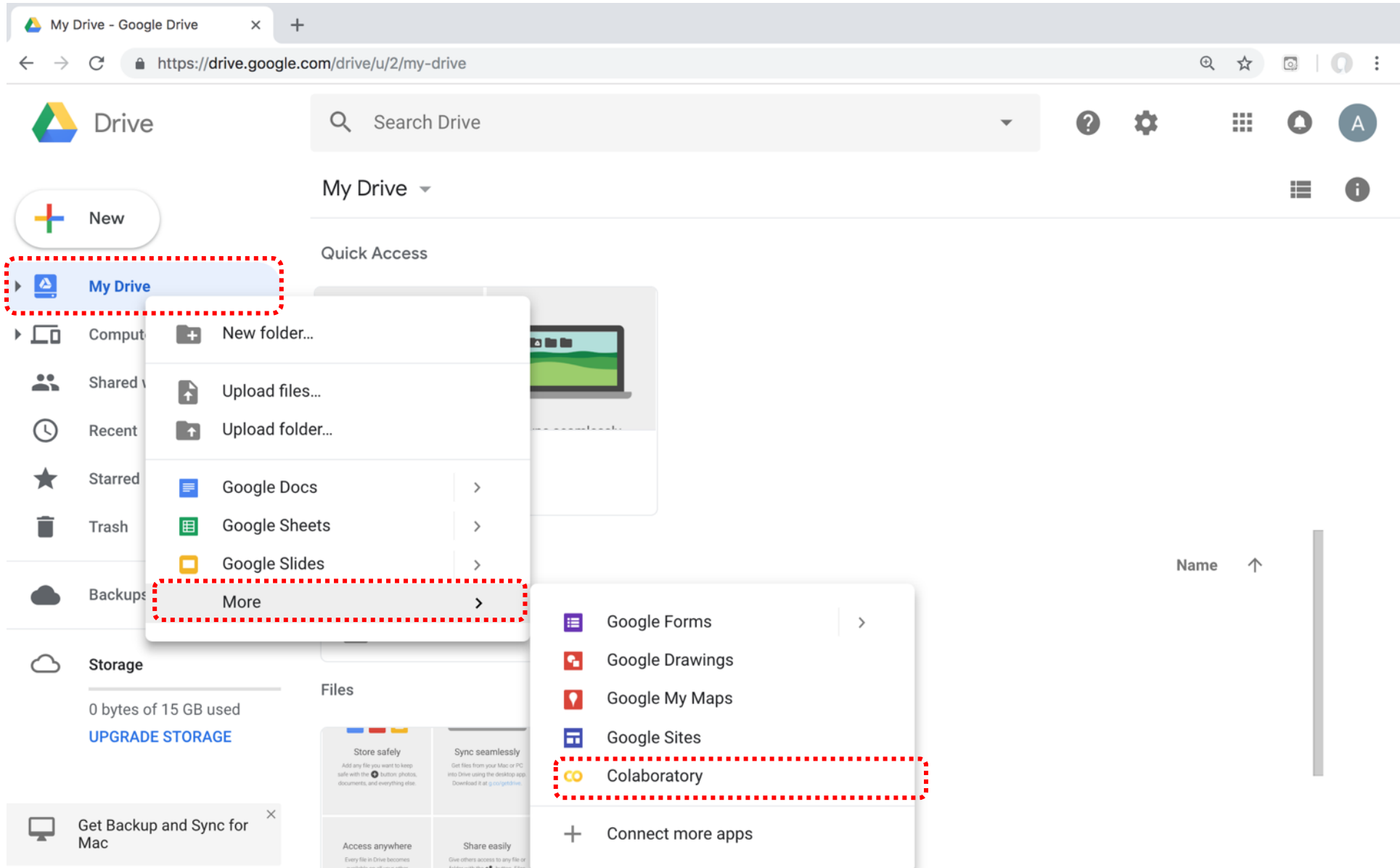
# Google Colab

The image shows a browser window with the Google Drive interface. A modal dialog titled "Connect apps to Drive" is open in the center. The dialog has a search bar containing "colab". Below the search bar, a list of apps is shown. The first app, "Colaboratory", is highlighted with a red dashed border. The app's details include a yellow "CO" logo, the name "Colaboratory", the URL "https://colab.research.google.com", a description: "A data analysis tool that combines code, output, and descriptive text into one collaborative document.", a "Productivity" category, and a 5-star rating with "(195)" reviews. A blue "+ CONNECT" button is located to the right of the app details. The background shows the Google Drive sidebar with options like "New", "My Drive", "Computers", "Shared with me", "Recent", "Starred", "Trash", "Backups", and "Storage". The storage status is "0 bytes of 15 GB used" with an "UPGRADE STORAGE" link. At the bottom, there is a notification for "Get Backup and Sync for Mac".

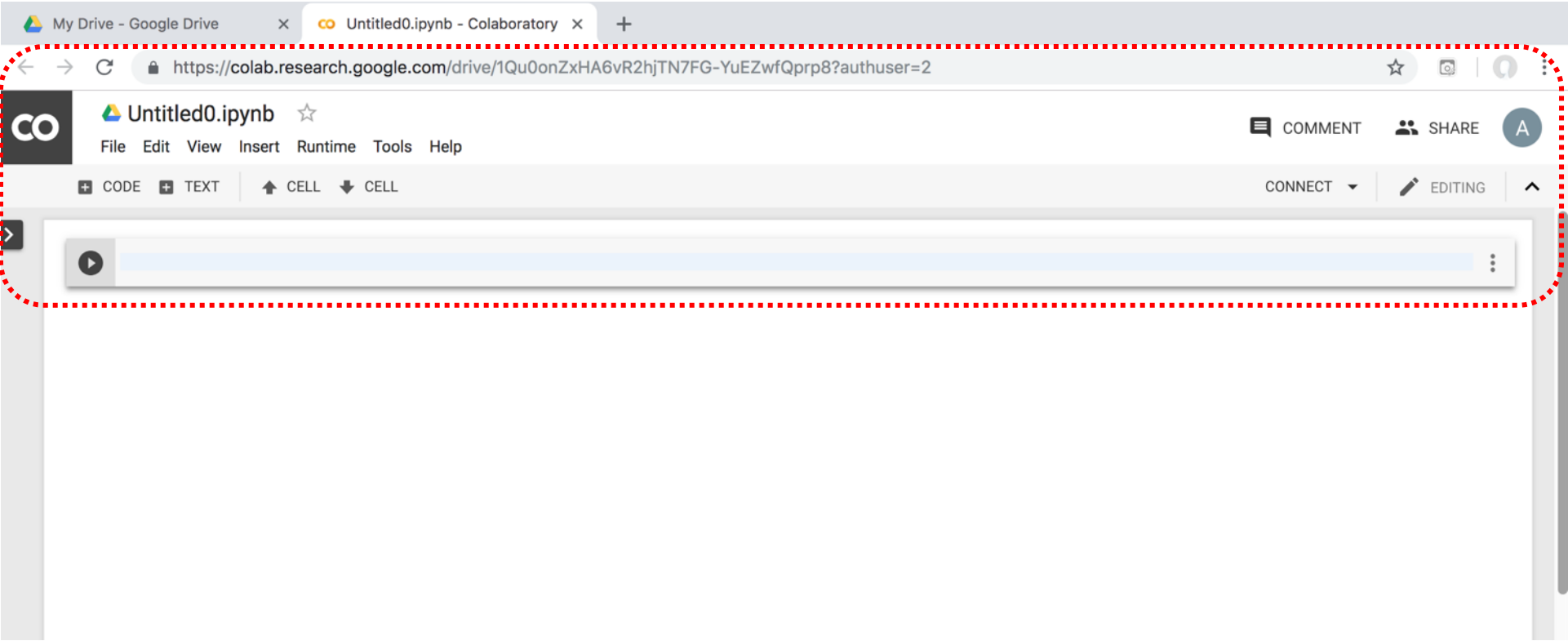
# Connect Colaboratory to Google Drive

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.

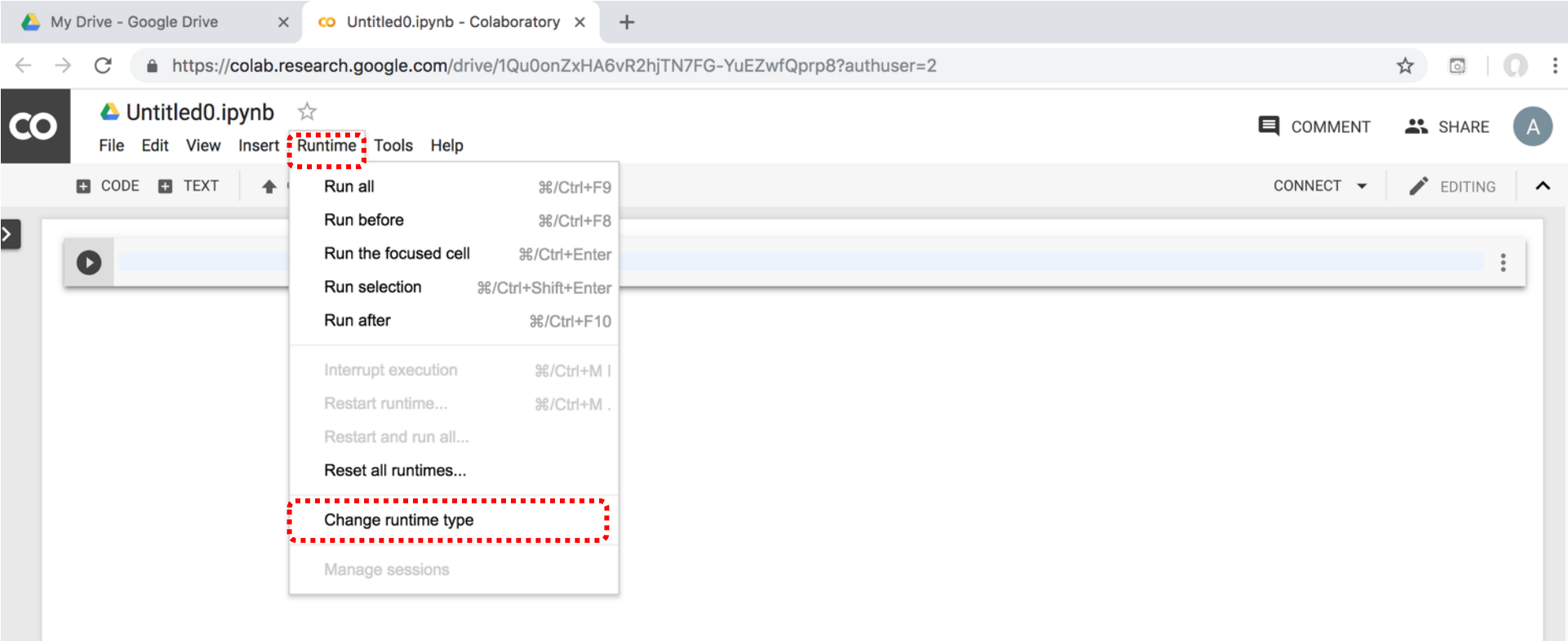
# Google Colab



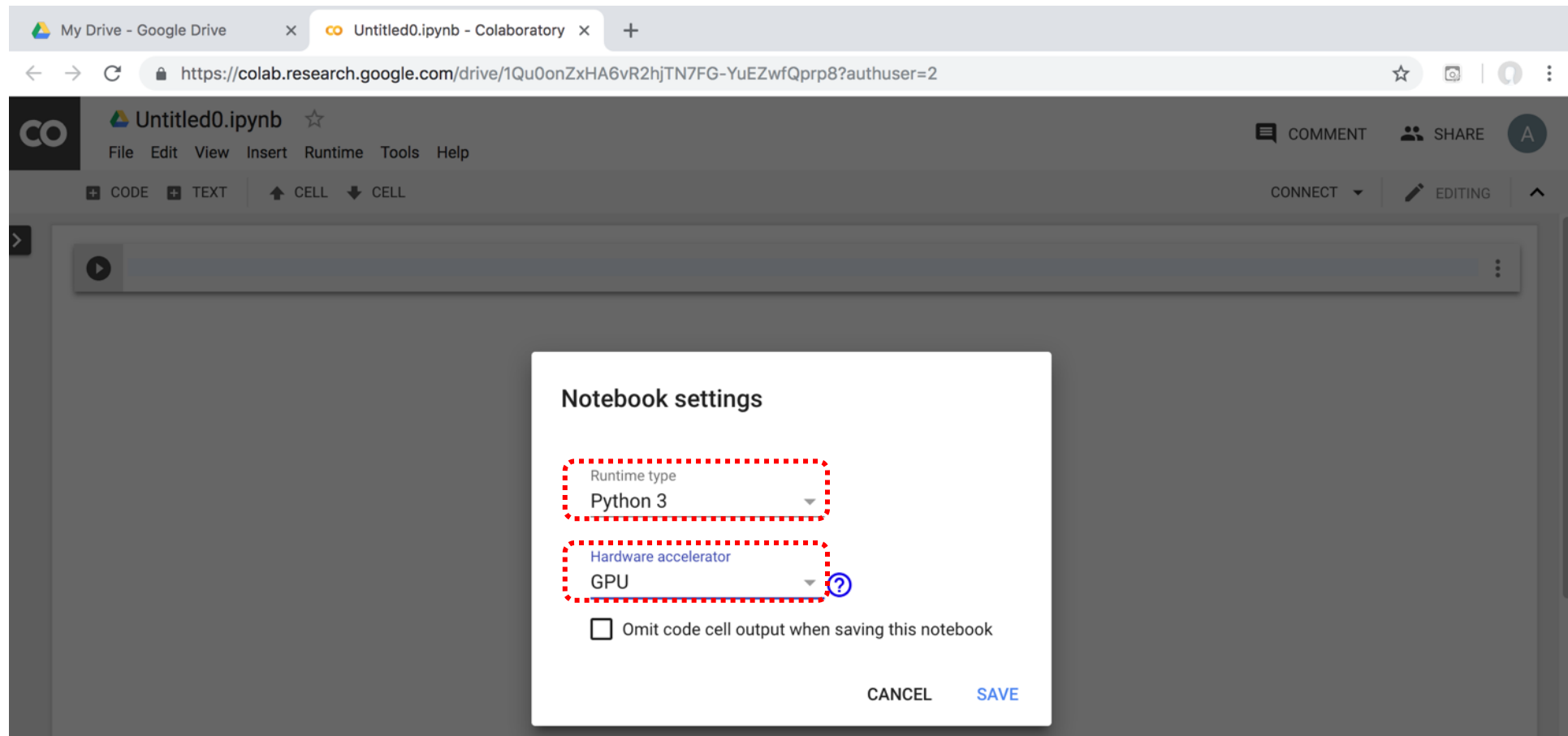
# Google Colab



# Google Colab



# Run Jupyter Notebook Python3 GPU Google Colab



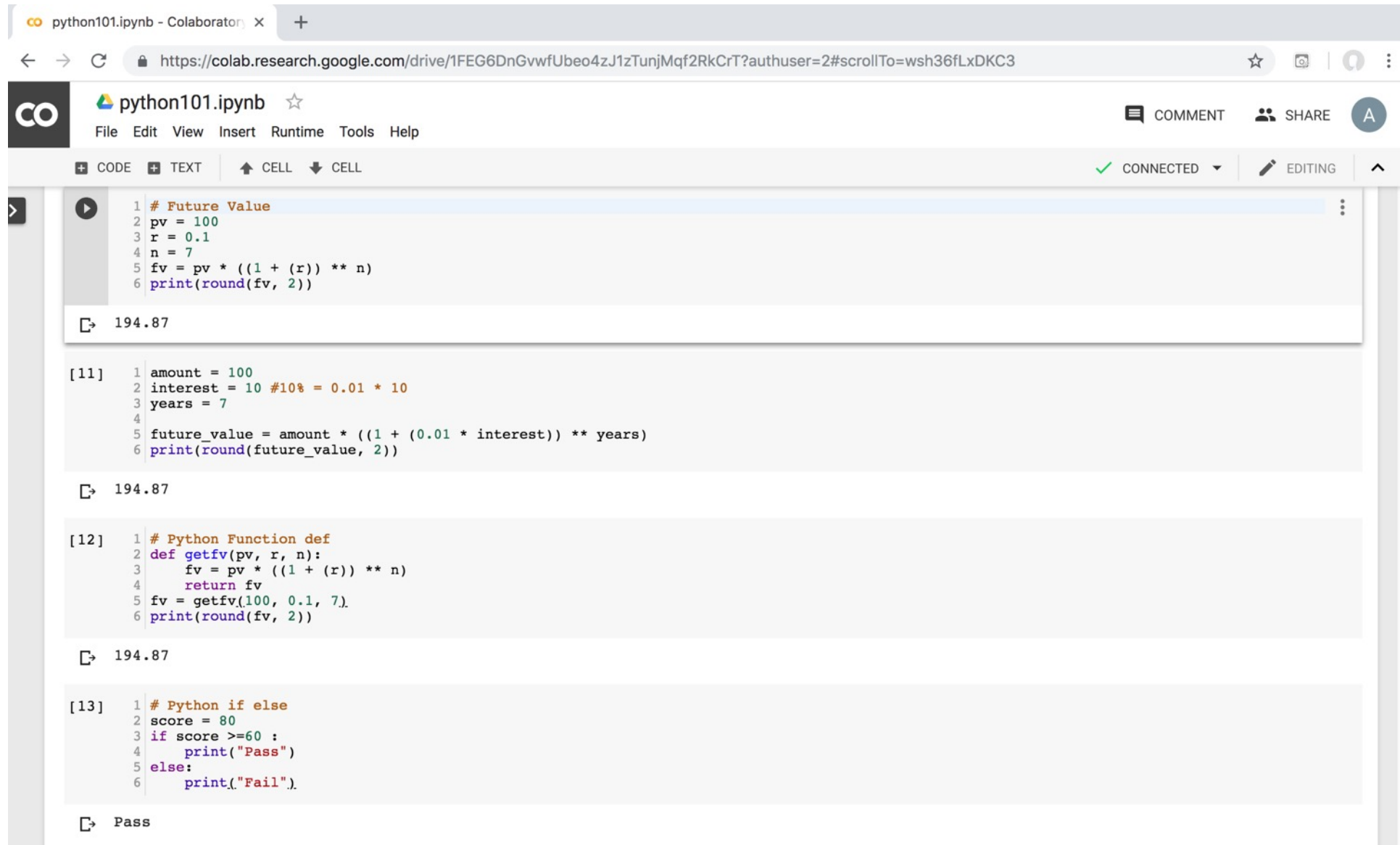
# Google Colab Python Hello World

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



# Python in Google Colab (Python101)

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



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=wsh36fLxDKC3>. The notebook title is "python101.ipynb". The interface includes a menu bar (File, Edit, View, Insert, Runtime, Tools, Help) and a toolbar with options for CODE, TEXT, CELL, and a status indicator showing "CONNECTED" and "EDITING".

The notebook contains four code cells:

- Cell 1:** A code cell with the following Python code:

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

The output is "194.87".
- Cell 2:** A code cell with the following Python code:

```
[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))
```

The output is "194.87".
- Cell 3:** A code cell with the following Python code:

```
[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))
```

The output is "194.87".
- Cell 4:** A code cell with the following Python code:

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

The output is "Pass".

<https://tinyurl.com/aintpupython101>



**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>

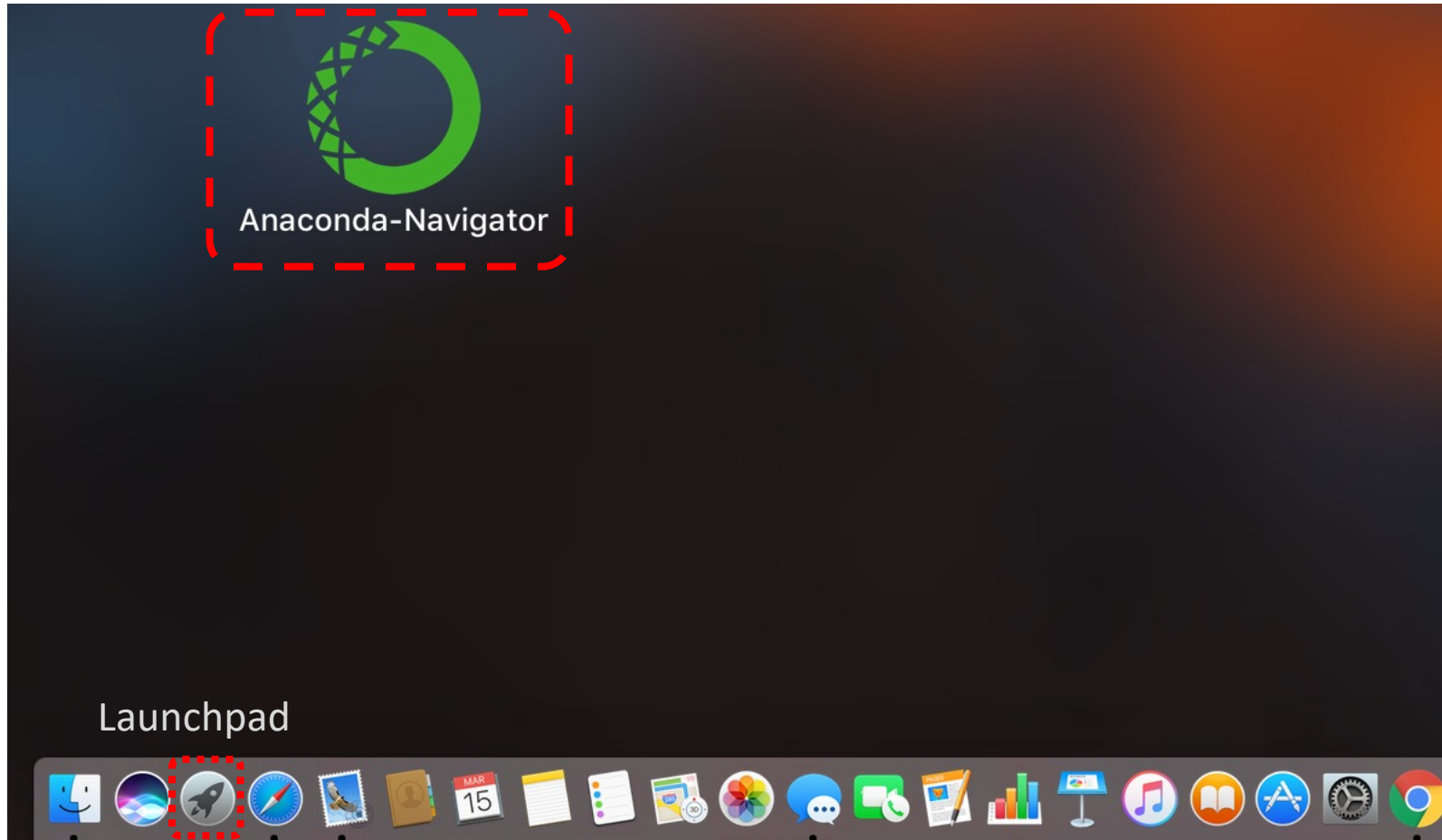




# Python

# 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 header features the "ANACONDA NAVIGATOR" logo on the left and a "Sign in to Anaconda Cloud" button 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 located 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]

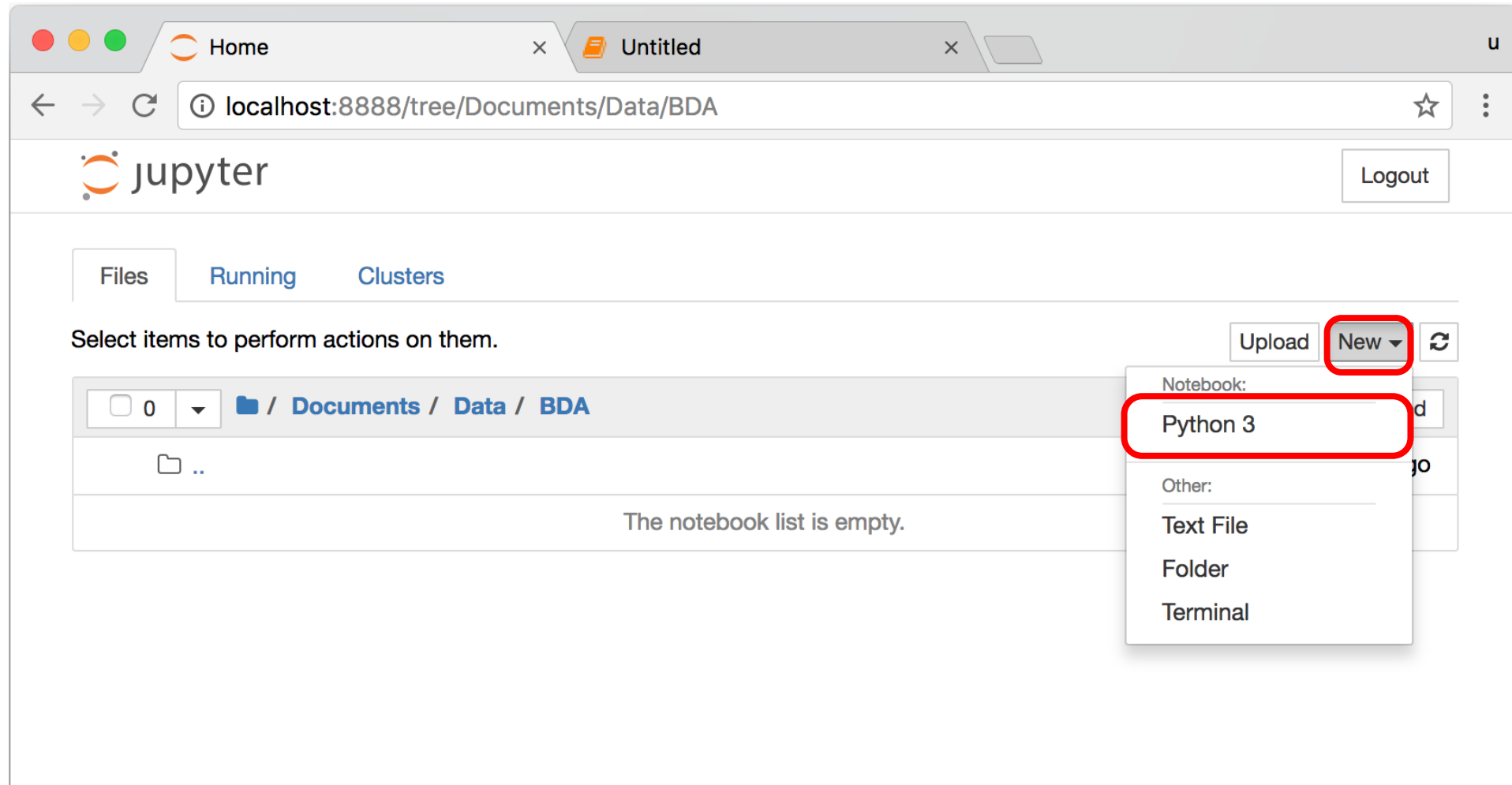
The "jupyter notebook" application card is highlighted with a red dashed border, and its "Launch" button is enclosed in a solid red box.

# 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 in the top navigation bar. Below the navigation bar, there are three tabs: "Files", "Running", and "Clusters", with "Files" being the active tab. A message "Select items to perform actions on them." is displayed above a set of action buttons: "Upload", "New", and a refresh icon. The main content area shows a file browser view for the directory `/ Documents / Data / BDA`. The browser view includes a header with a selection count of "0" and a dropdown arrow, and two columns: "Name" and "Last Modified". A single entry is visible: a folder icon followed by `..` in the "Name" column and "seconds ago" in the "Last Modified" column. Below the file list, a message states "The notebook list is empty." A red dashed box highlights the file browser area.

# 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 at the top. Below the navigation tabs ("Files", "Running", "Clusters"), there is a prompt: "Select items to perform actions on them." To the right of this prompt are buttons for "Upload", "New", and a refresh icon. The "New" button is highlighted with a red circle. A dropdown menu is open, showing options under "Notebook:" and "Other:". The "Python 3" option under "Notebook:" is highlighted with a red rectangle. Other options include "Text File", "Folder", and "Terminal". The file browser below shows the path `/ Documents / Data / BDA` and a message: "The notebook list is empty."

# print("hello, world")

The screenshot shows a web browser window with two tabs: 'Home' and 'HelloWorld'. The address bar shows the URL 'localhost:8888/notebooks/Documents/Data/BDA/HelloWorld.ipynb'. The Jupyter Notebook interface is displayed, with the title 'jupyter HelloWorld (autosaved)' and a 'Logout' button. The menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. The 'Run' button is highlighted with a red box. Below the menu bar, a toolbar contains icons for file operations and execution. The main area shows a code cell with the text 'In [1]: print("hello, world")' and its output 'hello, world'. The code cell is also highlighted with a red box. Below the code cell is an empty input field labeled 'In [ ]:'.





# Python

# Programming

# Foundations of Python Programming

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

# Python Hello World

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

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

# Python Syntax

# comment

```
# comment
```

# Python Syntax

## Indentation

the spaces at the beginning of a code line  
4 spaces

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

# Python Variables

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

# Python Variables

```
x = 2
```

```
y = x + 1
```

# python\_version()

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

Python Version: 3.10.12

# Python Data Types

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

# Python Data Types

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

# Python Data Types

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

# Python Casting

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

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

# Python Numbers

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

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

# Python Arithmetic Operators

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

# Python Basic Operators

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

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

# Python Booleans: True or False

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

# Python BMI Calculator

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

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

Your BMI is: 20.8

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

# How much is your \$100 worth after 7 years?

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

```
fv = 194.87
```

# Future Value

```
# Future Value
pv = 100
r = 0.1
n = 7

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

194.87

# Future Value

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

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

194.87

# Python Data Structures

# Python Data Structures

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

# Python Data Structures

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

# Python Data Types

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

# Python Collections

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

# Python Dictionaries {k:v}

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

# Lists []

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

4  
60  
70  
90

# Lists []

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

# Python List Methods

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

# 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])           10
print(x[1])           20
print(x[2])           30
print(x[-1])          50
```

# Sets {}

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

# Dictionary {key : value}

## Python Dictionary

Key → Value

'EN' → 'English'

'FR' → 'French'

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

English

# Python Data Structures

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

# Python Control Logic and Loops

# Python Control Logic and Loops

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

# Python **if...else**

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

# Python Conditions and **If** statements

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

# Python Comparison Operators

Operator	Name	Example
<code>==</code>	Equal	<code>x == y</code>
<code>!=</code>	Not equal	<code>x != y</code>
<code>&gt;</code>	Greater than	<code>x &gt; y</code>
<code>&lt;</code>	Less than	<code>x &lt; y</code>
<code>&gt;=</code>	Greater than or equal to	<code>x &gt;= y</code>
<code>&lt;=</code>	Less than or equal to	<code>x &lt;= y</code>

# Python Logical Operators

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

# Python if

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

# Python if else

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

# Python if elif else

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

# Python if elif else

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

# Python **for** Loops

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

1  
2  
3  
4  
5

# Python **for** loops

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

# Python **while** Loops

- **while**
  - **break**
  - **continue**

# Python **while** loops

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

# Summary

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

# References

- Wes McKinney (2022), "Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter", 3rd Edition, O'Reilly Media.
- Aurélien Géron (2023), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 3rd Edition, O'Reilly Media.
- Steven D'Ascoli (2022), Artificial Intelligence and Deep Learning with Python: Every Line of Code Explained For Readers New to AI and New to Python, Independently published.
- Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson.
- Denis Rothman (2024), Transformers for Natural Language Processing and Computer Vision - Third Edition: Explore Generative AI and Large Language Models with Hugging Face, ChatGPT, GPT-4V, and DALL-E 3, 3rd ed. Edition, Packt Publishing
- Ben Auffarth (2023), Generative AI with LangChain: Build large language model (LLM) apps with Python, ChatGPT and other LLMs, Packt Publishing.
- Varun Grover, Roger HL Chiang, Ting-Peng Liang, and Dongsong Zhang (2018), "Creating Strategic Business Value from Big Data Analytics: A Research Framework", Journal of Management Information Systems, 35, no. 2, pp. 388-423.
- Junliang Wang, Chuqiao Xu, Jie Zhang, and Ray Zhong (2022). "Big data analytics for intelligent manufacturing systems: A review." Journal of Manufacturing Systems 62 (2022): 738-752.
- Ramesh Sharda, Dursun Delen, and Efraim Turban (2017), Business Intelligence, Analytics, and Data Science: A Managerial Perspective, 4th Edition, Pearson
- Python Programming, <https://pythonprogramming.net/>
- Python, <https://www.python.org/>
- Python Programming Language, <http://pythonprogramminglanguage.com/>
- Numpy, <http://www.numpy.org/>
- Pandas, <http://pandas.pydata.org/>
- Skikit-learn, <http://scikit-learn.org/>
- W3Schools Python, <https://www.w3schools.com/python/>
- Learn Python, <https://www.learnpython.org/>
- Google's Python Class, <https://developers.google.com/edu/python>
- Harvard University (2024), CS50x 2024 - Lecture 6 - Python, <http://www.youtube.com/watch?v=EHiORDZ31VA>
- Min-Yuh Day (2024), Python 101, <https://tinyurl.com/aintpupython101>