Sustainability and ESG Data Analytics

NTPU
國立臺北大學
National Taipei University

Data Science for Sustainability and ESG

1131ESGDA03
MBA, IM, NTPU (M5265) (Fall 2024)
Wed 2, 3, 4 (9:10-12:00) (B3F17)

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

aws educate | Cloud Ambassador
2020 Cohort

aws academy Accredited Educator
aws certified Cloud Practitioner
aws certified Solutions Architect Associate

Min-Yuh Day, Ph.D,
Professor

Institute of Information Management, National Taipei University

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

2024-09-25

Syllabus

Week	Date	Subject/Topics
1	2024/09/11	Introduction Sustainability and ESG Data Analytics
2	2024/09/18	Environmental, Social, and Governance (ESG) in Net-Zero Digital Transformation
3	2024/09/25	Data Science for Sustainability and ESG
4	2024/10/02	Case Study on Sustainability and ESG Data Analytics I
5	2024/10/09	Web 3.0 and Big Data Analysis in Fintech, Green and Sustainable Finance
6	2024/10/16	Task Force on Climate-Related Financial Disclosures (TCFD) and En-Roads Interactive

Syllabus

Week	Date	Subject/Topics
7	2024/10/23	ESG Data Gathering, Analysis, and Visualization
8	2024/10/30	Midterm Project Report
9	2024/11/06	Self-Learning
10	2024/11/13	ESG Data Reporting; Corporate Sustainability Reports
11	2024/11/20	ESG Data Verification
12	2024/11/27	Case Study on Sustainability and ESG Data Analytics II

Syllabus

Week	Date	Subject/Topics
13	2024/12/04	Artificial Intelligence of things (AIoT) in ESG and Sustainability Applications
14	2024/12/11	Generative AI for ESG Rating and Reporting Generation
15	2024/12/18	Final Project Report I
16	2024/12/25	Final Project Report II

Data Science for Sustainability and ESG

Outline

- Data Science and Sustainability
- Data Collection and Analysis for Sustainability
- Implementing Data-Driven Sustainability Strategies
- ESG Data Science Foundation with Python

Sustainability and ESG Data Analytics

Data Science for Sustainability and ESG: Transforming Decision-Making for a Better Future

Data Science for Sustainability and ESG: Foundations & Frameworks

Data Science for Sustainability and ESG

Data Science for Sustainability and ESG

- **Data Science and Sustainability**
- **Data Collection and Analysis for Sustainability**
- **Implementing Data-Driven Sustainability Strategies**

Fundamentals of Data Science

- **Definition of Data Science:**
  - **Interdisciplinary field using scientific methods, processes, algorithms and systems to extract knowledge and insights from structured and unstructured data.**
- **Key components:**
  - **Statistics, Machine Learning, Data Engineering**

Data Science Process

1. Business Understanding
2. Data Understanding
3. Data Preparation
4. Modeling
5. Evaluation
6. Deployment

Data Science Essential tools

- **Python**
- **R**
- **SQL**
- **Tableau**
- **Power BI**

Sustainability and ESG

- **Sustainability: Meeting present needs without compromising future generations**
- **ESG: Environmental, Social, and Governance**
  - **Environmental: Climate change, resource depletion, waste, pollution, deforestation**
  - **Social: Human rights, labor standards, workplace safety, community relations**
  - **Governance: Board diversity, executive compensation, ethics, transparency**

Sustainability and ESG: Business case

- **Risk management**
- **Cost savings**
- **Innovation**
- **Brand value**
- **Investor attraction**

Intersection of Data Science with Sustainability and ESG

- **Data-driven decision making for sustainability initiatives**
- **Predictive analytics for environmental impact**
- **Machine learning for optimizing resource usage**
- **Big data analysis for social impact assessment**
- **AI-powered governance risk management**

The Interconnectedness of Data, Sustainability & ESG

- **Sustainability encompasses environmental, social, and governance concerns.**
- **ESG provides a framework for measuring and reporting sustainability performance.**
- **Data science offers the tools to collect, analyze, and use ESG data for decision-making.**

ESG:

Environmental

Social

Governance

Sustainable Development Goals (SDGs)

Source: https://sdgs.un.org/goals

Sustainable Development Goals (SDGs) and 5P

Partnership

Peace

Prosperity

People

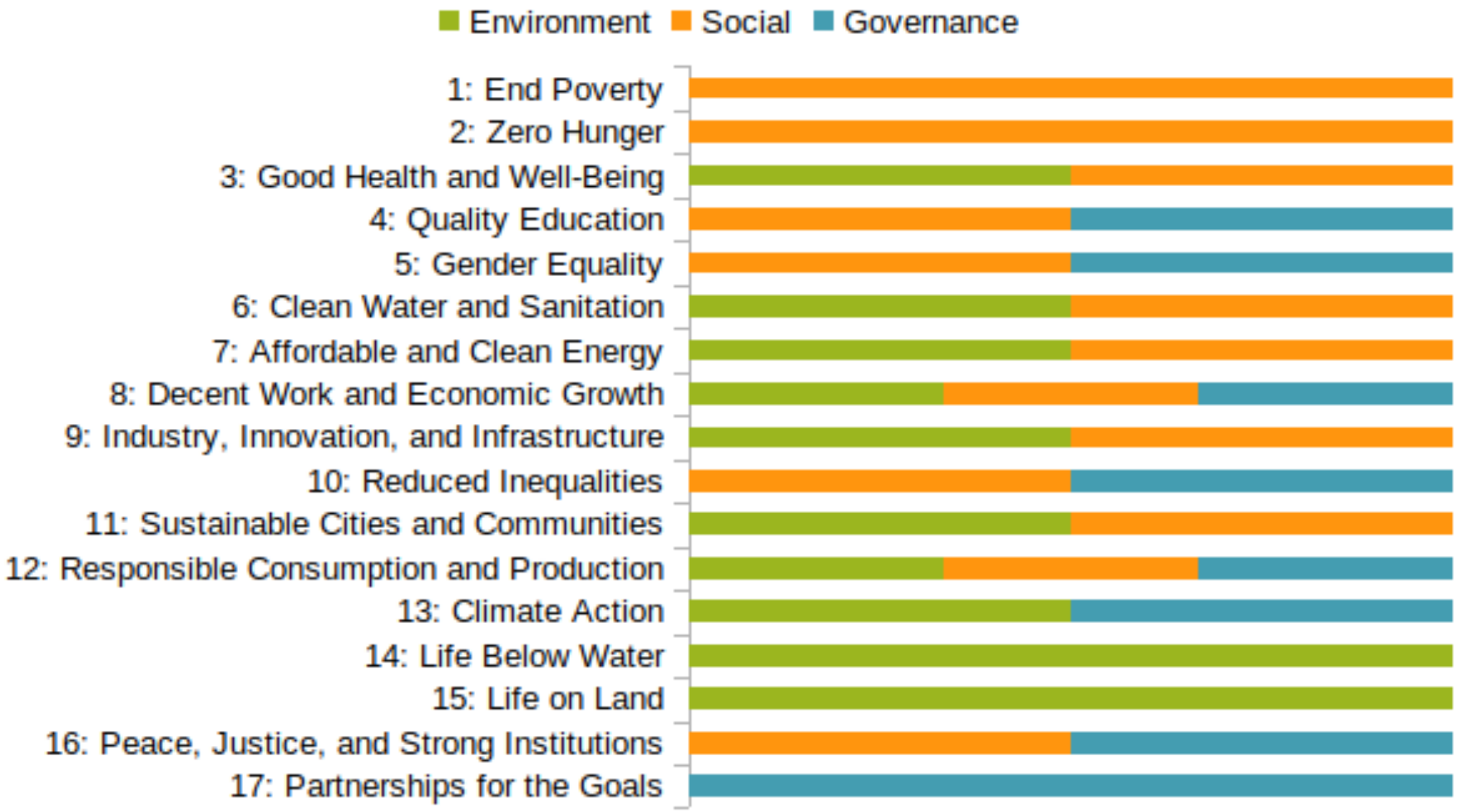
Planet

Source: Folke, Carl, Reinette Biggs, Albert V. Norström, Belinda Reyers, and Johan Rockström. "Social-ecological resilience and biosphere-based sustainability science."Ecology and Society 21, no. 3 (2016).

ESG to 17 SDGs

Source: Henrik Skaug Sætra (2021) "A Framework for Evaluating and Disclosing the ESG Related Impacts of AI with the SDGs." Sustainability 13, no. 15 (2021): 8503.

ESG to 17 SDGs



Source: https://sustainometric.com/esg-to-sdgs-connected-paths-to-a-sustainable-future/

Data Collection and Analysis for Sustainability

Data Collection and Analysis for Sustainability

- **Types of sustainability data and collection methods**
- **Data quality, preparation, and analysis techniques**
- **Case studies of data-driven sustainability initiatives**

Types of Data Relevant to Sustainability and ESG

- **Environmental**: GHG emissions, energy usage, water consumption, waste generation
- **Social**: Employee demographics, health and safety incidents, community engagement metrics
- **Governance**: Board composition, executive compensation, ethical violations
- **Financial**: ESG-related investments, sustainable product revenues, carbon pricing

Understanding ESG Data

- **Quantitative data: Measurable metrics (e.g., carbon emissions, employee diversity)**
- **Qualitative data: Text-based information (e.g., policies, news articles)**
- **Structured data: Organized in databases**
- **Unstructured data: Requires processing (e.g., social media)**

Challenges of ESG Data

- Lack of standardization:
  - Different reporting frameworks and metrics
- Reliability:
  - Concerns about "greenwashing" and accuracy
- Availability:
  - Data gaps, especially for smaller companies and certain sectors

Data Collection Methods and Sources

- **Internal:**
  - **ERP systems, HR databases, facility management systems**
- **External:**
  - **Government databases, NGO reports, industry benchmarks**

Data Collection Methods and Sources

- IoT and sensor data
  - Smart meters, environmental sensors
- Satellite and geospatial data
  - Land use changes, deforestation monitoring
- Social media and web scraping
  - Brand sentiment, consumer trends
- Surveys and stakeholder engagement
  - Employee satisfaction, community feedback

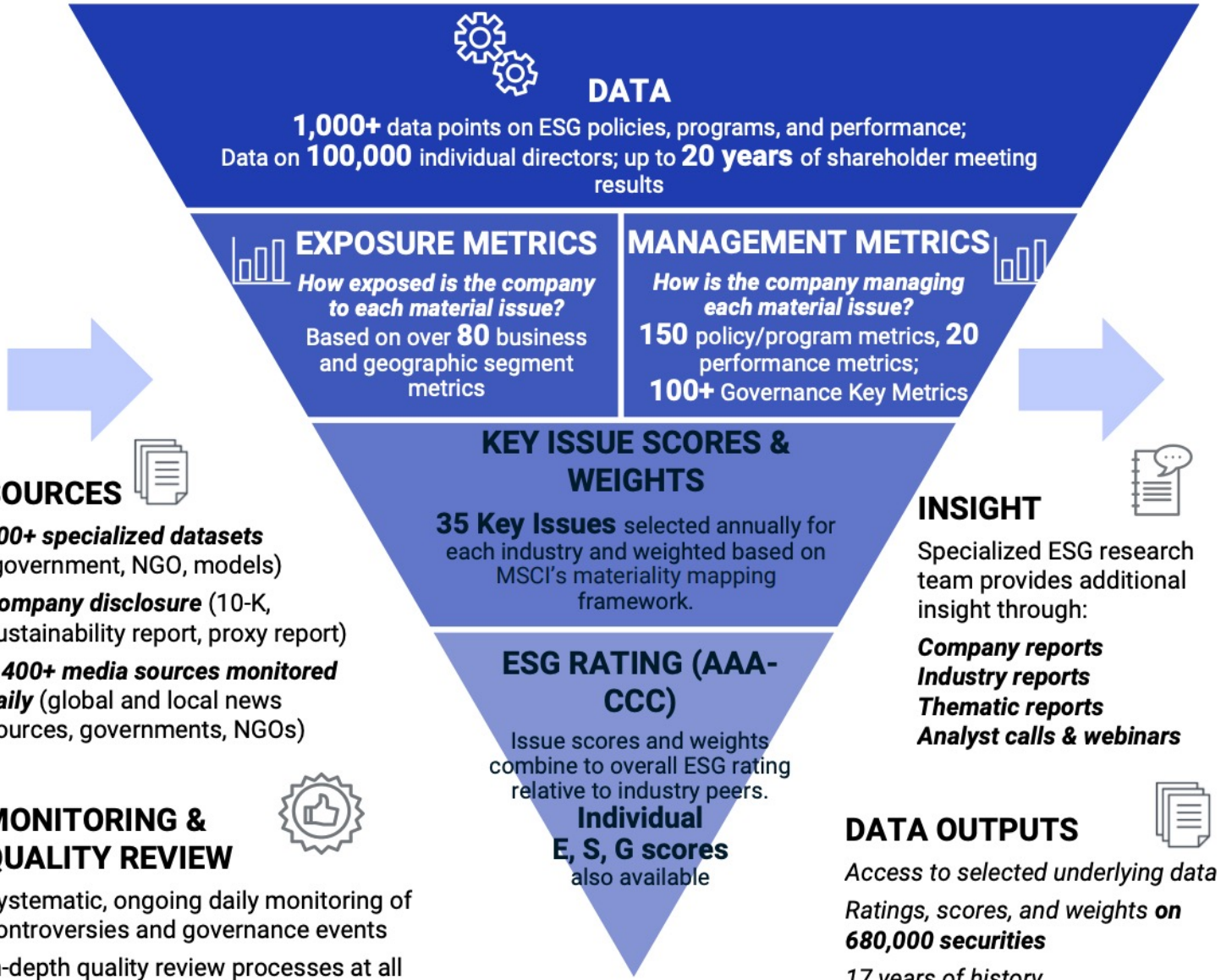
Data Quality and Preparation

- Ensuring data accuracy:
  - Validation checks, cross-referencing
- Data cleaning:
  - Handling outliers, formatting inconsistencies
- Addressing missing data:
  - Imputation techniques, understanding data gaps
- Data integration:
  - Combining multiple sources, resolving conflicts
- Standardization:
  - Aligning with industry standards (e.g., GRI, SASB)

Data Analysis Techniques for Sustainability

- **Descriptive analytics:**
  - **KPI tracking, trend analysis**
- **Diagnostic analytics:**
  - **Root cause analysis of sustainability issues**
- **Predictive analytics:**
  - **Forecasting future sustainability performance**
- **Prescriptive analytics:**
  - **Optimization of resource allocation**
- **Machine learning applications:**
  - **Anomaly detection, pattern recognition in sustainability data**

MSCI ESG Rating Framework



Source: https://www.msci.com/documents/1296102/21901542/ESG-Ratings-Methodology-Exec-Summary.pdf

MSCI ESG Key Issue Hierarchy

3 Pillars	10 Themes	35 ESG Key Issues	
Environment	**Climate Change**	Carbon Emissions	Financing Environmental Impact
		Product Carbon Footprint	Climate Change Vulnerability
	Natural Capital	Water Stress	Raw Material Sourcing
		Biodiversity & Land Use	
	Pollution & Waste	Toxic Emissions & Waste	Electronic Waste
		Packaging Material & Waste	
	Environmental Opportunities	Opportunities in Clean Tech	Opportunities in Renewable Energy
		Opportunities in Green Building	
Social	**Human Capital**	Labor Management	Human Capital Development
		Health & Safety	Supply Chain Labor Standards
	Product Liability	Product Safety & Quality	Privacy & Data Security
		Chemical Safety	Responsible Investment
		Consumer Financial Protection	Health & Demographic Risk
	Stakeholder Opposition	Controversial Sourcing	
		Community Relations	
	Social Opportunities	Access to Communications	Access to Health Care
		Access to Finance	Opportunities in Nutrition & Health
Governance	**Corporate Governance**	Ownership & Control	Pay
		Board	Accounting
	Corporate Behavior	Business Ethics	
		Tax Transparency	

Source: https://www.msci.com/documents/1296102/21901542/ESG-Ratings-Methodology-Exec-Summary.pdf

Data Preparation & Cleaning: Preparing ESG Data for Analysis

- **Handling missing values:**
  - **Imputation, deletion, etc.**
- **Detecting and addressing outliers.**
- **Ensuring data consistency (formatting, units).**
- **Feature engineering:**
  - **Creating new metrics based on raw ESG data.**

Exploratory Data Analysis (EDA) & Visualization: Uncovering Patterns in ESG Data

- **Exploratory Data Analysis (EDA)**
  - **Using statistical summaries and visualizations.**
- **Identifying trends, correlations, and unusual patterns in ESG data.**
- **Storytelling with data:**
  - **Effective visualizations for ESG insights.**

ESG Predictive Modeling: Forecasting with ESG Data

- **Regression**
  - **Predicting continuous ESG outcomes (e.g., emissions)**
- **Classification:**
  - **Categorizing companies (e.g., high/low ESG risk)**
- **Time Series:**
  - **Forecasting ESG metrics over time.**

Natural Language Processing (NLP) for Sentiment Analysis: Sentiment Analysis for ESG Insights

- **Natural Language Processing (NLP):**
  - **Enabling computers to understand human language.**
- **Sentiment analysis:**
  - **Determining the emotional tone of text (positive, negative, neutral)**
- **Applications for ESG:**
  - **Tracking public opinion, identifying ESG risks and opportunities.**

Analyzing Sustainability Reports

- Company sustainability report
- Identify which framework was used, key ESG metrics, areas of strength and weakness
  - How the company presents its ESG data and whether it aligns with the ESG key framework identified

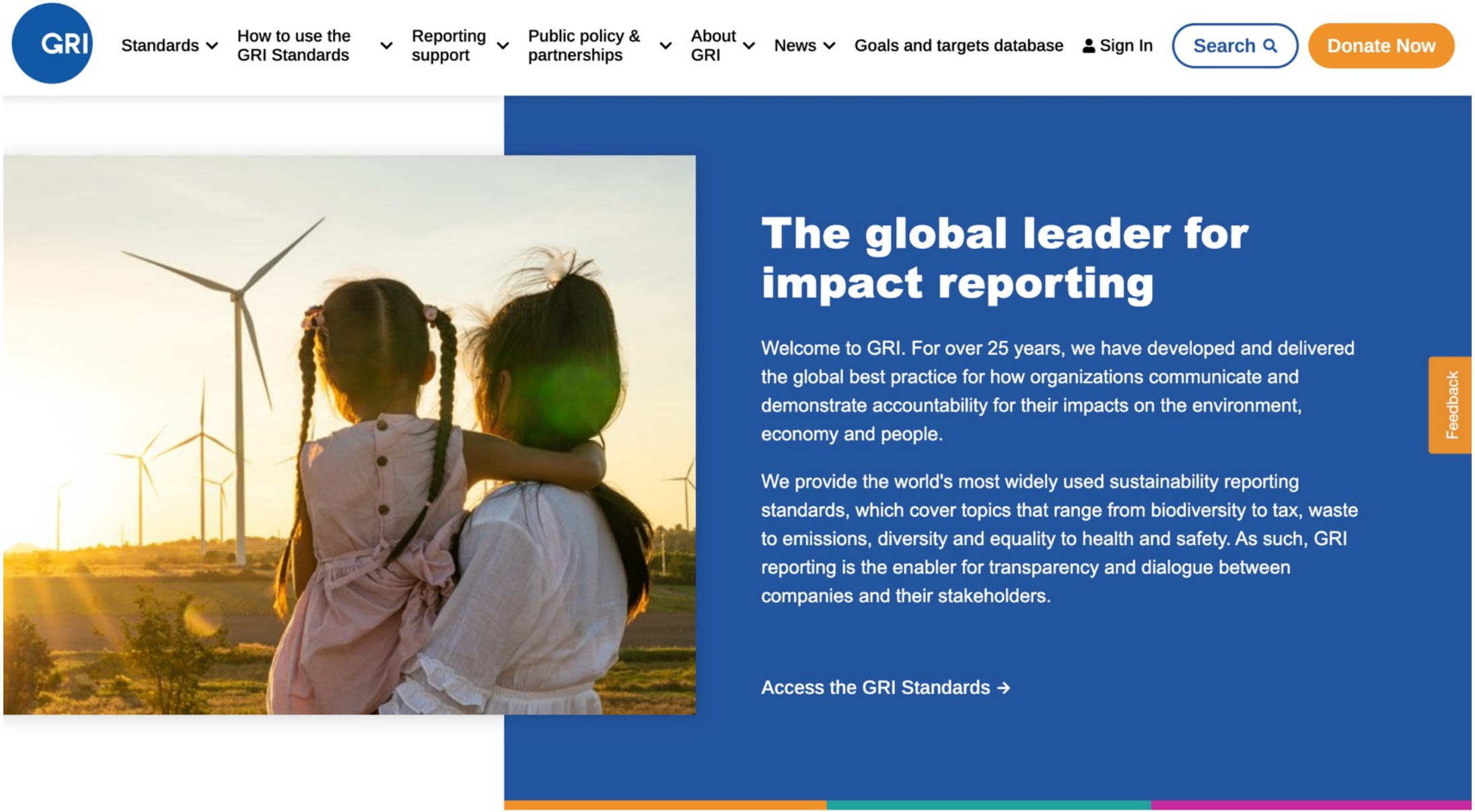
ESG Key Frameworks

- **Global Reporting Initiative (GRI):**
  - **Comprehensive sustainability standards**
- **Sustainability Accounting Standards Board (SASB):**
  - **Industry-specific metrics**
- **UN Sustainable Development Goals (SDGs):**
  - **Global goals for 2030**

Case Studies:
Data-Driven Sustainability Initiatives

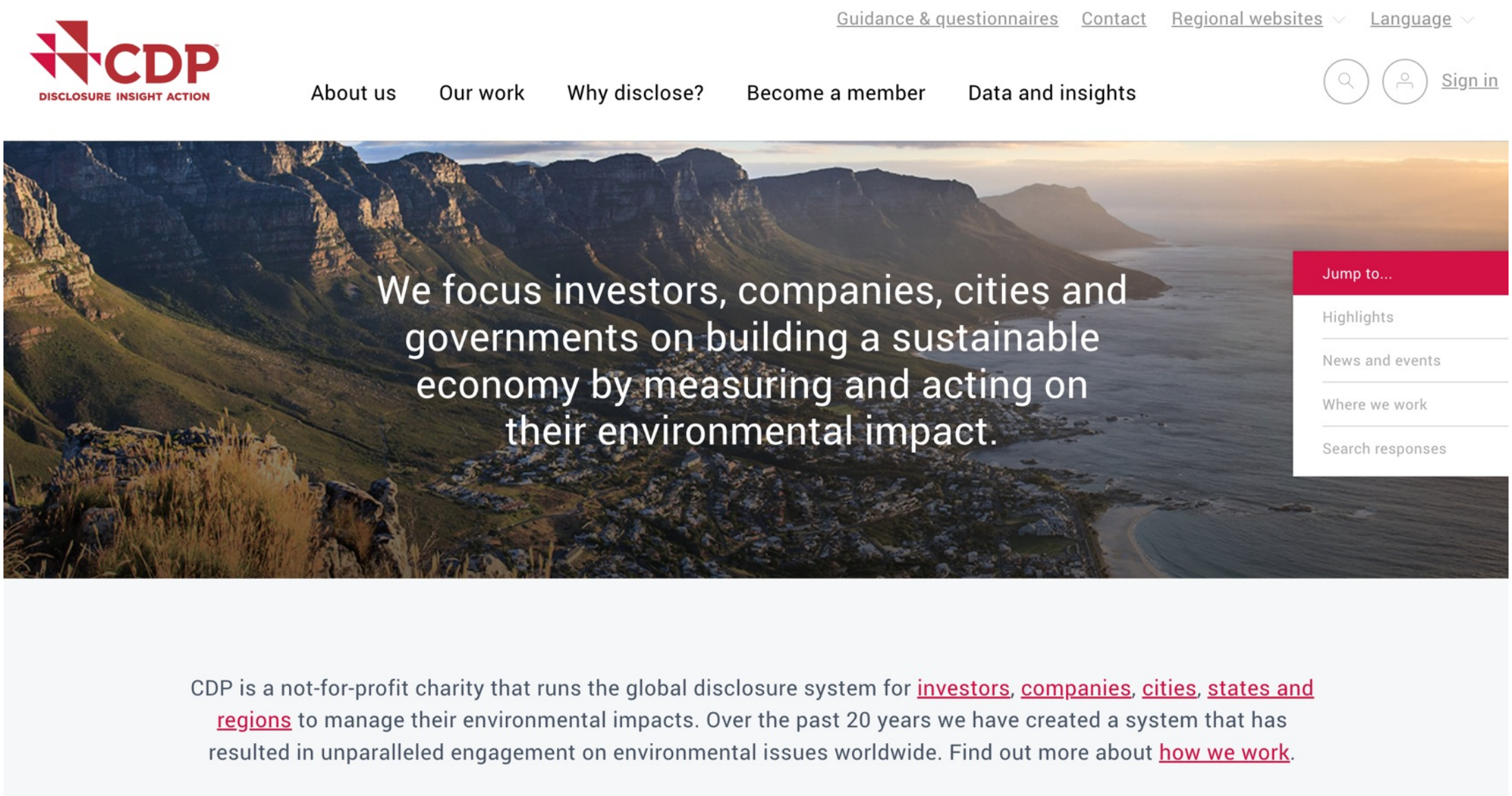
- **Carbon footprint reduction:**
  - **Using ML to optimize logistics routes**
- **Supply chain optimization:**
  - **Predictive analytics for sustainable sourcing**
- **Energy efficiency improvement:**
  - **IoT-driven smart building management**
- **Diversity and inclusion analysis:**
  - **NLP on job descriptions and employee feedback**

GRI (Global Report Initiative)



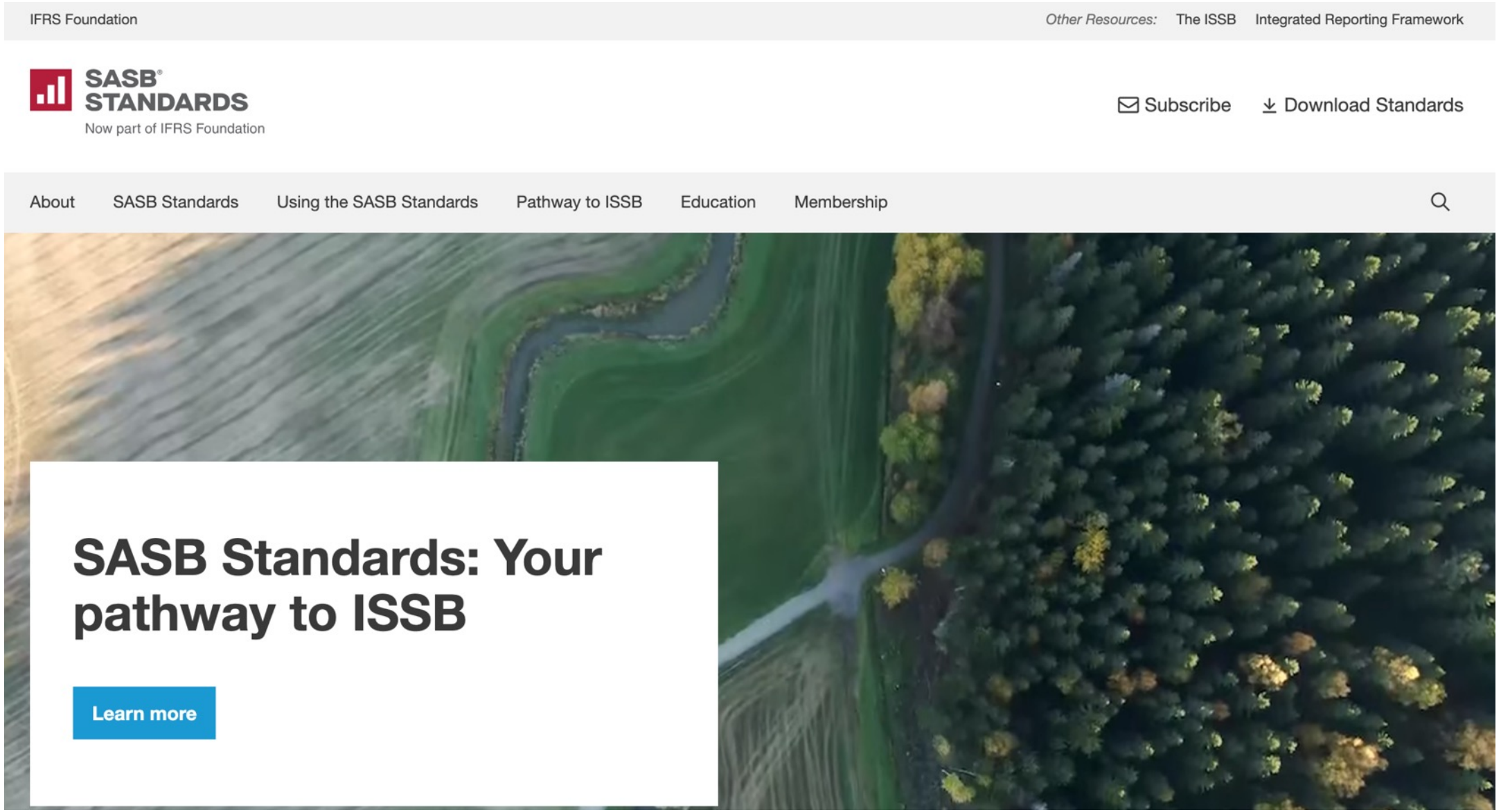
https://www.globalreporting.org/

CDP (Carbon Disclosure Project)



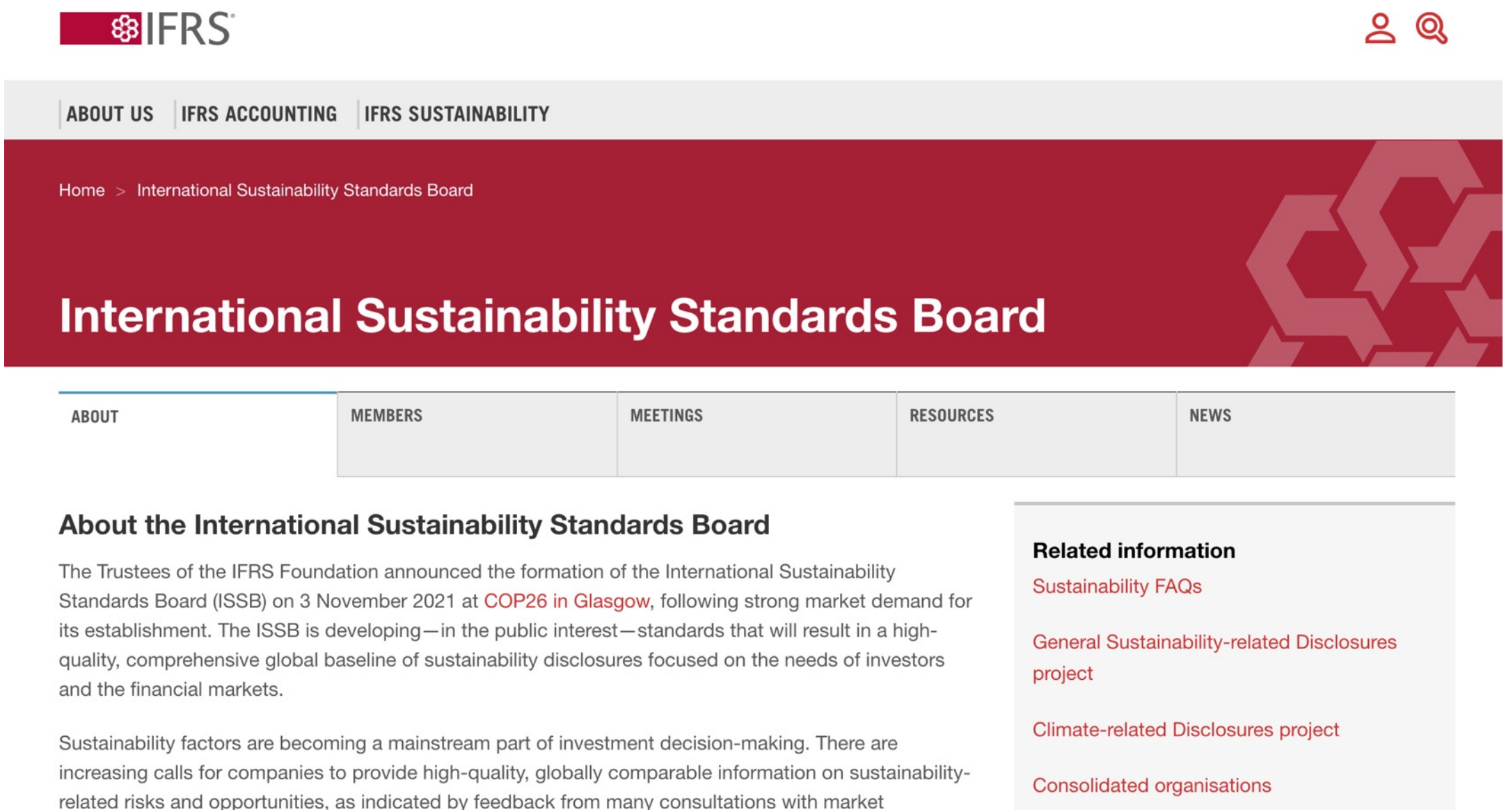
https://www.cdp.net/

SASB (Sustainability Accounting Standards Board)



https://sasb.org/

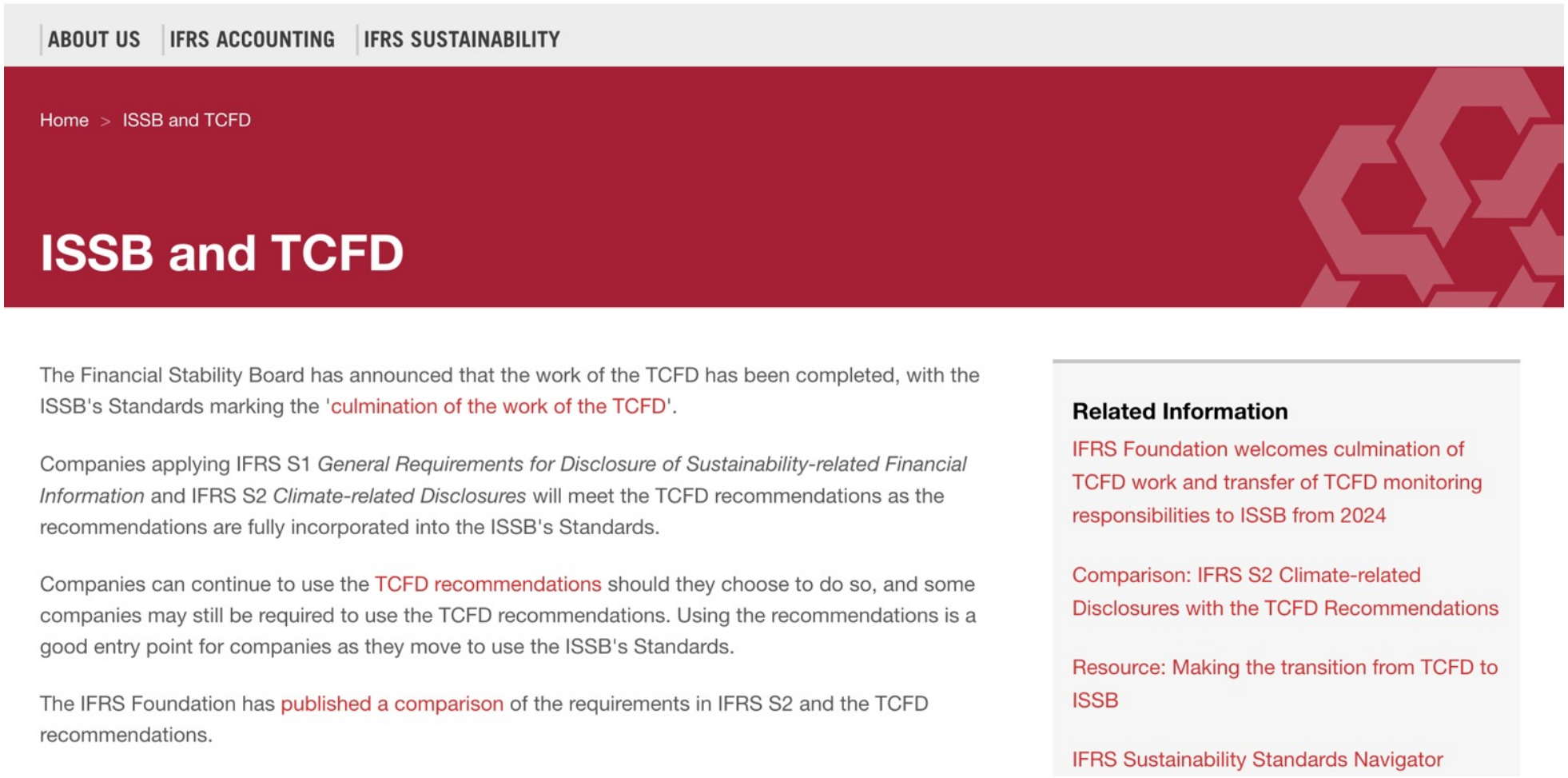
ISSB (International Sustainability Standards Board)



https://www.ifrs.org/groups/international-sustainability-standards-board/

TCFD
(Task Force on Climate-related Financial Disclosures)

https://www.ifrs.org/sustainability/tcfd/



https://www.fsb-tcfd.org/

Implementing Data-Driven Sustainability Strategies

Implementing Data-Driven Sustainability Strategies

- **Setting science-based targets and integrating metrics into operations**
- **Advanced analytics for sustainability strategy**
- **Overcoming challenges and future trends in data science for sustainability**

Setting Science-Based Sustainability Targets

- Science-based targets:
  - Emissions reduction goals aligned with Paris Agreement
- Using historical data and projections to set realistic targets
- Aligning with Sustainable Development Goals (SDGs)
- Case study:
  - Unilever's science-based targets and progress tracking

Integrating Sustainability Metrics into Business Operations

- Embedding ESG KPIs in performance management
- Developing sustainability scorecards and dashboards
- Data-driven decision making for sustainability
- Challenges: Data silos, resistance to change
- Best practices: Cross-functional teams, executive sponsorship

Advanced Analytics for Sustainability Strategy

- Scenario analysis for climate risk assessment
- AI for optimizing energy consumption in real-time
- Predictive modeling for supply chain sustainability risks
- NLP for analyzing sustainability-related news and social media

Communicating ESG Performance to Stakeholders

- Comprehensive sustainability reporting (GRI, SASB standards)
- Data visualization: Infographics, interactive dashboards
- Real-time sustainability performance portals
- Addressing data privacy in sustainability communication

Overcoming Challenges in Data-Driven Sustainability

- **Improving data quality and standardization**
- **Change management strategies for adoption**
- **Ethical data collection and use in sustainability**
- **Staying compliant with evolving ESG regulations**

Data-Driven Sustainable Investing: Building ESG-Conscious Portfolios

- **ESG investing:**
  - **Considering environmental, social, and governance factors in investment decisions.**
- **Performance of ESG-focused funds vs. traditional funds.**
- **Building ESG portfolios:**
  - **Screening, index funds, impact investing.**

ESG Risk Assessment and Mitigation: Managing ESG Risks for Resilience

- **Identifying ESG risks across the supply chain (environmental impact, labor practices).**
- **Using data to quantify potential financial consequences of ESG risks.**
- **Developing strategies to mitigate ESG risks.**
- **Integrating ESG risk monitoring into decision-making for greater resilience.**

Driving Corporate Sustainability:
Data-Powered Sustainability Transformations

- **Setting data-driven ESG targets and tracking progress.**
- **Utilizing dashboards and analytics for real-time ESG insights.**
- **Engaging stakeholders through transparent ESG reporting.**

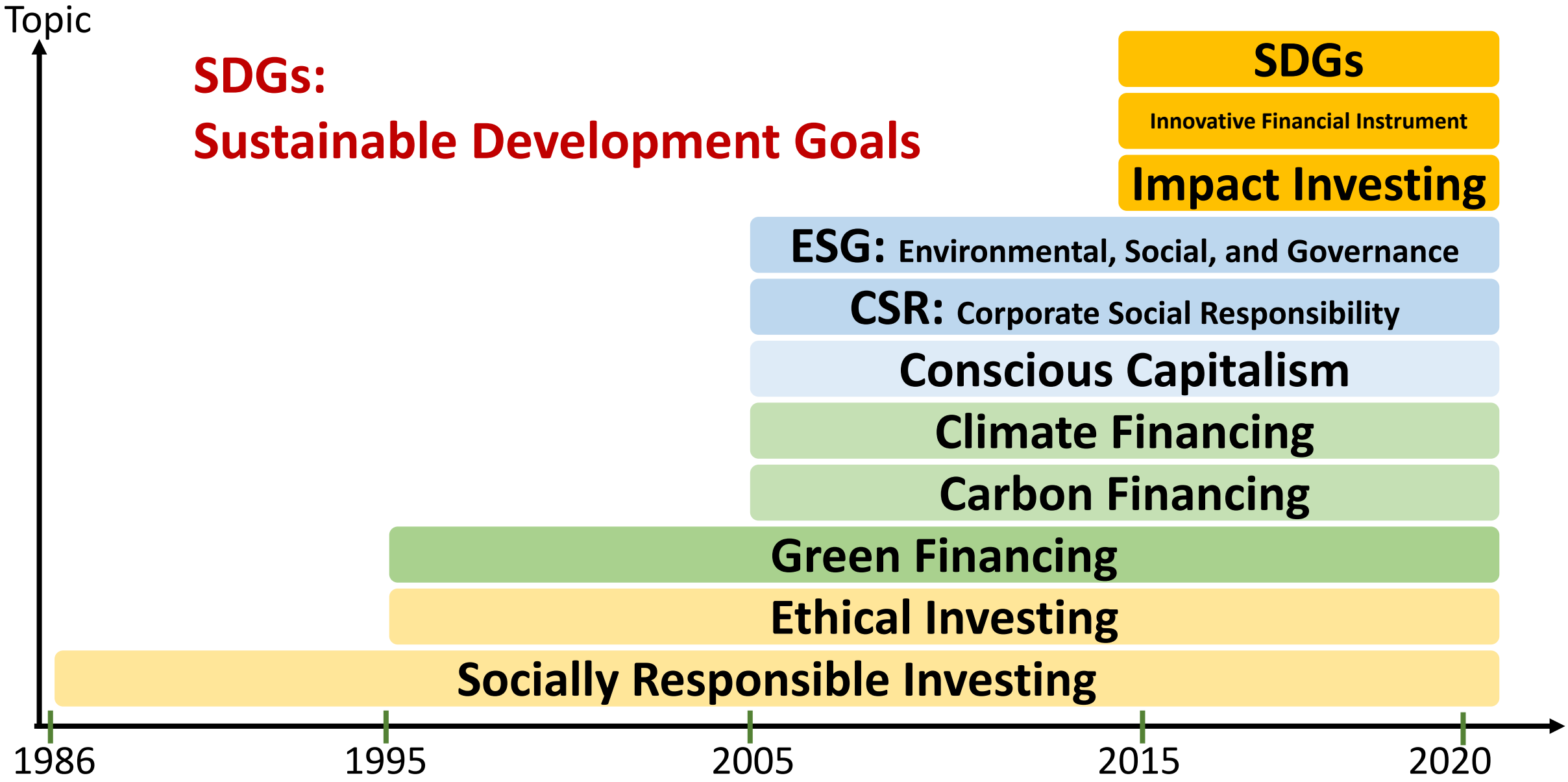
The Future of Data-Driven ESG: A Sustainable Future Powered by Data

- **Emerging trends in ESG data and analytics**
  - **e.g., AI for ESG insights**
- **The potential of data to drive progress towards global sustainability goals.**
- **The role of data scientists in shaping ethical and equitable ESG practices.**
- **Be part of the ESG Data Science solution.**

Future Trends in Data Science for Sustainability

- **Blockchain for transparent, sustainable supply chains**
- **Digital twins for environmental impact simulation**
- **Edge computing for real-time sustainability management**
- **Quantum computing for complex climate modeling**

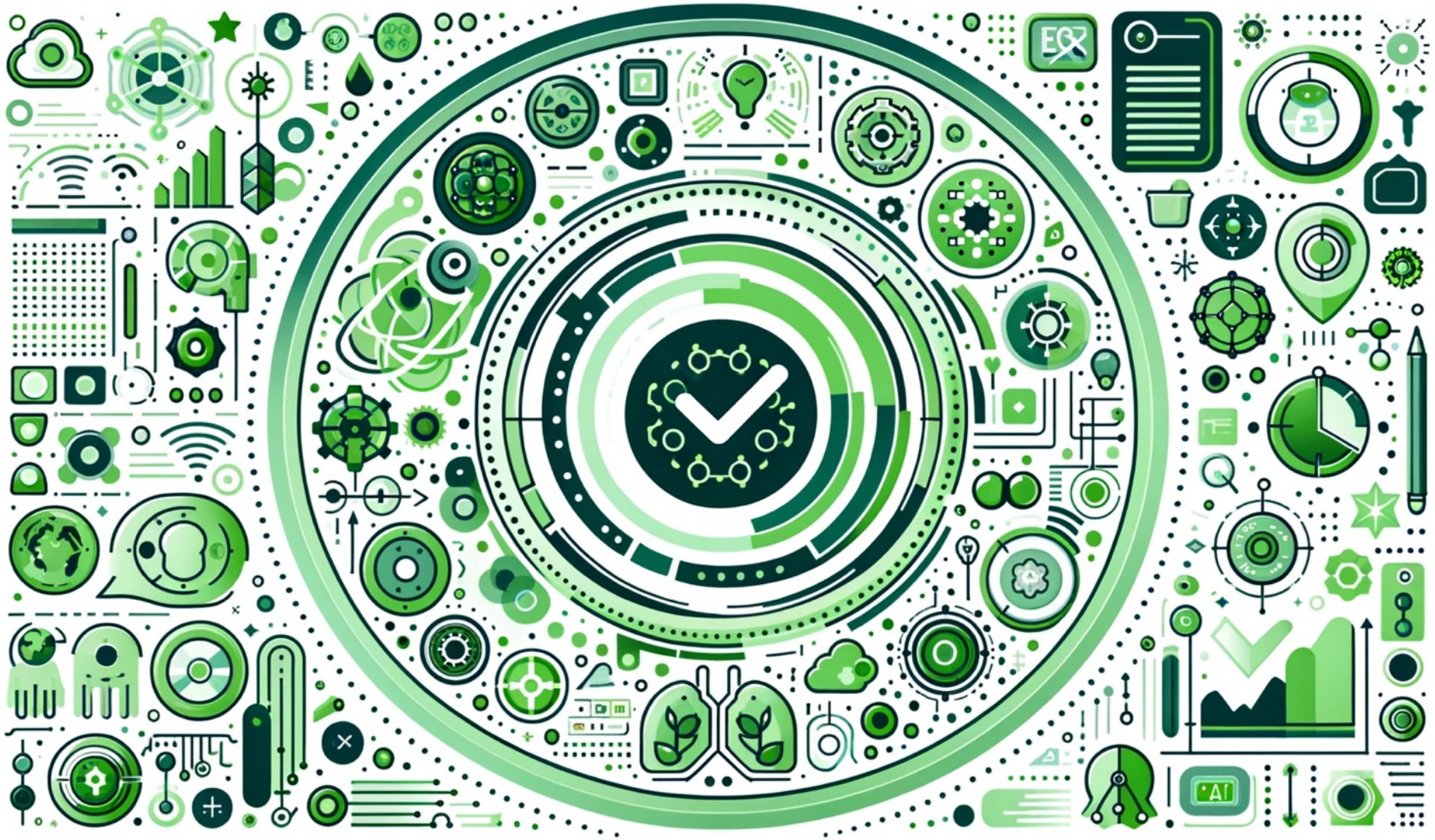
Evolution of Sustainable Finance Research



Source: Kumar, S., Sharma, D., Rao, S., Lim, W. M., & Mangla, S. K. (2022). Past, present, and future of sustainable finance: Insights from big data analytics through machine learning of scholarly research. Annals of Operations Research, 1-44.

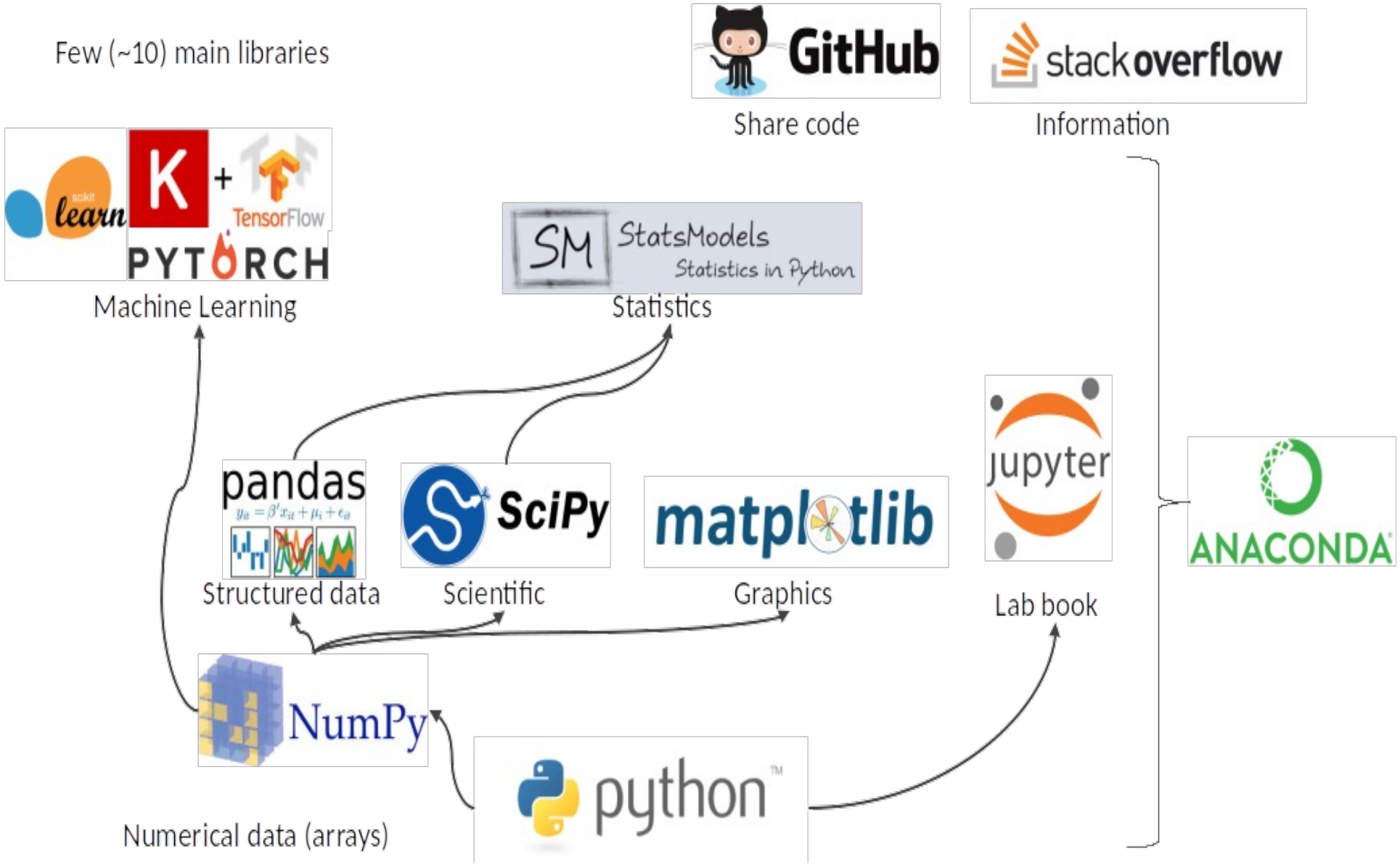
Generative AI and LLMs for Sustainability and ESG Data Analytics

Generative AI for ESG Rating and Reporting Generation



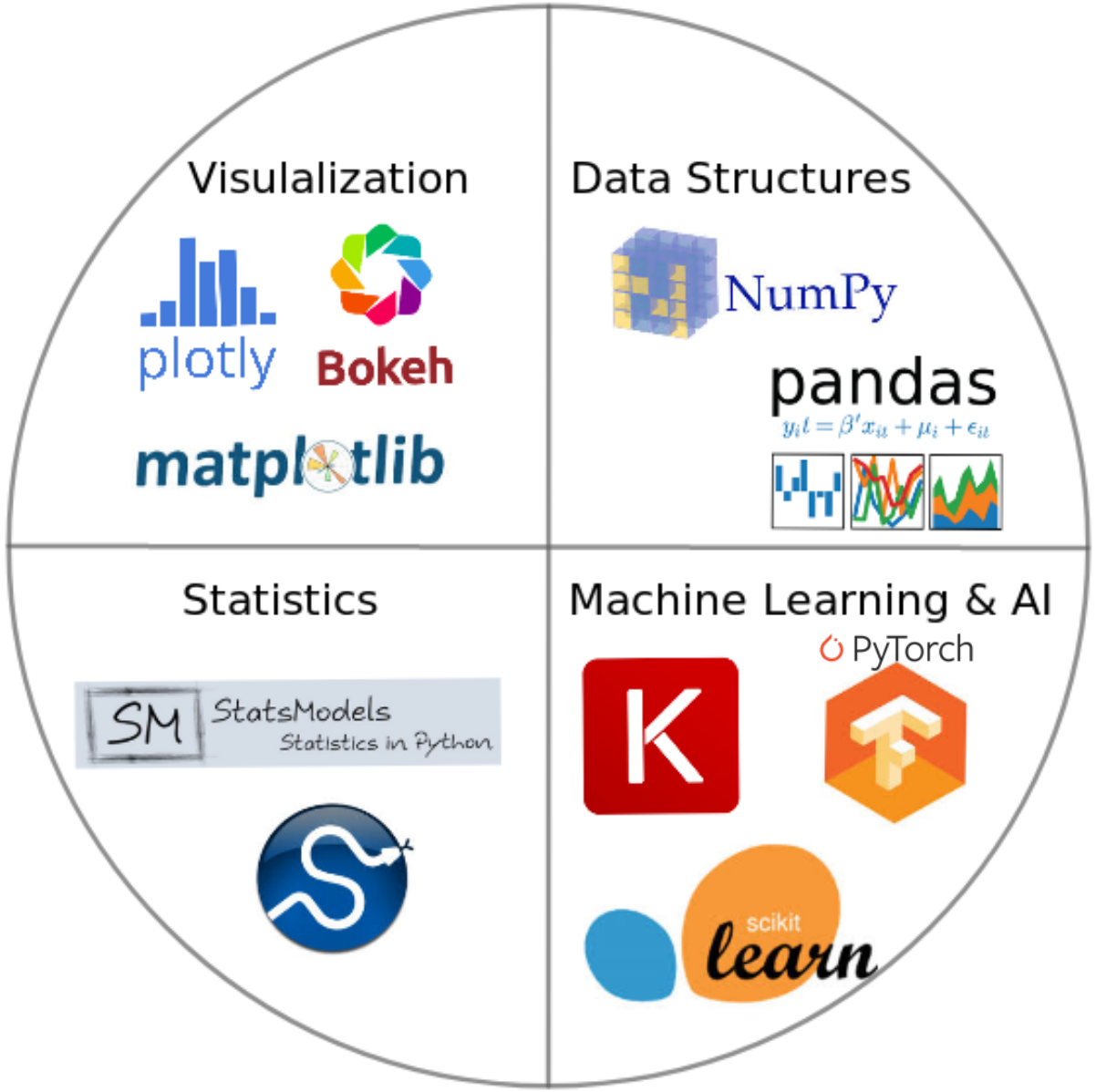
ESG Data Science Foundation with Python

Python Ecosystem for Data Science



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

Python Ecosystem for Data Science

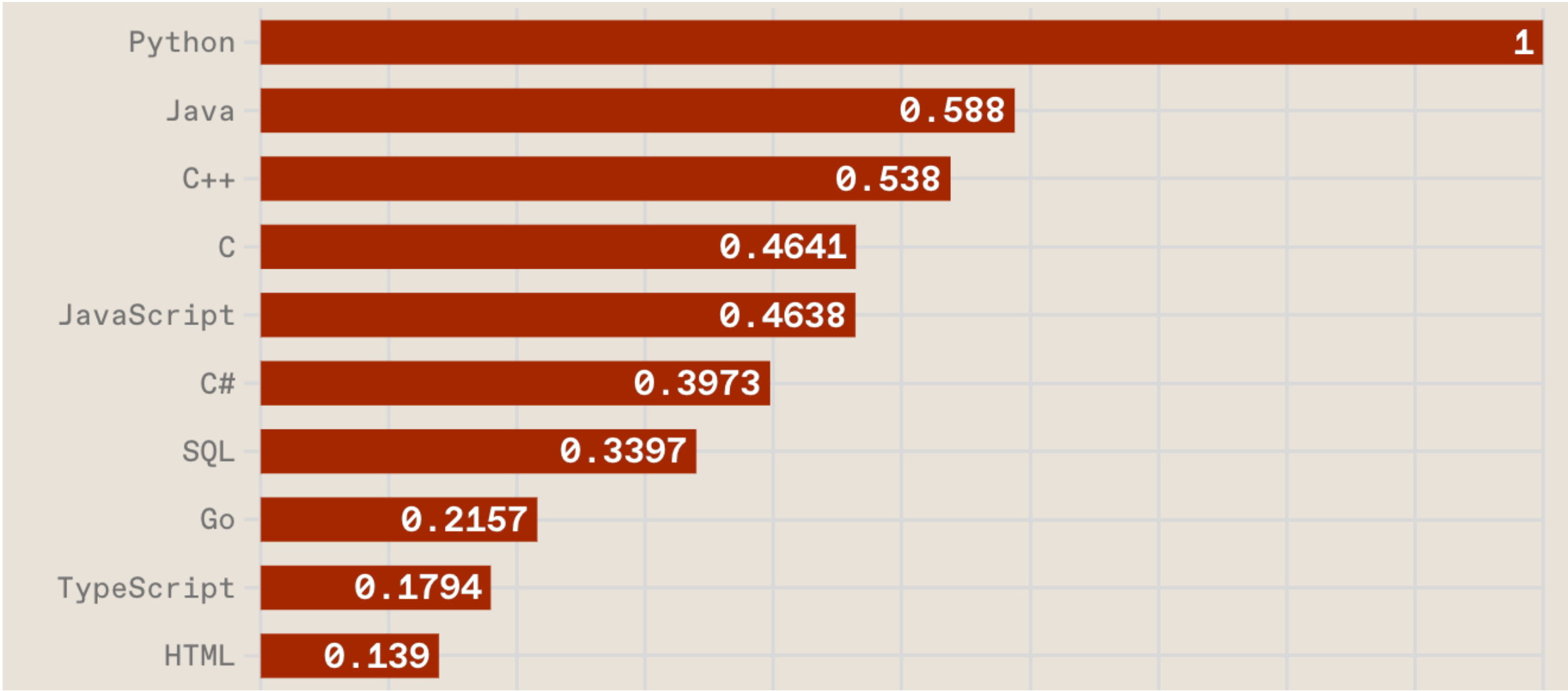


Source:https://duchesnay.github.io/pystatsml/introduction/python_ecosystem.html

Python

Programming

Top Programming Languages



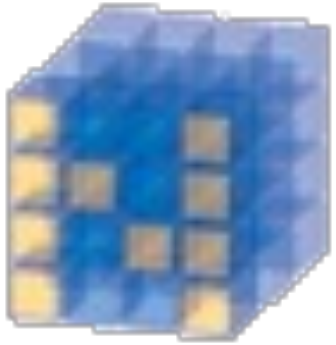
https://spectrum.ieee.org/the-top-programming-languages-2023

python™

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

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

Numpy



NumPy

Base

N-dimensional array

package

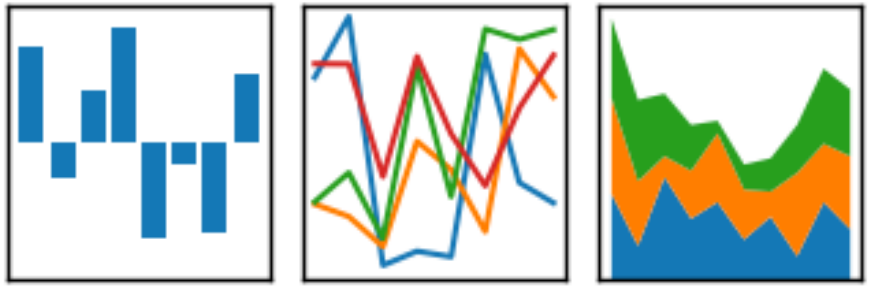
Python matplotlib

Source: https://matplotlib.org/

Python Pandas

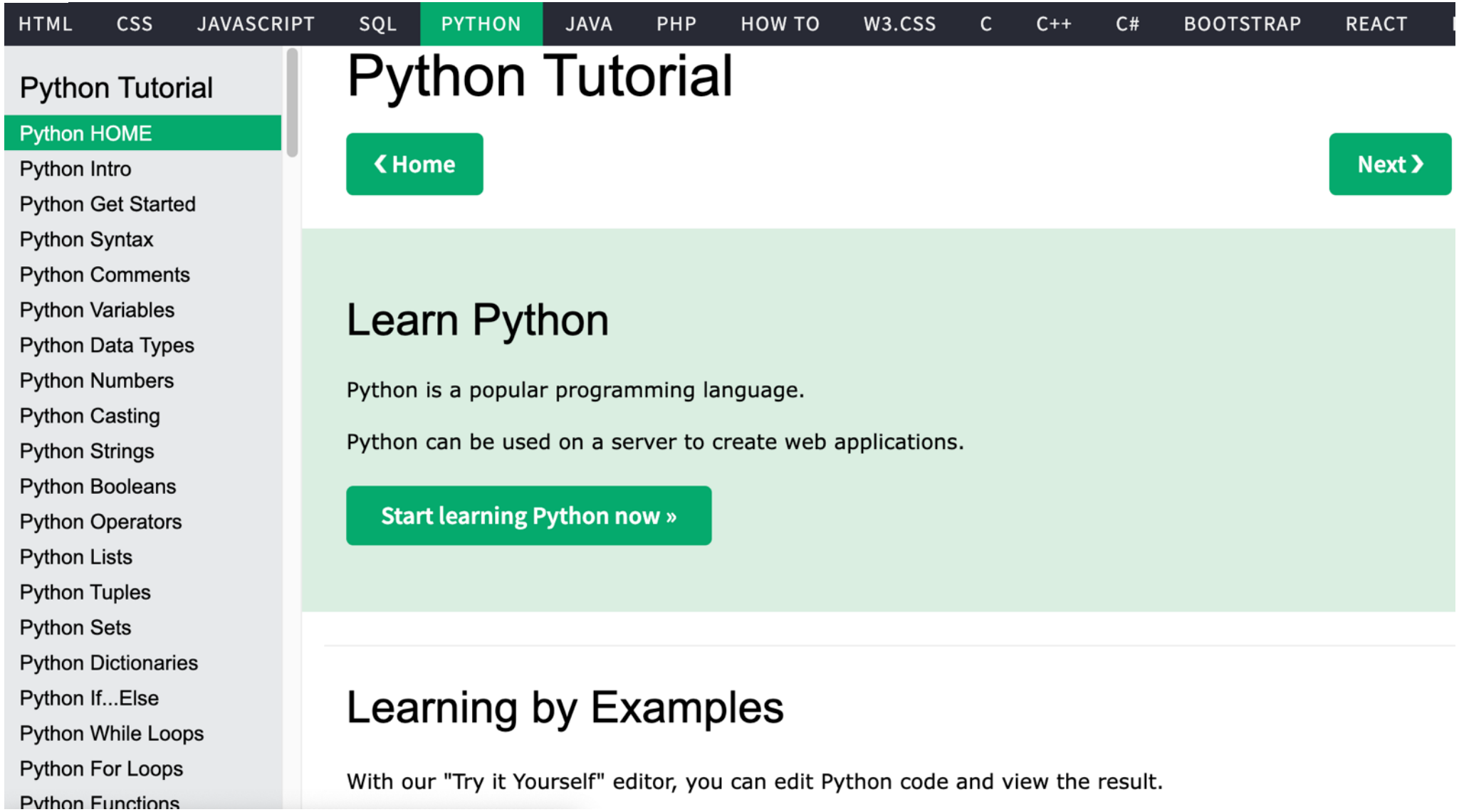
pandas

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



http://pandas.pydata.org/

W3Schools Python

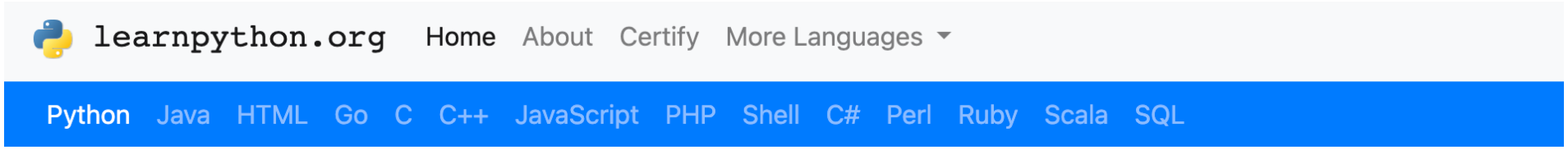


https://www.w3schools.com/python/

W3Schools Python: Try Python

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

LearnPython.org



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

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

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

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

Welcome

Welcome to the LearnPython.org interactive Python tutorial.

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

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

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

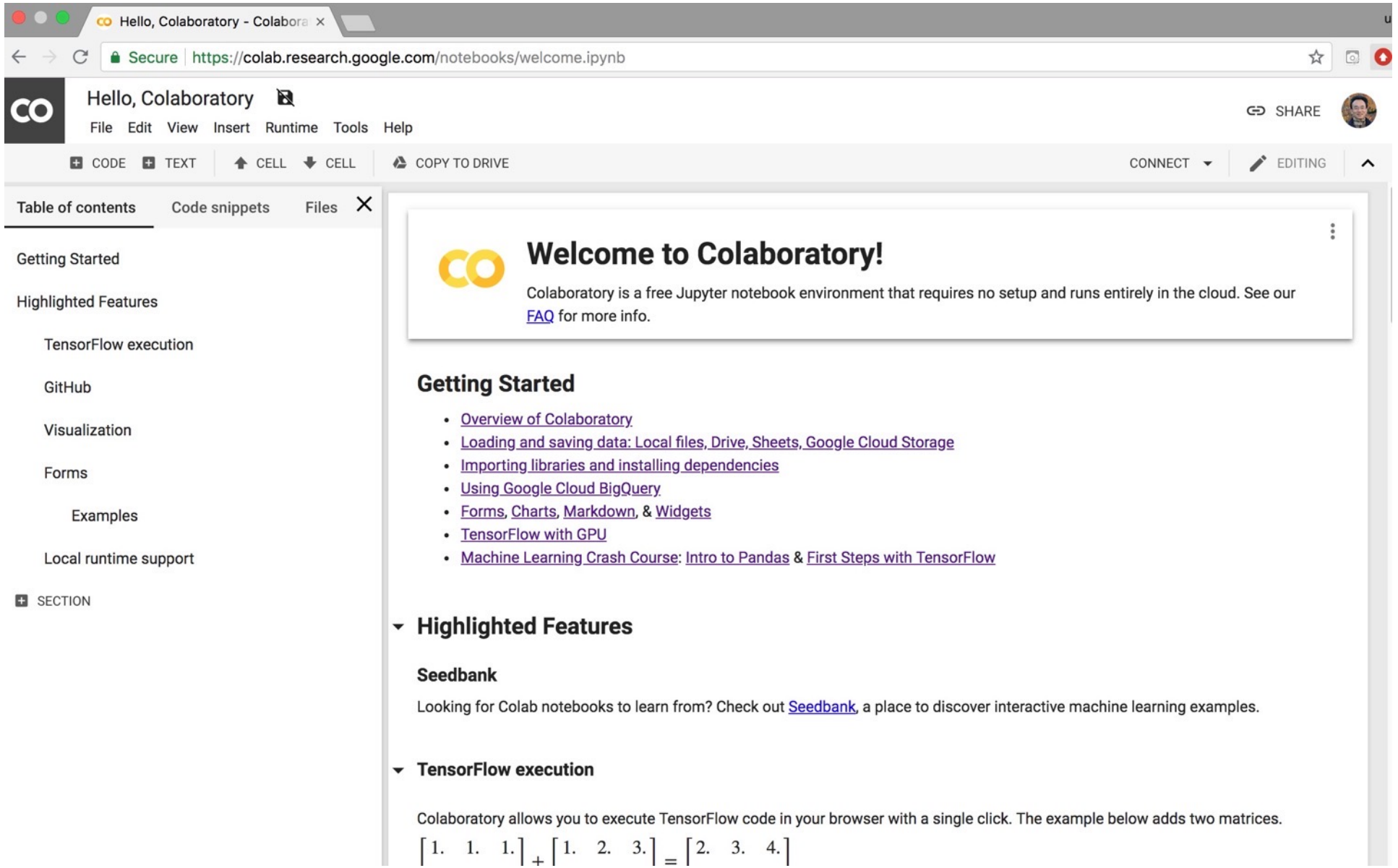
Just click on the chapter you wish to begin from, and follow the instructions. Good luck!

https://www.learnpython.org/

Google's Python Class

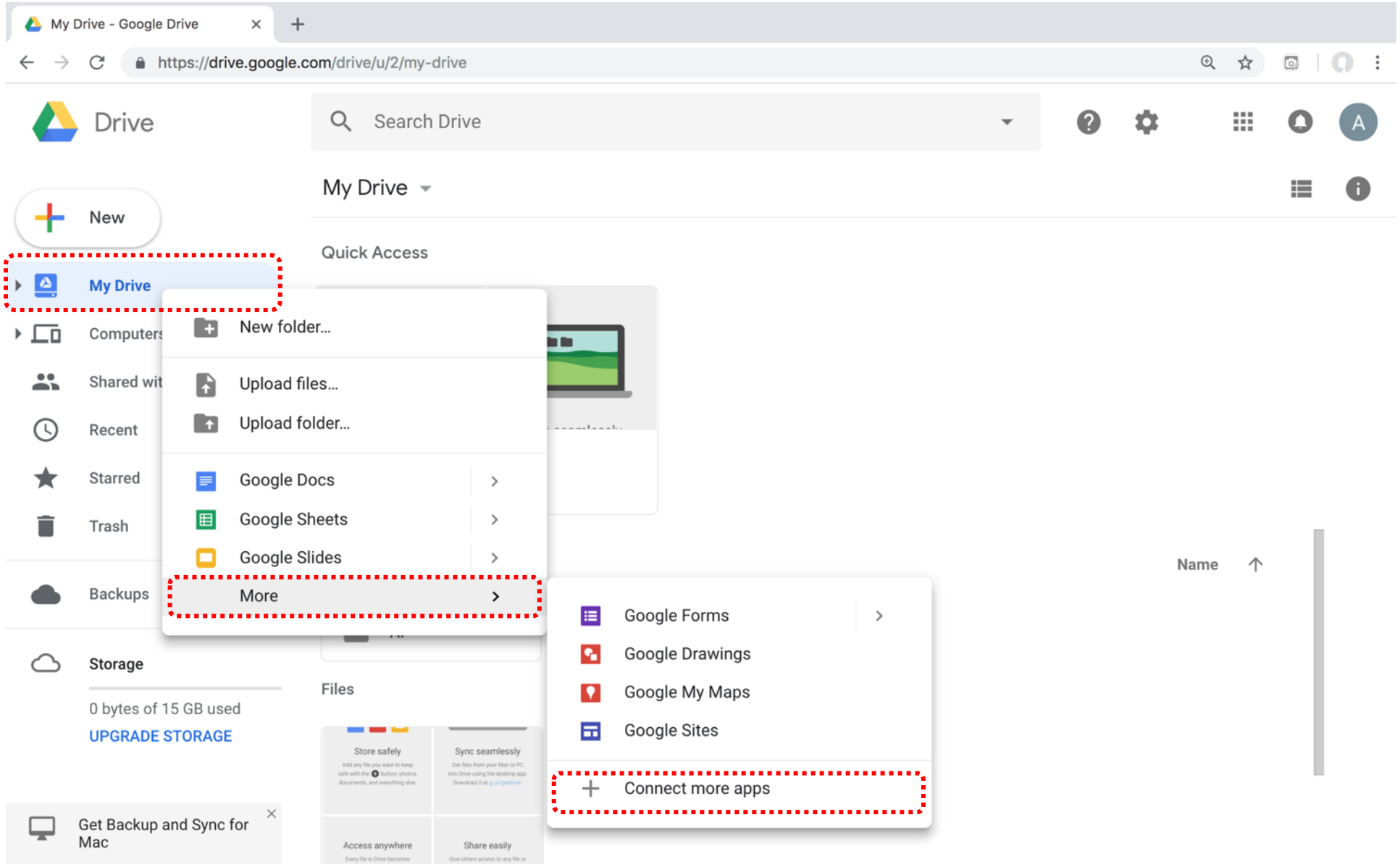
https://developers.google.com/edu/python

Google Colab

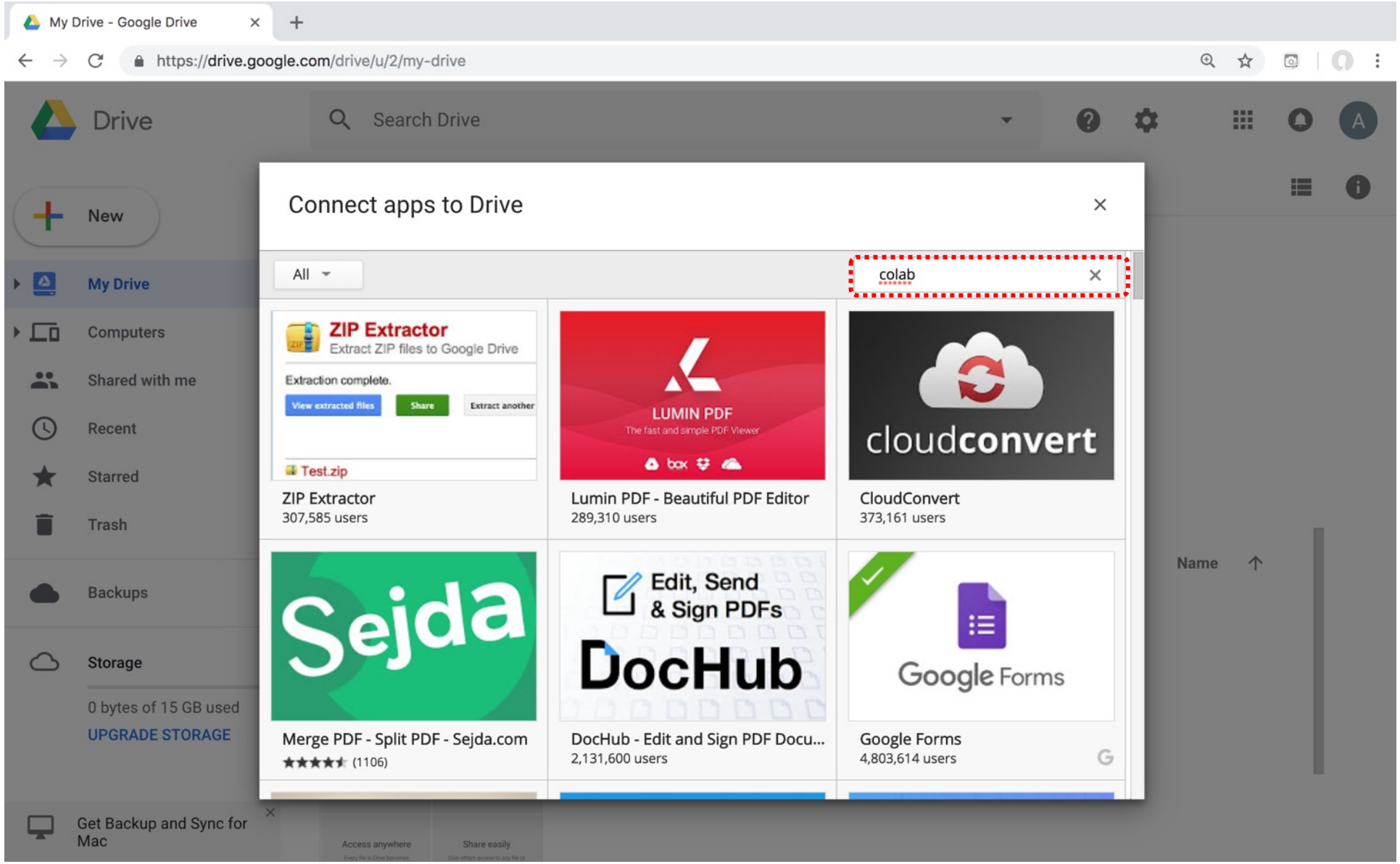


https://colab.research.google.com/notebooks/welcome.ipynb

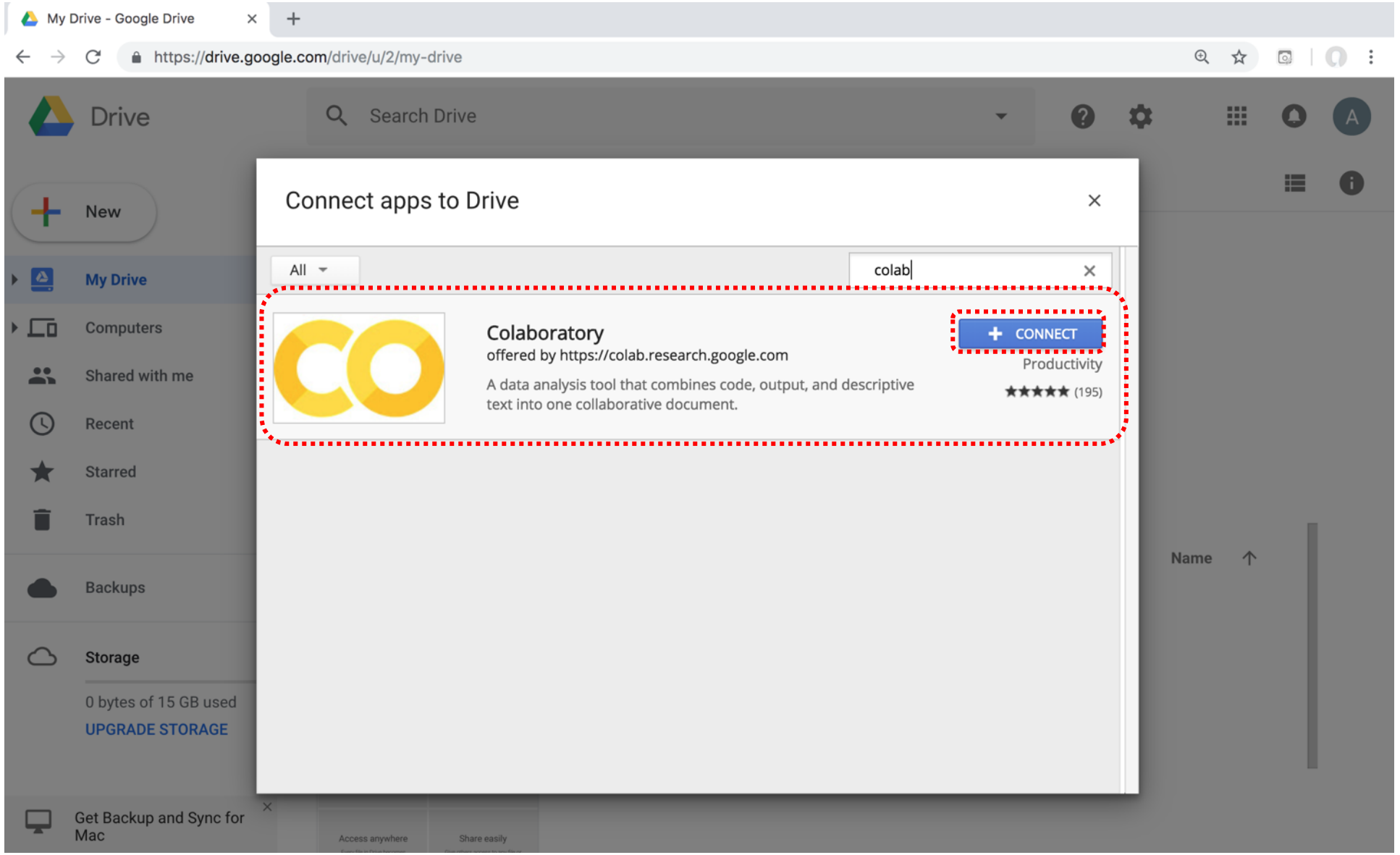
Connect Google Colab in Google Drive



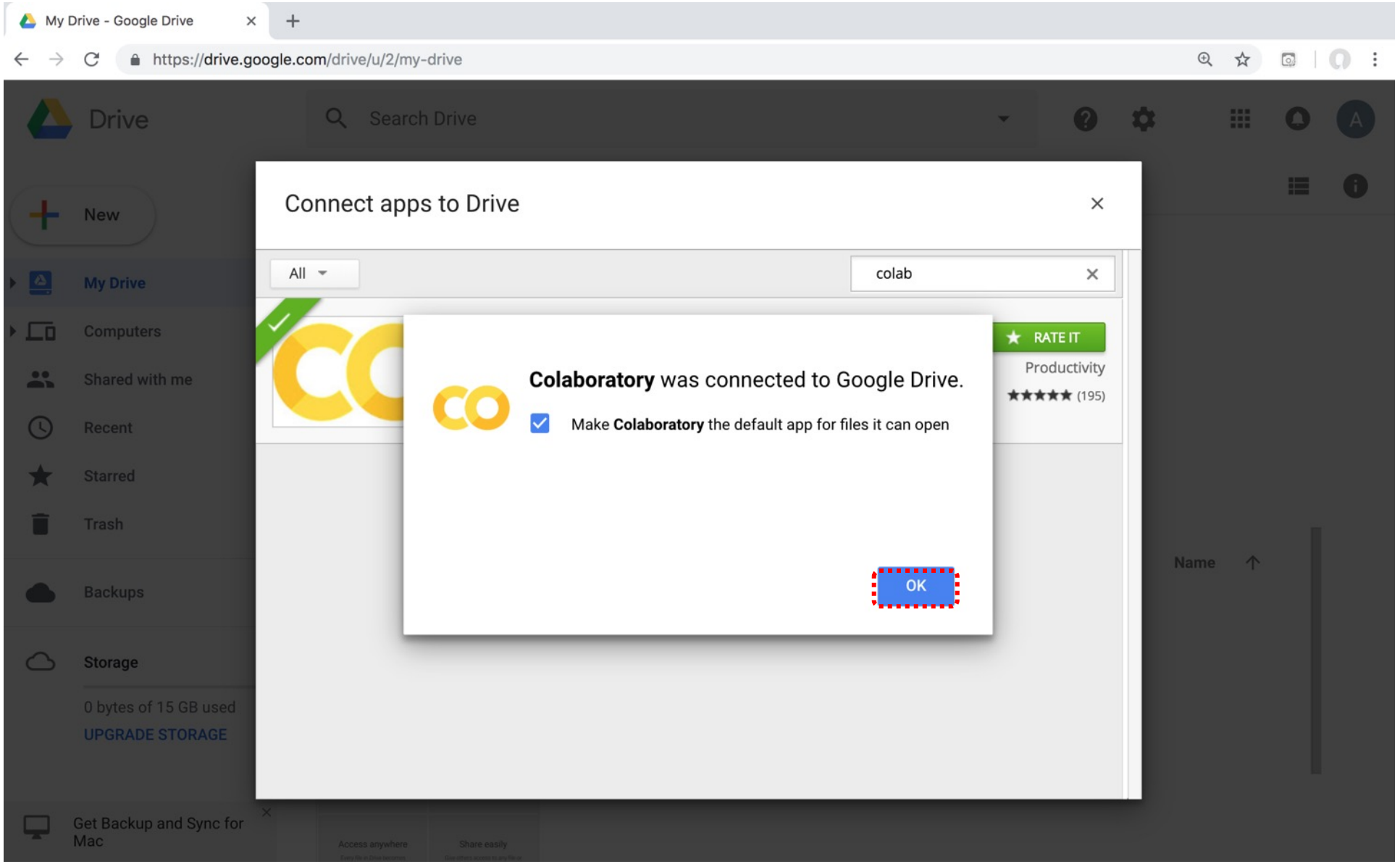
Google Colab



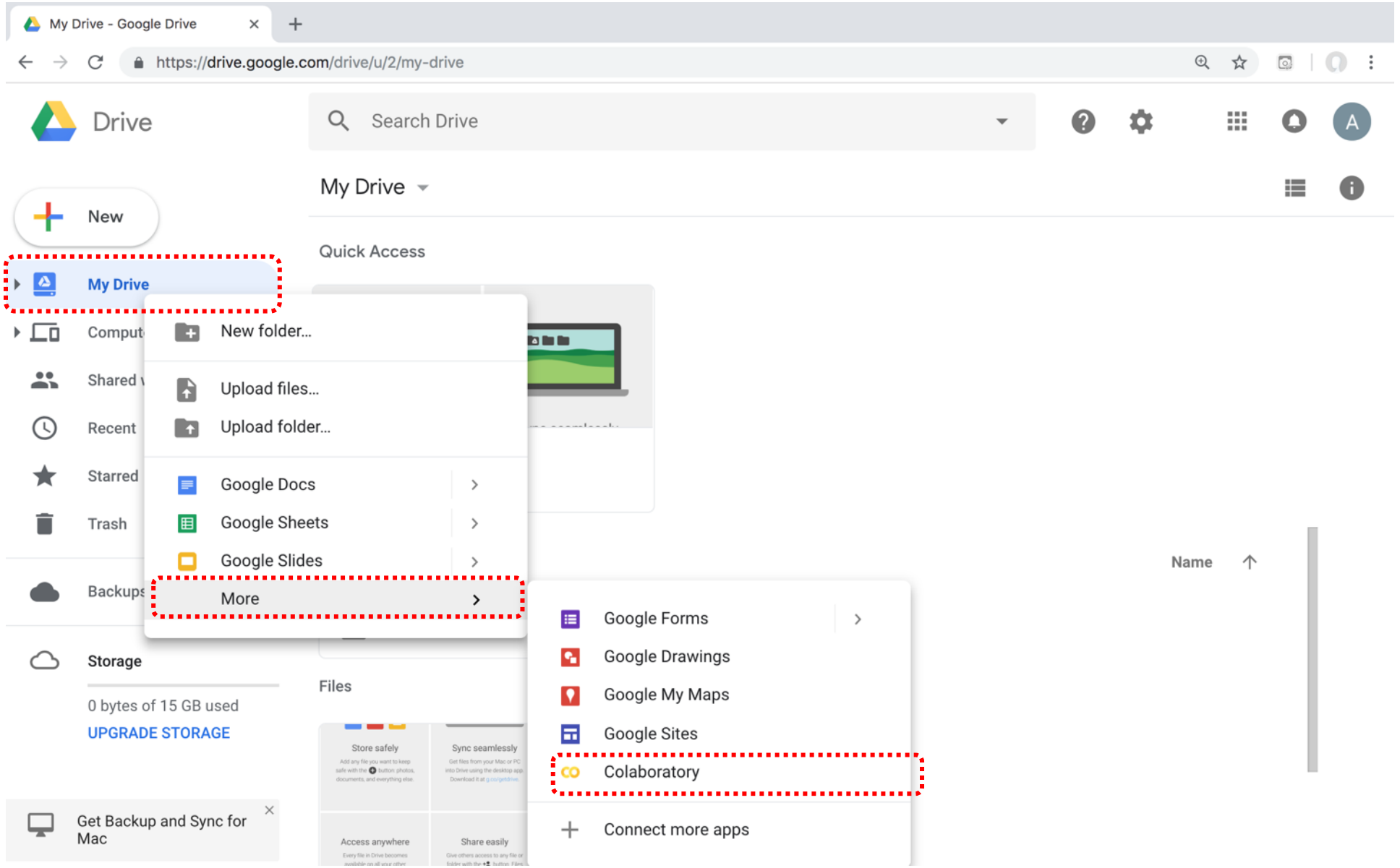
Google Colab



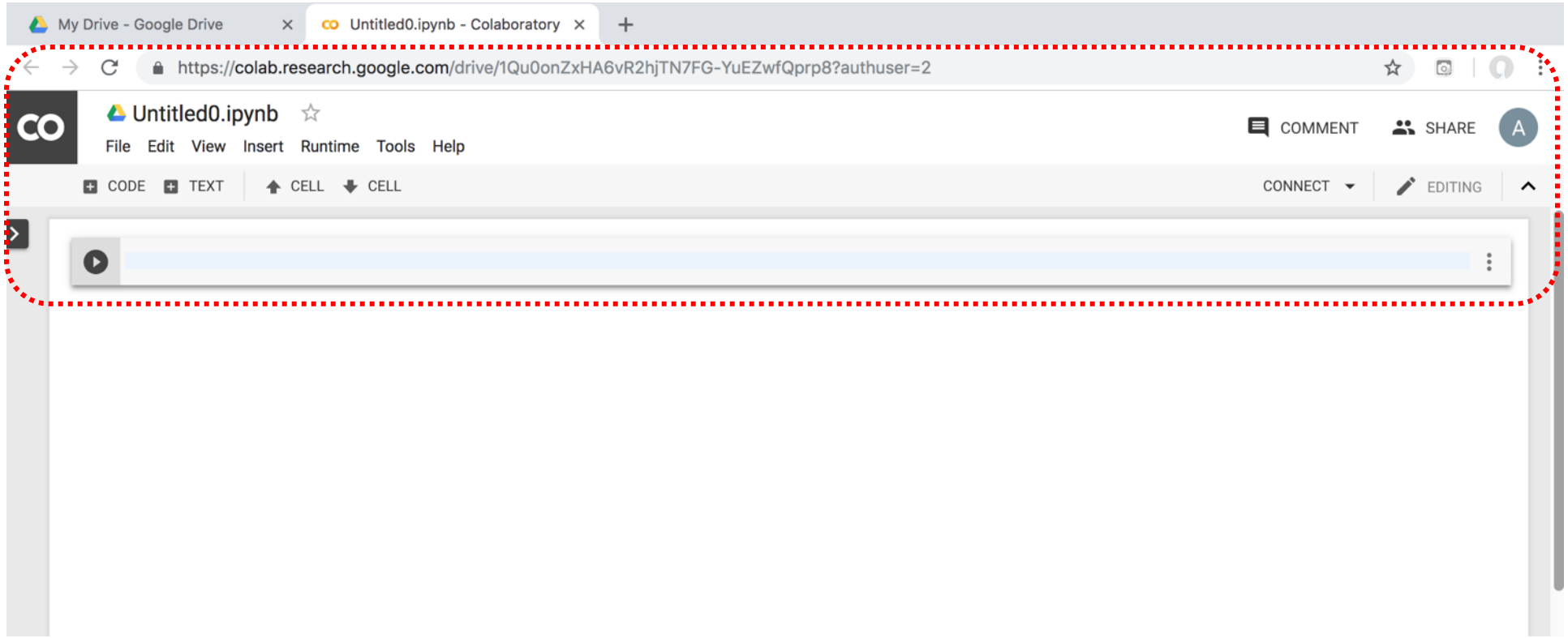
Connect Colaboratory to Google Drive



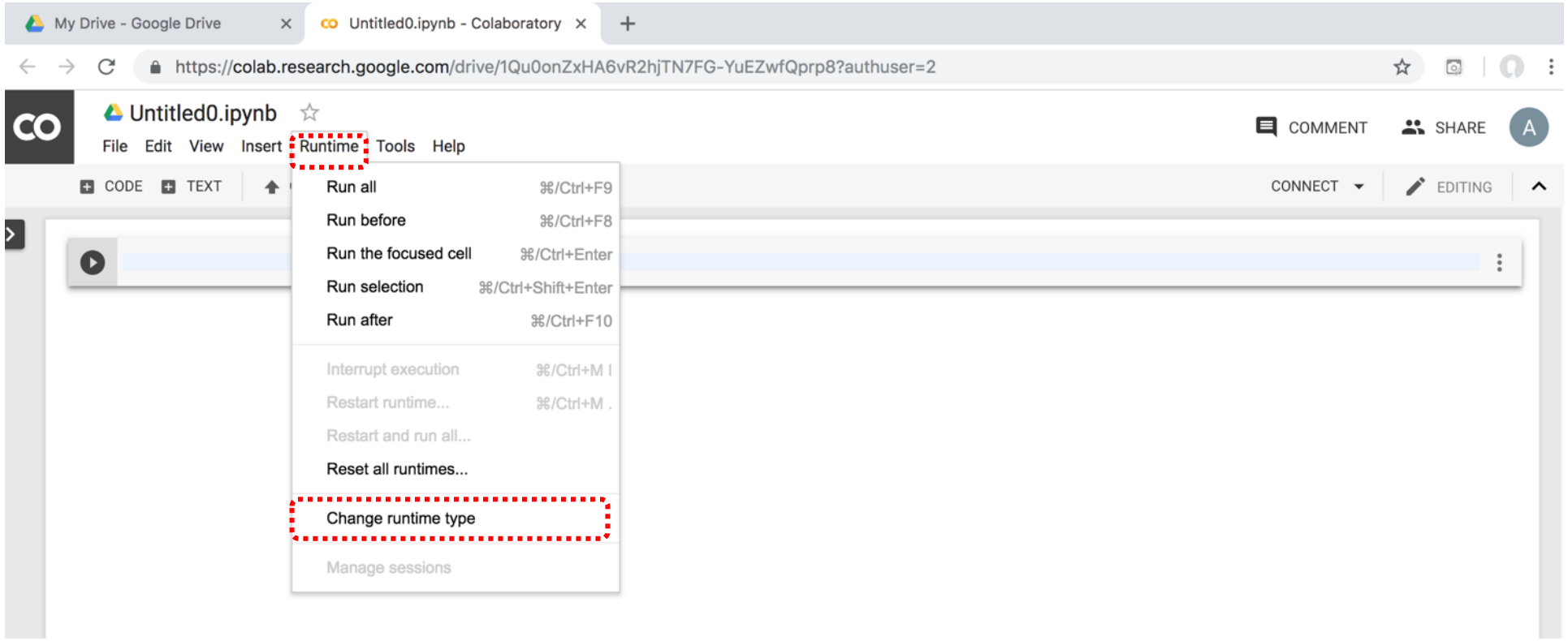
Google Colab



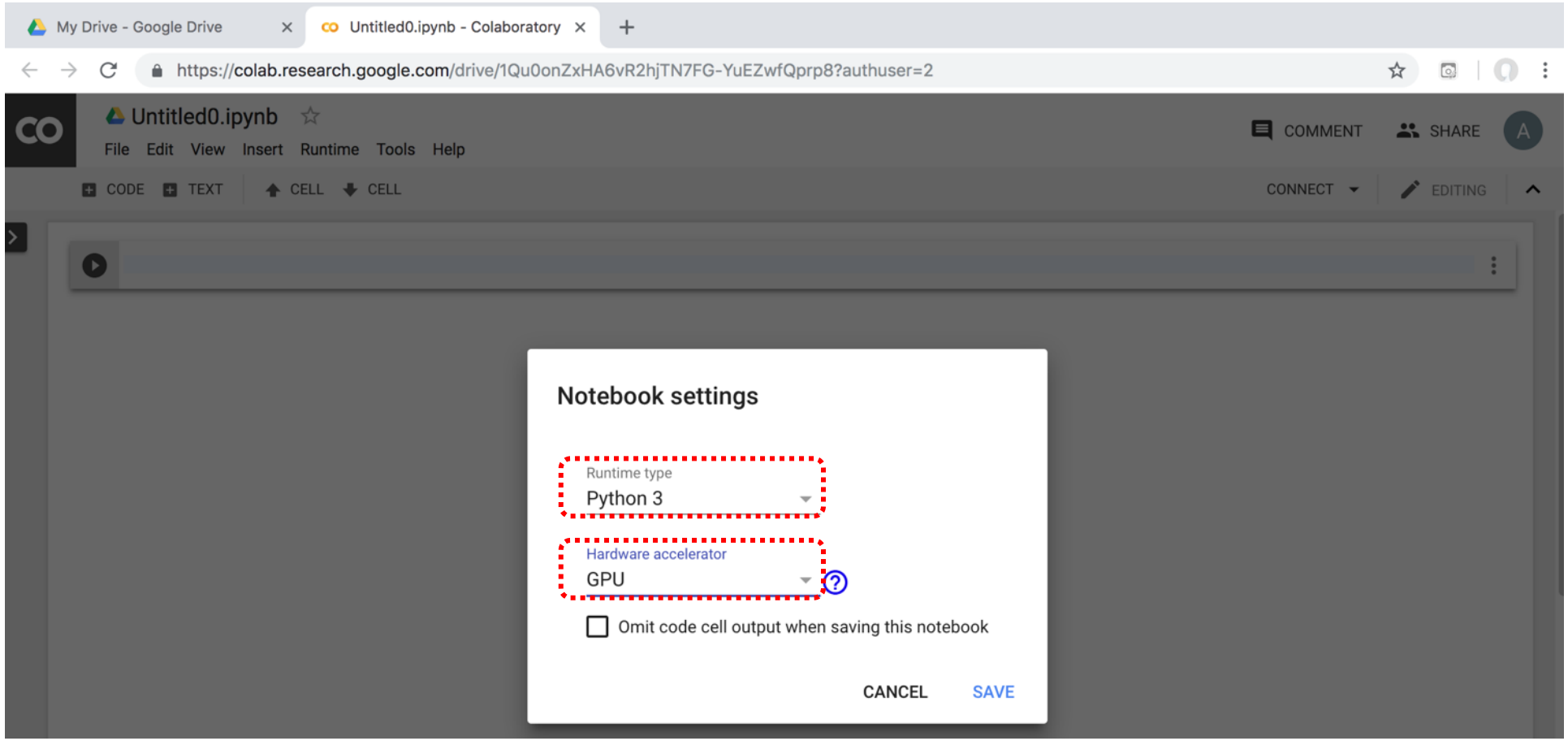
Google Colab



Google Colab



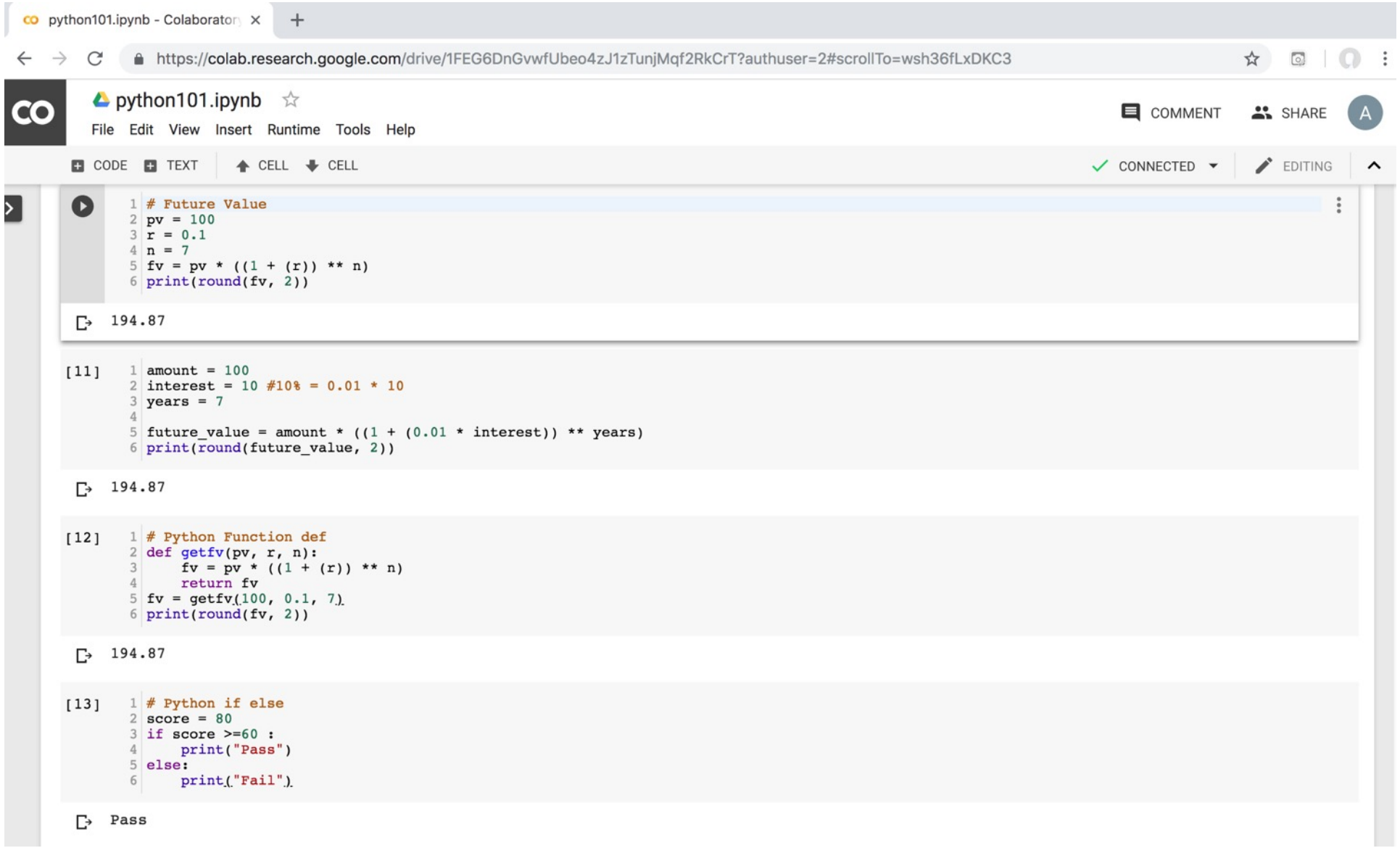
Run Jupyter Notebook
Python3 GPU
Google Colab



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

Python in Google Colab (Python101)

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

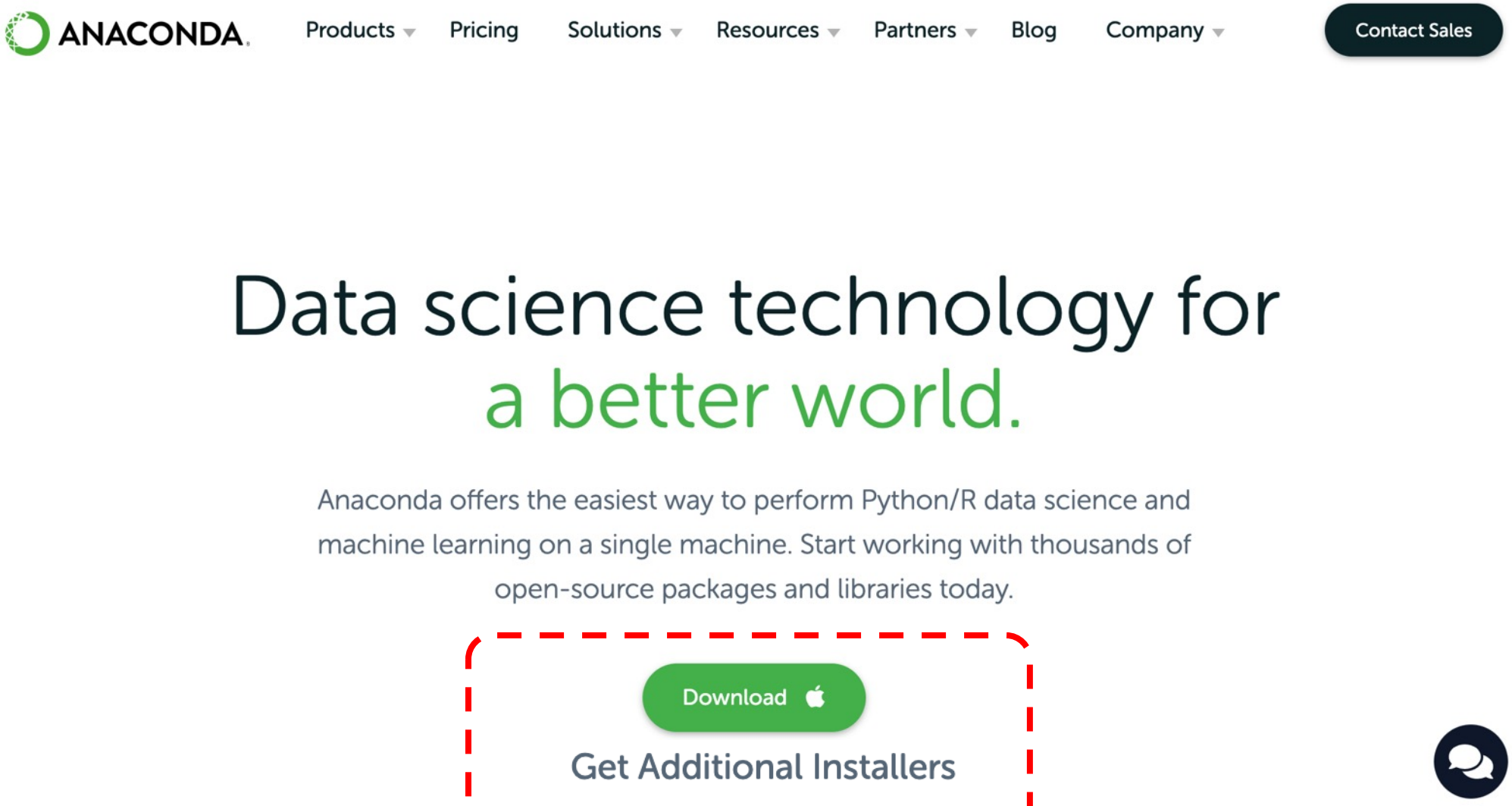


https://tinyurl.com/aintpupython101

Anaconda
The Most Popular
Python
Data Science Platform

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

Download Anaconda

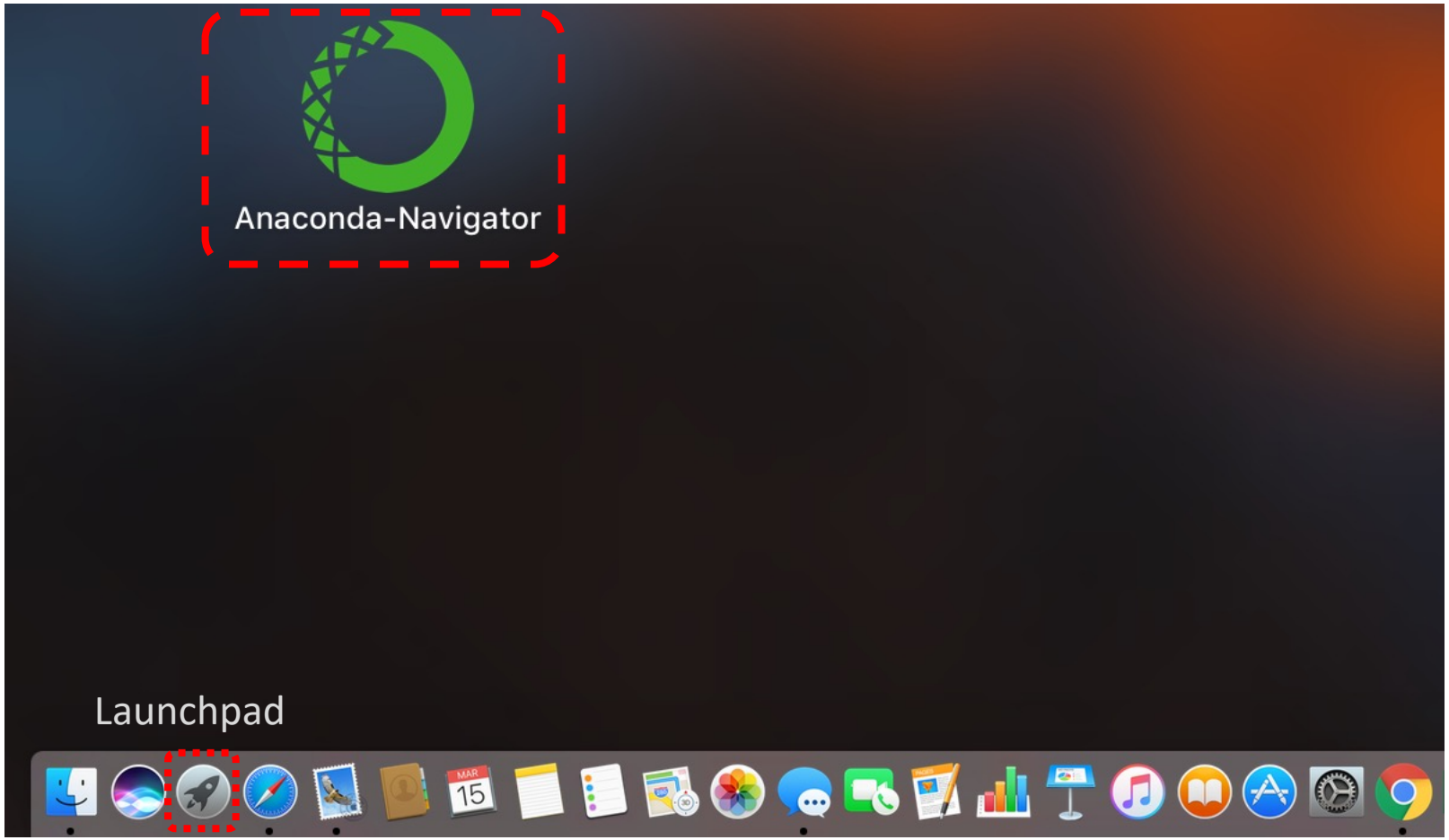


https://www.anaconda.com/download

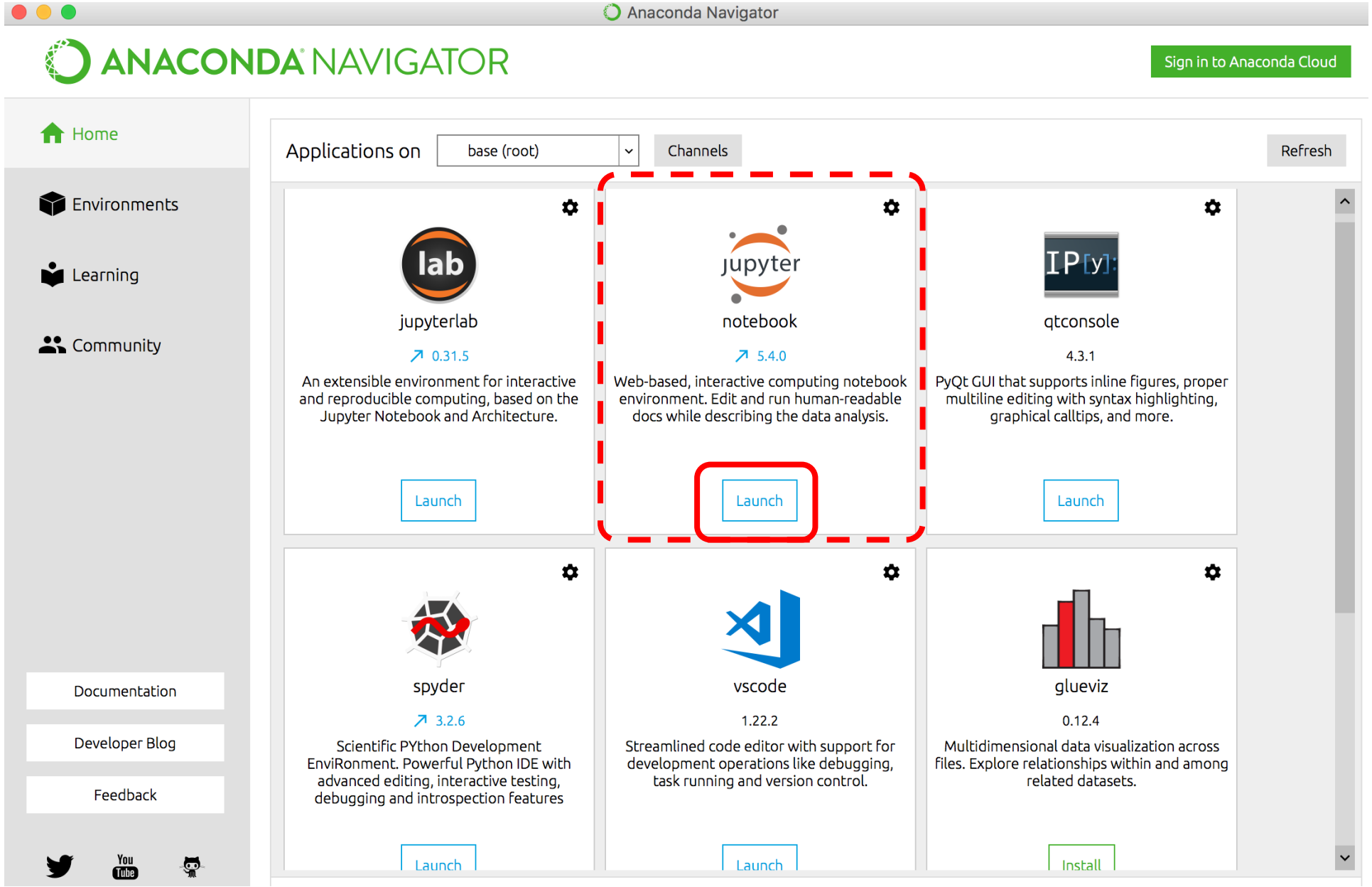
Python

HelloWorld

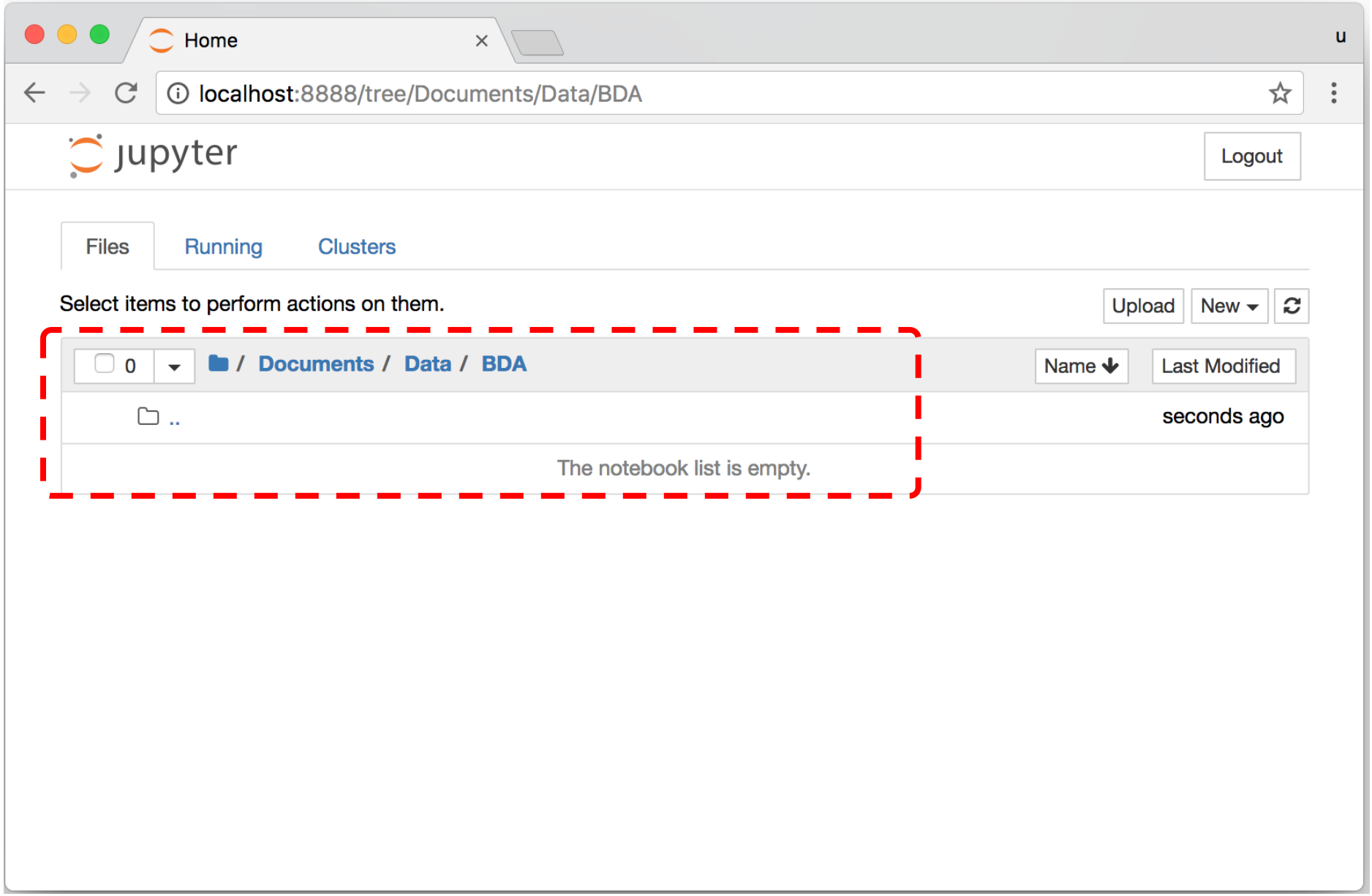
Anaconda-Navigator



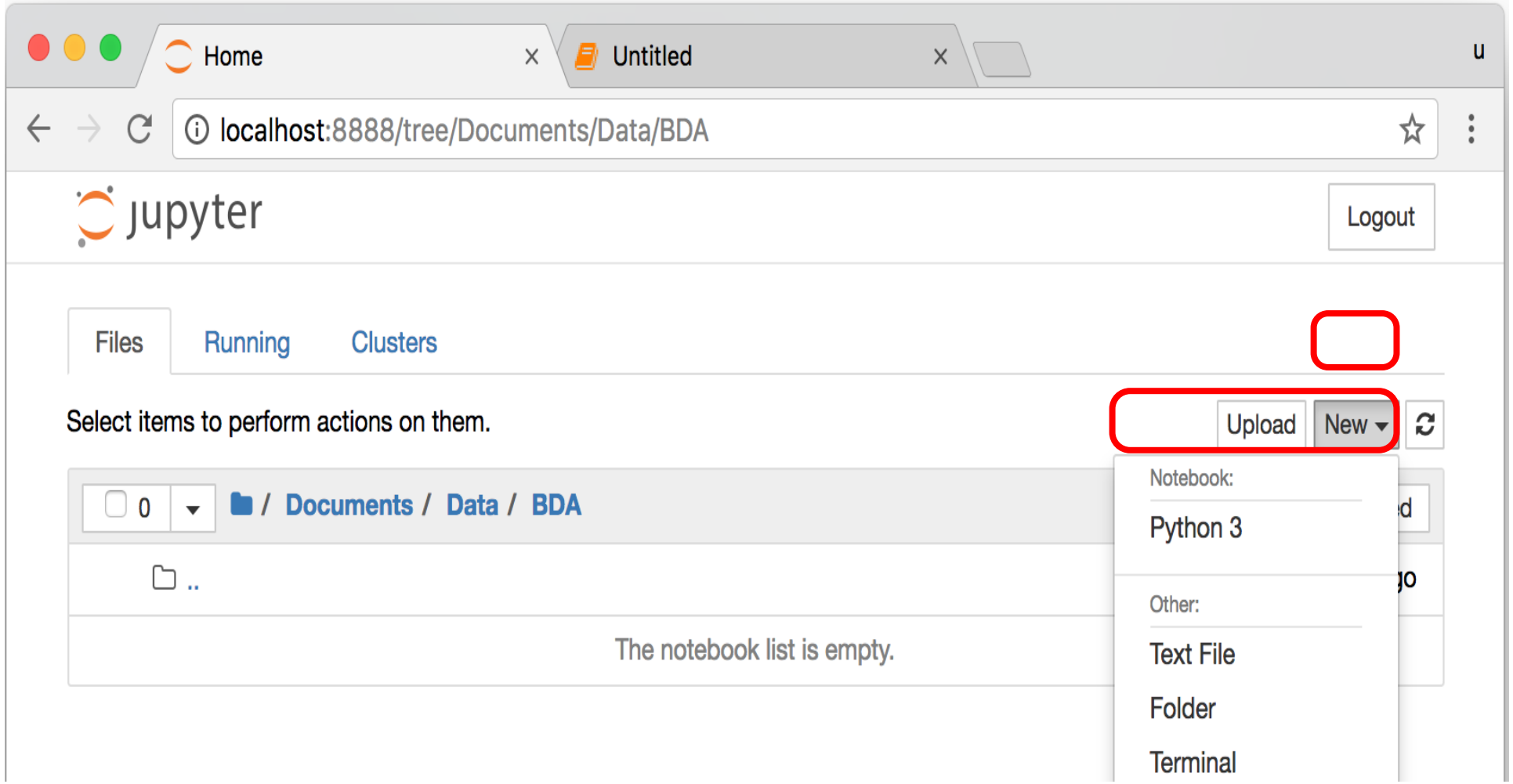
Anaconda Navigator



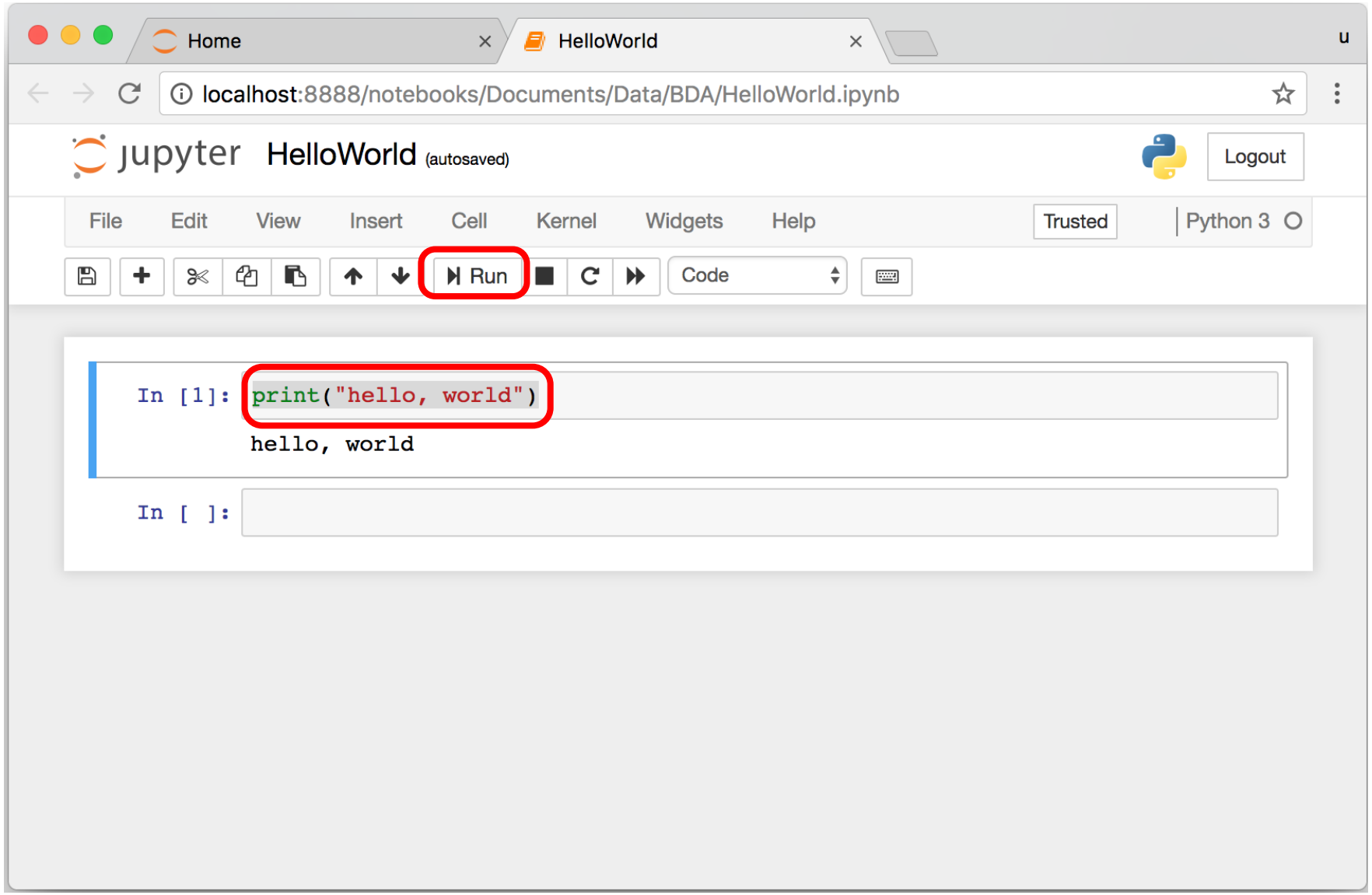
Jupyter Notebook



Jupyter Notebook
New Python 3



print("hello, world")



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)
```

```
True
False
False
```

Python BMI Calculator

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

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

```
Your BMI is: 20.8
```

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

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

How much is your $100 worth after 7 years?

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

```
fv = 194.87
```

Future Value

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

```
194.87
```

Future Value

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

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

```
194.87
```

Python
Data Structures

Python Data Structures

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

Python Data Structures

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

Python Data Types

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

Python Collections

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

Python Dictionaries {k:v}

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

Lists []

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

```
4
60
70
90
```

Lists []

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

Python List Methods

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

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

Tuples ()

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

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

```
10
20
30
50
```

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

Sets {}

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

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

Dictionary {key : value}

```
Python Dictionary
Key   → Value
'EN'  → 'English'
'FR'  → 'French'
```

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

```
English
```

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

Python Data Structures

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

Python Control Logic and Loops

Python Control Logic and Loops

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

Python if...else

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

Python Conditions and If statements

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

Python Comparison Operators

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

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

Python Logical Operators

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

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

Python if

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

Python if else

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

Python if elif else

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

Python if elif else

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

Python for Loops

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

```
1
2
3
4
5
```

Python for loops

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

Python while Loops

- while
  - break
  - continue

Python while loops

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

Python Functions, Classes, and Modules

Python Functions, Classes, Modules

- Python Functions
  - def myfunction():
- Python Classes/Objects
  - class MyClass:
- Python Modules
  - mymodule.py
  - import mymodule

Python Functions

Python Functions

- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- A function can return data as a result.
- Creating a Function
  - In Python a function is defined using the def keyword:

Python Function def

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

```
194.87
```

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

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

How much is your $100 worth after 7 years?

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

```
fv = 194.87
```

Future Value

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

```
194.87
```

Future Value

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

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

```
194.87
```

Python Function

def getfv() define get future value function

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

```
194.87
```

Python Classes/Objects

```
class MyClass:
```

Python Classes/Objects

- Python is an object oriented programming language.
- Almost everything in Python is an object, with its properties and methods.
- A Class is like an object constructor, or a "blueprint" for creating objects.
- Create a Class:
  - To create a class, use the keyword class:

Python Classes/Objects

class MyClass:

```
# Python class
class MyClass:
x = 5


c1 = MyClass()
print(c1.x)
```

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

Python Classes/Objects

```
class Person:
    def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("Alan", 20)

print(p1.name)
print(p1.age)
```

```
Alan
20
```

Python Classes/Objects

```
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()
```

Python Classes/Objects

```
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
```

Python Classes and Obects

```
class Vehicle:
    name = ""
    kind = "car"
    color = ""
    value = 100.00
    def description(self):
        desc_str = "%s is a %s %s worth $%.2f." %
(self.name, self.color, self.kind, self.value)
        return desc_str
```

Python Classes and Objects

```
car1 = Vehicle()
car1.name = "Fer"
car1.color = "red"
car1.kind = "convertible"
car1.value = 60000.00

car2 = Vehicle()
car2.name = "Jump"
car2.color = "blue"
car2.kind = "van"
car2.value = 10000.00

print(car1.description())
print(car1.name)
print(car2.description())
print(car2.name)
```

```
class Vehicle:
      name = ""
      kind = "car"
      color = ""
      value = 100.00
      def description(self):
            desc_str = "%s is a %s %s
worth $%.2f." % (self.name, self.color,
self.kind, self.value)
            return desc_str
```

```
Fer is a red convertible worth $60000.00.
Fer
Jump is a blue van worth $10000.00.
Jump
```

https://www.learnpython.org/en/Classes_and_Objects

Python Modules

Python Modules

- Consider a module to be the same as a code library.
- A file containing a set of functions you want to include in your application.
- Create a Module
  - To create a module just save the code you want in a file with the file extension .py:
- Use a Module
  - import module

Python Modules

```
# mymodule.py
def greeting(name):
    print("Hello, " + name)
```

```
import mymodule
mymodule.greeting("Alan")
```

```
mymodule.py
def greeting(name):
     print("Hello, " + name)
```

https://tinyurl.com/aintpupython101

Python File Input / Output

```
# Python 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)
```

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

Python File Input / Output

```
# Python File Input / Output
filename = 'mymodule.py'
with open(filename, 'w') as file:
     text = '''def greeting(name):
     print("Hello, " + name)
     '''
     file.write(text)

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

```
mymodule.py
def greeting(name):
     print("Hello, " + name)
```

https://tinyurl.com/aintpupython101

Python Modules

import mymodule

```
# mymodule.py
def greeting(name):
    print("Hello, " + name)
```

```
import mymodule
mymodule.greeting("Alan")
```

```
Hello, Alan
```

Python `main()` function

```
#Python main() function
def main():
    print("Hello World!")

if __name__ == "__main__":
    main()
```

Files
and
Exception Handling

Files and Exception Handling

- Python Files (File Handling)
  - open()
  - f = open("myfile.txt")
- Python Try Except (Exception Handling)
  - try:
    except:
    else:
    finally:

File Handling

- The key function for working with files in Python is the open() function.
- The open() function takes two parameters; filename, and mode.
- There are four different methods (modes) for opening a file:
  - "r" - Read - Default value. Opens a file for reading, error if the file does not exist
  - "a" - Append - Opens a file for appending, creates the file if it does not exist
  - "w" - Write - Opens a file for writing, creates the file if it does not exist
  - "x" - Create - Creates the specified file, returns an error if the file exists

Python Files (File Handling)

```
f = open("myfile.txt", "w")
f.write("Hello World")
f.close()

f = open("myfile.txt", "r")
text = f.read()
print(text)
f.close()
```

```
Hello World
```

Python Files (File Handling)

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

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

```
Hello World
```

Python Files

```
# Python File Input / Output
with open('myfile.txt', 'w') as file:
    file.write('Hello World\nPython File IO')

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

```
Hello World
Python File IO
```

https://tinyurl.com/aintpupython101

Python Files

```
# Python 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)
```

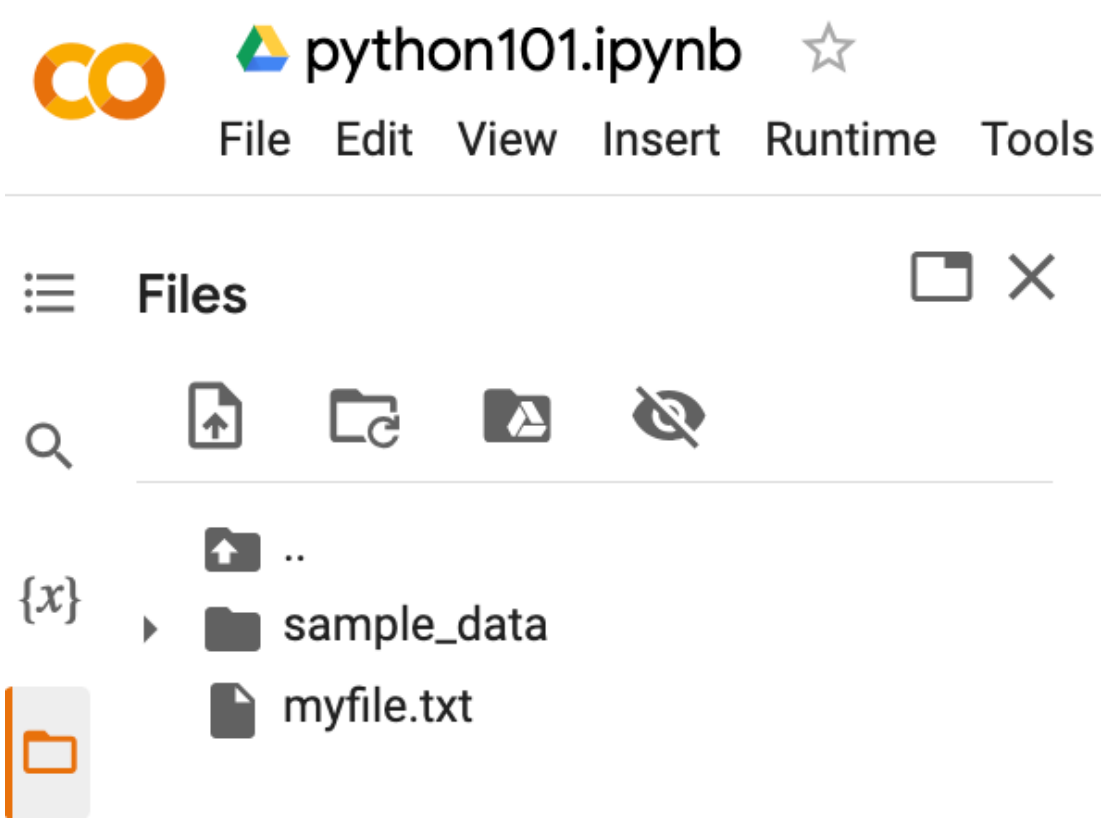
```
Hello World
Python File IO
New line
```

https://tinyurl.com/aintpupython101

Python Files

```
# !ls list files
!ls
```

```
myfile.txt sample_data
```



Python OS, IO, files, and Google Drive

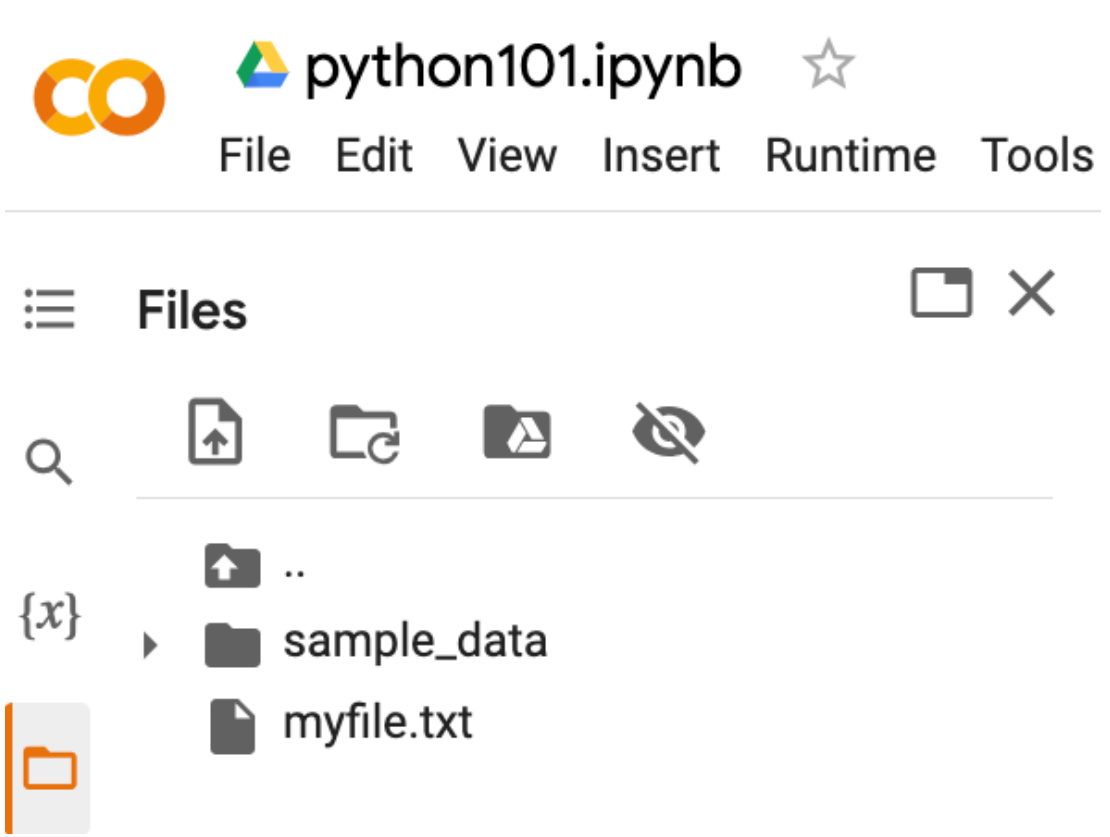
```
import os


cwd = os.getcwd()
print(cwd)
```

```
/content
```

os.listdir()

```
os.listdir(cwd)
```

```
['.config',
'myfile.txt',
'sample_data']
```

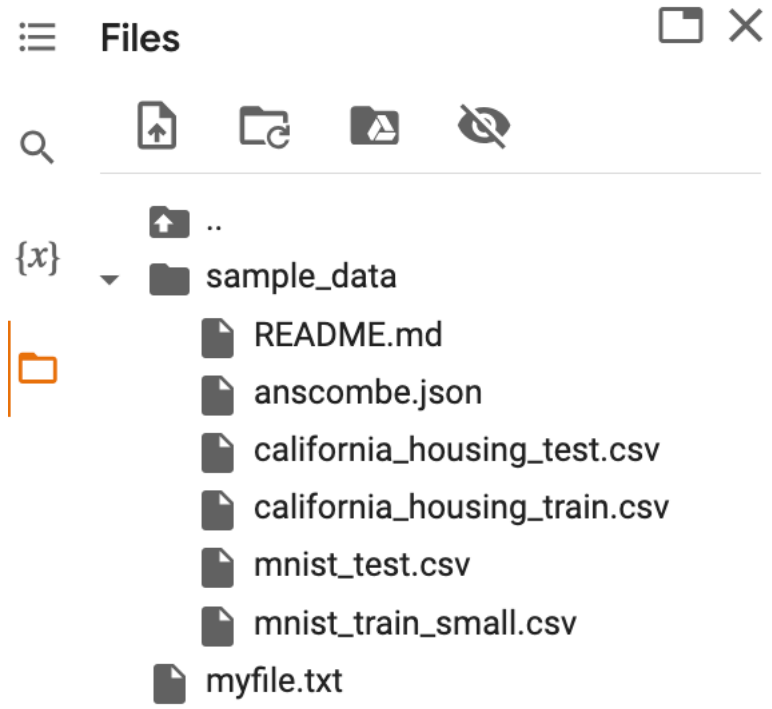


https://tinyurl.com/aintpupython101

os.path.join()

```
path = os.path.join(cwd, 'sample_data')
print(path)
os.listdir(path)
```

```
/content/sample_data
['README.md', 'anscombe.json',
'mnist_train_small.csv',
'mnist_test.csv',
'california_housing_train.csv',
'california_housing_test.csv']
```



from google.colab import files

```
from google.colab import files

with open('io_file_myday.txt', 'w') as f:
    f.write('Google Colab File Write Text some content Myday')

import time
time.sleep(1) # time sleep 1 second

files.download('io_file_myday.txt')
print('downloaded')
```

```
downloaded
```

Python Files

```
from google.colab import files
uploaded = files.upload()

for fn in uploaded.keys():
    print('User uploaded file "{name}"
with length {length} bytes'.format(
name=fn, length=len(uploaded[fn])))
```

User uploaded file "io_file_myday2.txt" with length 47 bytes

os.remove()

```
import os
if os.path.exists("myfile.txt"):
    os.remove("myfile.txt")
    print("myfile.txt removed")
else:
    print("The file does not exist")
```

```
myfile.txt removed
```

os.mkdir("myfolder1")
os.rmdir("myfolder1")

```
import os
os.listdir()
os.mkdir("myfolder1")
os.listdir()
os.rmdir("myfolder1")
os.listdir()
```

Python Try Except

- The **try** block lets you test a block of code for errors.
- The **except** block lets you handle the error.
- The **else** block lets you execute code when there is no error.
- The **finally** block lets you execute code, regardless of the result of the try- and except blocks.

Python Try Except (Exception Handling)

try: except:

```
#Python try except
try:
    print(x)
except:
    print("Exception Error")
```

Python try: except: finally:

```
#Python try except finally
try:
    print("Hello")
except:
    print("Exception Error")
finally:
    print("Finally process")
```

```
Hello
Finally process
```

Python try: except: else:

```
#Python try except else
try:
    print("Hello")
except:
    print("Exception Error")
else:
    print("No exception")
```

```
Hello
No exception
```

Python try: except: else: finally:

```
try:
    print("Hello")
except:
    print("Exception Error")
else:
    print("No exception")
finally:
    print("Finally process")
```

```
Hello
No exception
Finally process
```

Python try: except: else: finally:

```
try:
    price = float(input("Enter the price of the stock (e.g. 10):"))
    shares = int(input("Enter the number of shares (e.g. 2):"))
    total = price * shares
except Exception as e:
    print("Exception error:", str(e))
else:
    print("The total value of the shares is:", total)
finally:
    print("Thank you.")
```

```
Enter the price of the stock (e.g. 10):10
Enter the number of shares (e.g. 2):2
The total value of the shares is: 20.0
Thank you.
```

Python try: except: else: finally:

```
try:
    file = open("myfile.txt")
    file.write("Python write file")
    print("file saved")
except:
    print("Exception file Error")
```

```
Exception file Error
```

Python try: except: else: finally:

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

```
Exception file Error
Finally process
```

Python try: except: else: finally:

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

```
file saved
Finally process
```

Data Analytics and Visualization with Python

Data Analytics and Visualization with Python

- NumPy
  - Numerical Python N-dimensional array
- Pandas
  - Data Analytics
- Matplotlib
  - Basic Data Visualization
- Seaborn
  - Advanced Visualization

W3Schools Python Numpy

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

Python Modules

- NumPy Tutorial
- Pandas Tutorial
- SciPy Tutorial
- Django Tutorial

NumPy Tutorial

- NumPy HOME
- NumPy Intro
- NumPy Getting Started
- NumPy Creating Arrays
- NumPy Array Indexing
- NumPy Array Slicing
- NumPy Data Types
- NumPy Copy vs View
- NumPy Array Shape
- NumPy Array Reshape
- NumPy Array Iterating
- NumPy Array Join
- NumPy Array Split
- NumPy Array Search
- NumPy Array Sort
- NumPy Array Filter

NumPy Random

- Random Intro
- Data Distribution

NumPy Tutorial

❮ Home Next ❯

NumPy is a Python library.

NumPy is used for working with arrays.

NumPy is short for "Numerical Python".

Learning by Reading

We have created 43 tutorial pages for you to learn more about NumPy.

Starting with a basic introduction and ends up with creating and plotting random data sets, and working with NumPy functions:

Basic Random ufunc

https://www.w3schools.com/python/numpy/default.asp

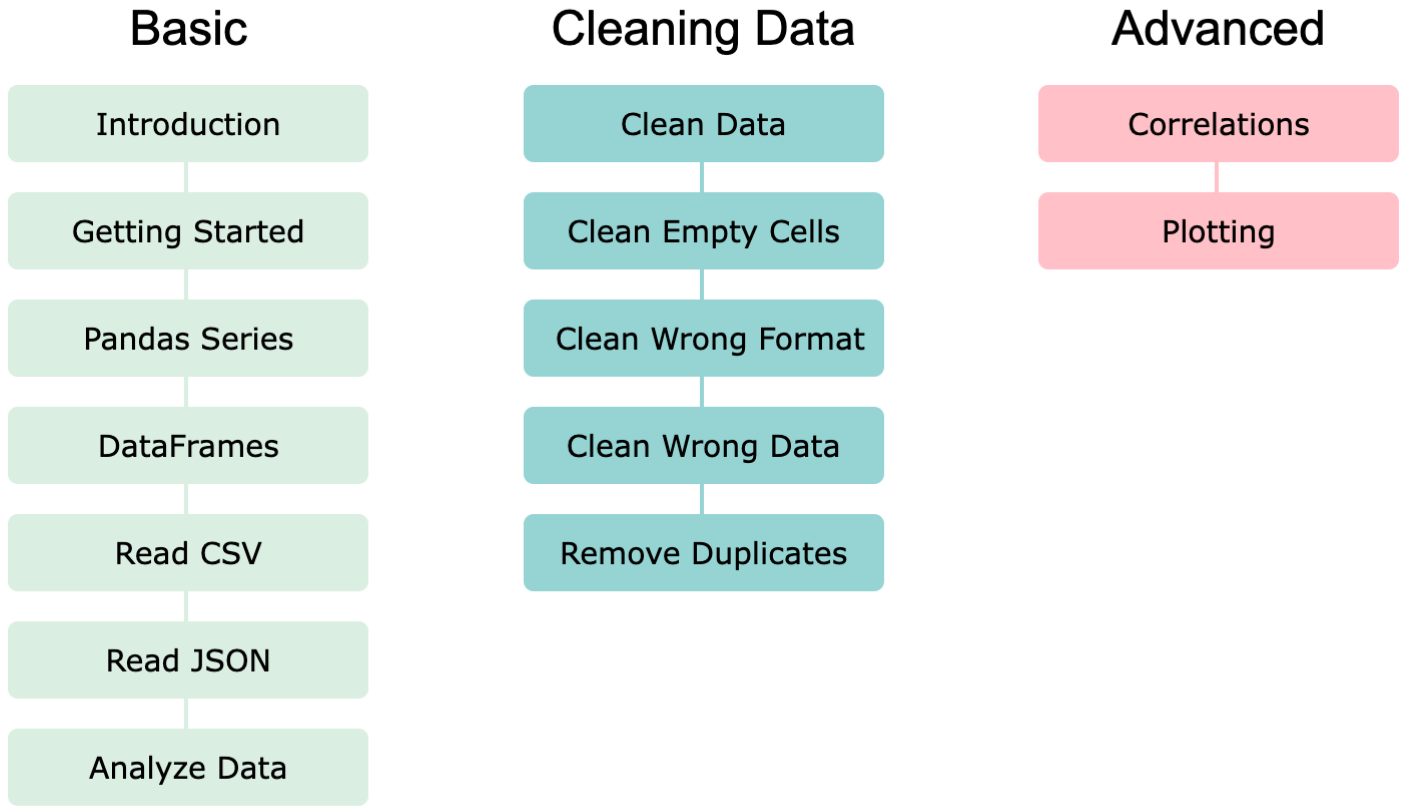
W3Schools Python Pandas

Learning by Reading

Pandas Tutorial

We have created 14 tutorial pages for you to learn more about Pandas.

Starting with a basic introduction and ends up with cleaning and plotting data:



Python Modules

NumPy Tutorial
Pandas Tutorial
SciPy Tutorial
Django Tutorial

Pandas Tutorial

Pandas HOME
Pandas Intro
Pandas Getting Started
Pandas Series
Pandas DataFrames
Pandas Read CSV
Pandas Read JSON
Pandas Analyzing Data

Cleaning Data

Cleaning Data
Cleaning Empty Cells
Cleaning Wrong Format
Cleaning Wrong Data
Removing Duplicates

Correlations

Pandas Correlations

Plotting

Pandas Plotting

https://www.w3schools.com/python/pandas/default.asp

W3Schools Python

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

Python Modules

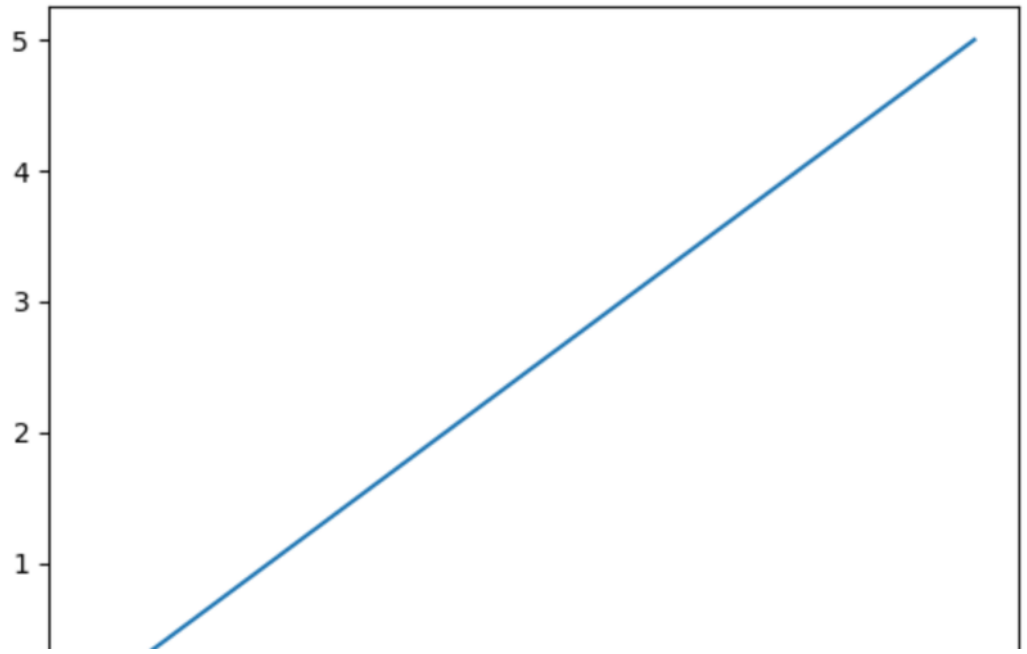
NumPy Tutorial
Pandas Tutorial
SciPy Tutorial
Django Tutorial

Python Matplotlib

Matplotlib Intro
Matplotlib Get Started
Matplotlib Pyplot
Matplotlib Plotting
Matplotlib Markers
Matplotlib Line
Matplotlib Labels
Matplotlib Grid
Matplotlib Subplot
Matplotlib Scatter
Matplotlib Bars
Matplotlib Histograms
Matplotlib Pie Charts

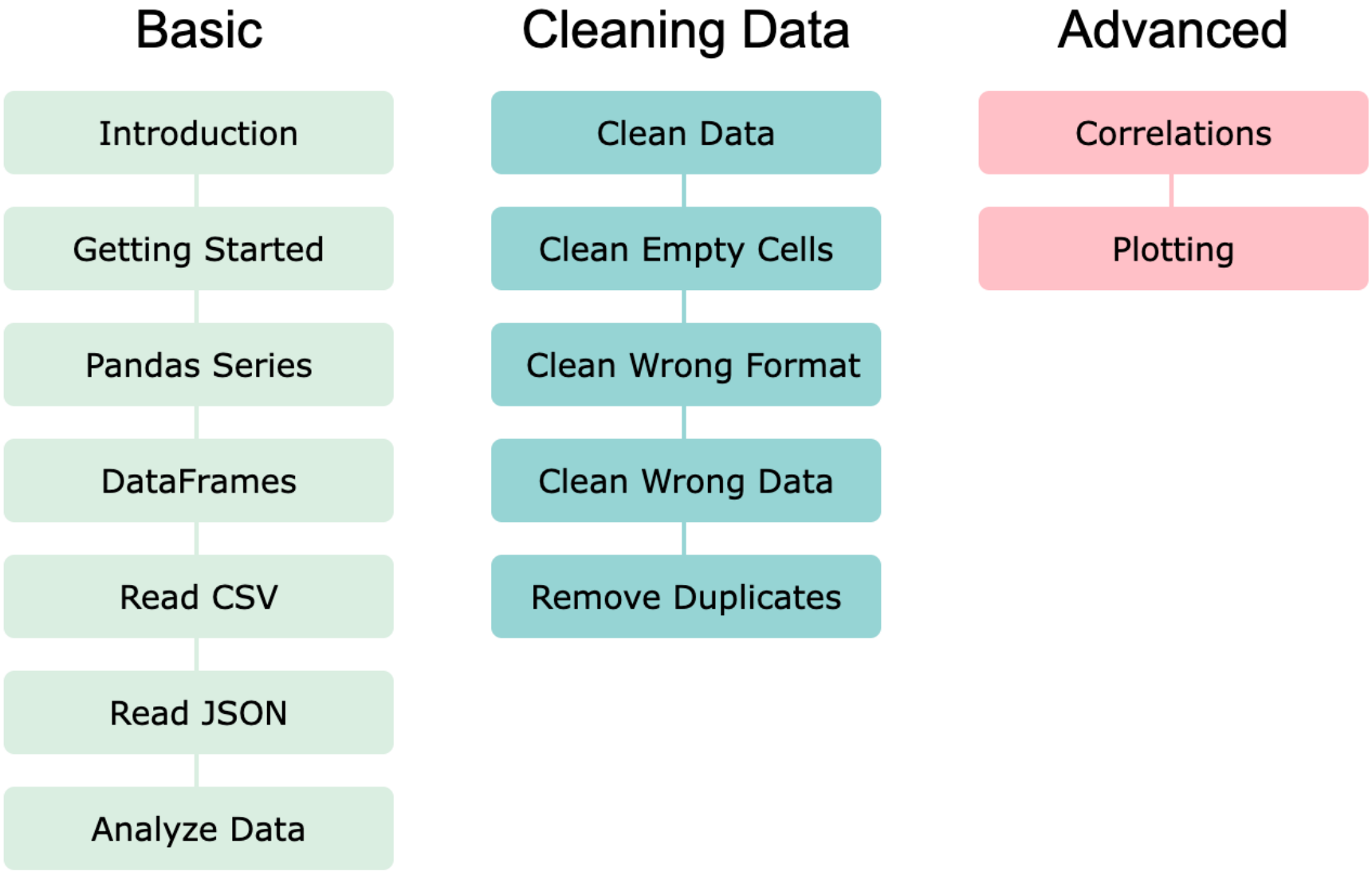
Matplotlib Tutorial

❮ Previous Next ❯



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

Pandas: Data Analytics and Visualization



https://www.w3schools.com/python/pandas/default.asp

Wes McKinney (2022), "Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter", 3rd Edition, O'Reilly Media.

wesm / pydata-book Public

Notifications | Fork 14.1k | Star 19k

Code | Issues 2 | Pull requests 1 | Actions | Projects | Wiki | Security | Insights

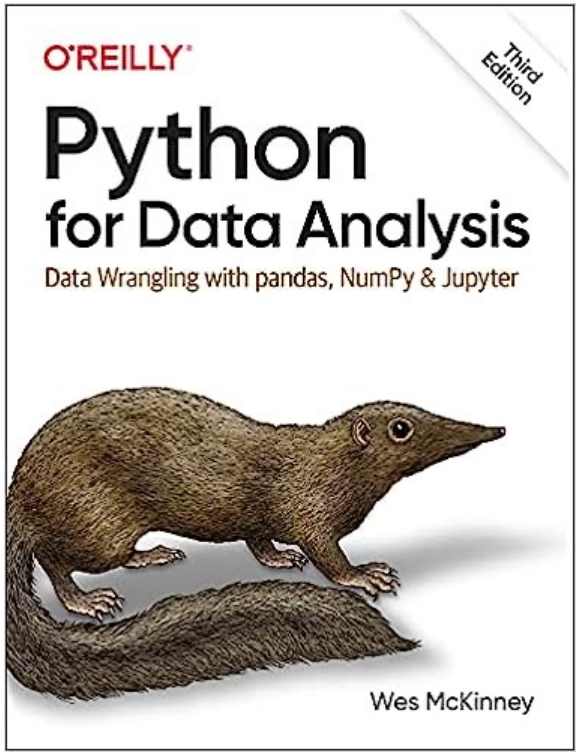
3rd-edition | 3 branches | 0 tags | Go to file | Code

wesm Upload cleaner notebook files without internal build toolchai... ... f1757b8 3 days ago 70 commits

datasets	Add fec.parquet	10 months ago
examples	Simplifying datasets	10 months ago
.gitignore	Add gitignore	8 years ago
COPYING	Update COPYING	4 months ago
README.md	Update notebooks in advance of publication	7 months ago
appa.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
appb.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch02.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch03.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch04.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch05.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch06.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago

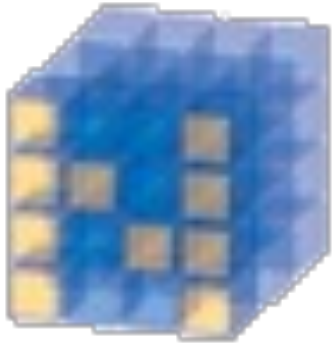
About

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



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

Numpy



NumPy

Base

N-dimensional array

package

NumPy

is the

fundamental package

for

scientific computing

with Python.

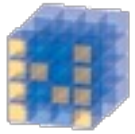
Source: http://www.numpy.org/

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.**

Source: Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly



NumPy

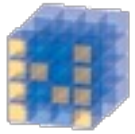
NumPy ndarray

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**

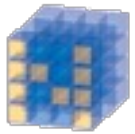


NumPy

NumPy

```
v = list(range(1, 6))
v
2 * v
import numpy as np
v = np.arange(1, 6)
v
2 * v
```

Source: Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly



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])
```

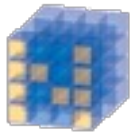
Python Data Structures

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

Lists []

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

```
4
60
70
90
```



NumPy

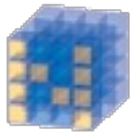
NumPy Create Array

```
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])
```

Source: Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly



NumPy

NumPy

```python
import numpy as np

a = np.zeros((2,2))  # Create an array of all zeros
print(a)             # Prints "[[ 0.  0.]
                     #          [ 0.  0.]]"

b = np.ones((1,2))   # Create an array of all ones
print(b)             # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7) # Create a constant array
print(c)               # Prints "[[ 7.  7.]
                       #          [ 7.  7.]]"

d = np.eye(2)        # Create a 2x2 identity matrix
print(d)             # Prints "[[ 1.  0.]
                     #          [ 0.  1.]]"

e = np.random.random((2,2)) # Create an array filled with random values
print(e)                    # Might print "[[ 0.91940167  0.08143941]
                            #               [ 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]]
```

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

```
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
```

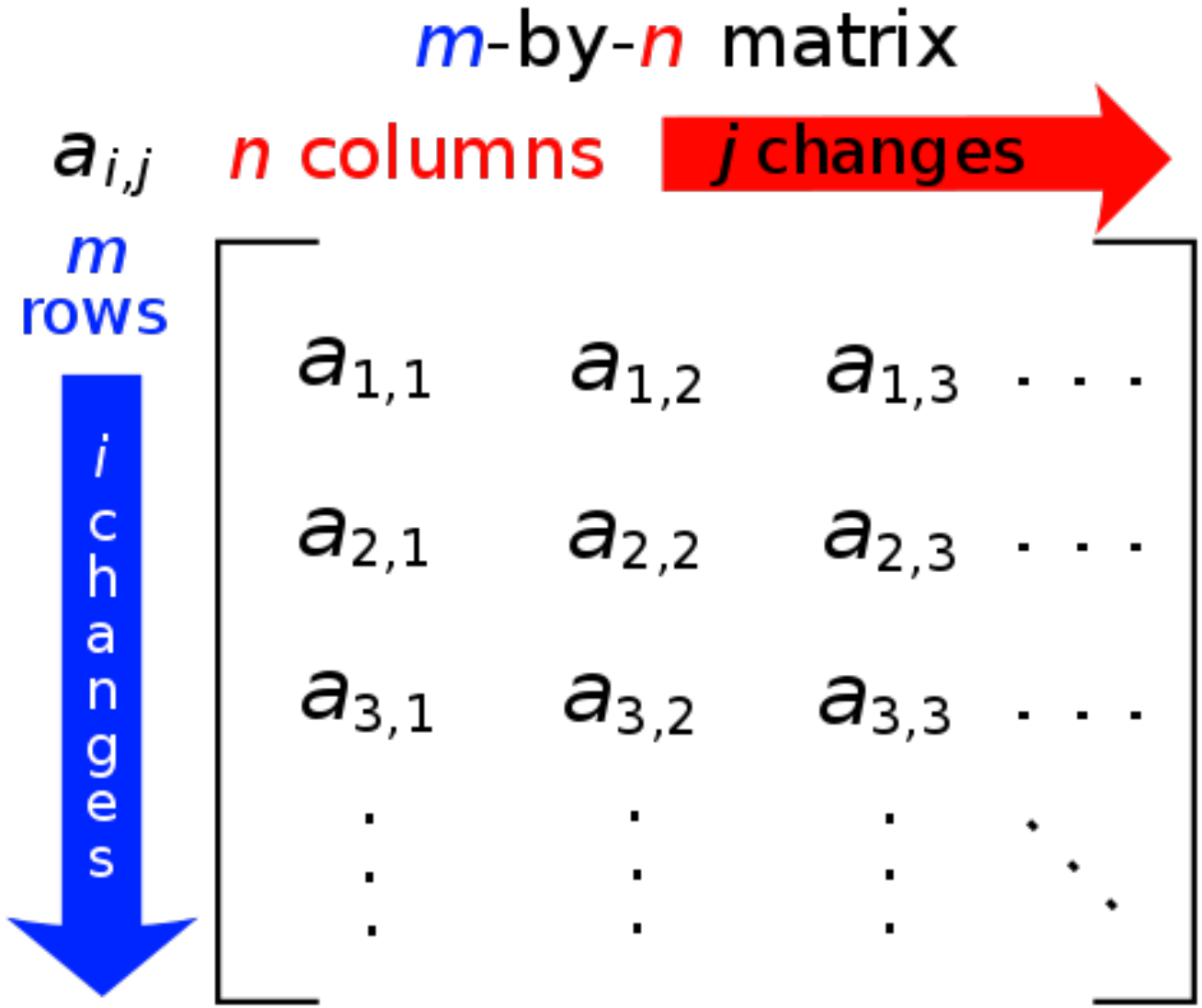
```
2
```

```
a.dtype.name
```

```
'int64'
```

Source: https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

Matrix



Source: https://simple.wikipedia.org/wiki/Matrix_(mathematics)

NumPy ndarray:
Multidimensional Array Object

NumPy ndarray

One-dimensional Array (1-D Array)

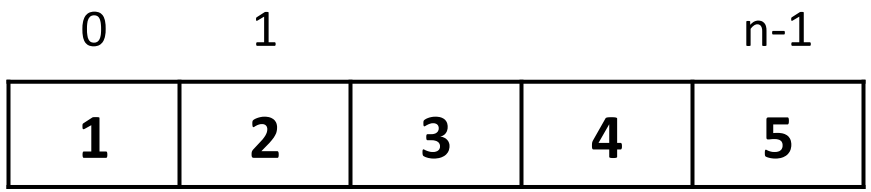
0	1			n-1
1	**2**	**3**	**4**	**5**

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)
```

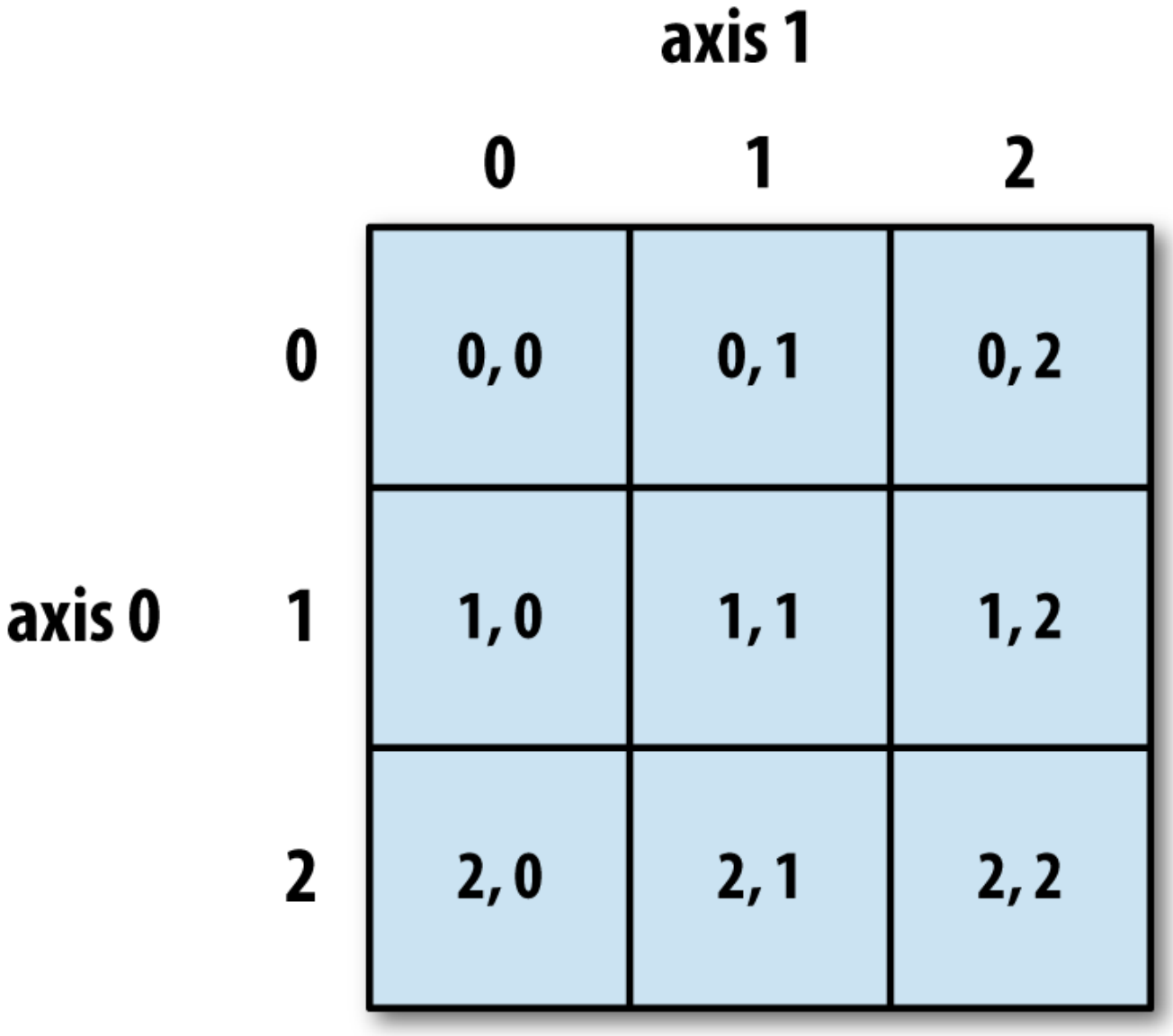
```
(3, 4)
```

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

NumPy Basics: Arrays and Vectorized Computation

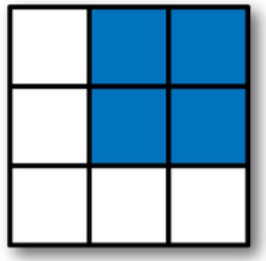
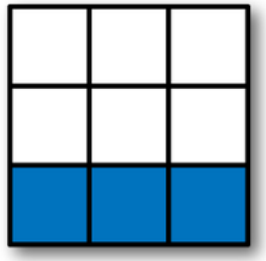
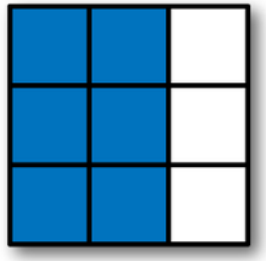
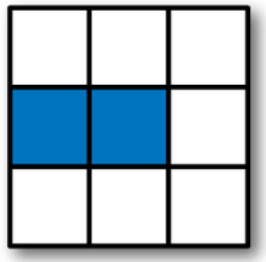
Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

NumPy Array



Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

Numpy Array

	Expression	Shape
	`arr[:2, 1:]`	`(2, 2)`
	`arr[2]`	`(3,)`
	`arr[2, :]`	`(3,)`
	`arr[2:, :]`	`(1, 3)`
	`arr[:, :2]`	`(3, 2)`
	`arr[1, :2]`	`(2,)`
	`arr[1:2, :2]`	`(1, 2)`

Source: https://www.safaribooksonline.com/library/view/python-for-data/9781449323592/ch04.html

TensorFlow

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]

https://www.tensorflow.org/

Scalar 80

Vector [50 60 70]

Matrix

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

Tensor

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

pandas

Python Data Analysis Library

providing high-performance, easy-to-use
data structures and data analysis tools
for the Python programming language.

Source: http://pandas.pydata.org/

pandas

pandas:
powerful Python data analysis toolkit

- **Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet**
- **Ordered and unordered (not necessarily fixed-frequency) time series data.**
- **Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels**
- **Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure**

Source: http://pandas.pydata.org/pandas-docs/stable/

pandas

Series
DataFrame

- Primary data structures of pandas
  - Series (1-dimensional)
  - DataFrame (2-dimensional)
- Handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering.

Source: http://pandas.pydata.org/pandas-docs/stable/

pandas

pandas DataFrame

- **DataFrame provides everything that R's data.frame provides and much more.**
- **pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.**

pandas

pandas
Comparison with SAS

pandas	SAS
DataFrame	data set
column	variable
row	observation
groupby	BY-group
NaN	.

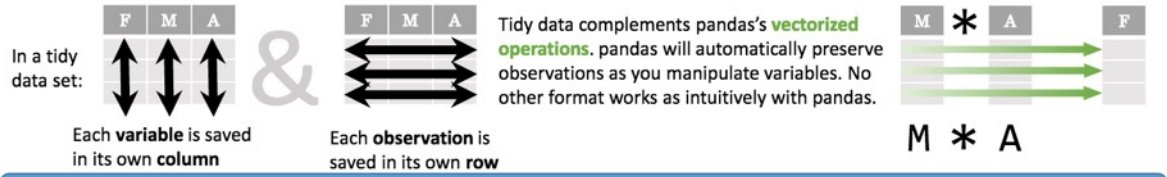
Source: http://pandas.pydata.org/pandas-docs/stable/comparison_with_sas.html

Python Pandas Cheat Sheet

Data Wrangling with pandas Cheat Sheet

http://pandas.pydata.org

Tidy Data – A foundation for wrangling in pandas



Tidy data complements pandas's vectorized operations. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

Syntax – Creating DataFrames

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
       index = [1, 2, 3])
```
Specify values for each column.

```
df = pd.DataFrame(
    [[4, 7, 10],
     [5, 8, 11],
     [6, 9, 12]],
    index=[1, 2, 3],
    columns=['a', 'b', 'c'])
```
Specify values for each row.

		a	b	c
n	v			
d	1	4	7	10
	2	5	8	11
e	2	6	9	12

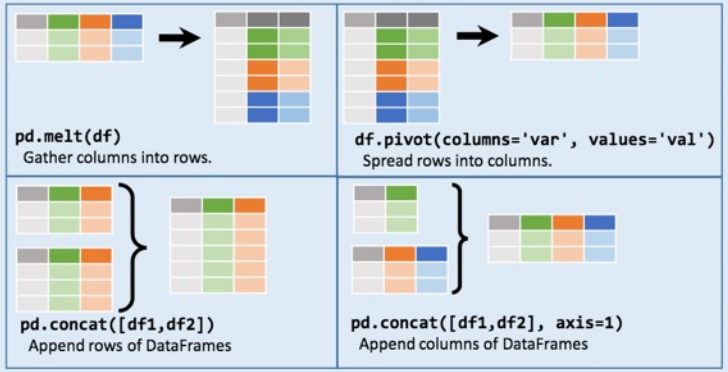
```
df = pd.DataFrame(
        {"a" : [4 ,5, 6],
         "b" : [7, 8, 9],
         "c" : [10, 11, 12]},
index = pd.MultiIndex.from_tuples(
        [('d',1),('d',2),('e',2)],
          names=['n','v'])))
```
Create DataFrame with a MultiIndex

Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = (pd.melt(df)
        .rename(columns={
                'variable' : 'var',
                'value' : 'val'})
        .query('val >= 200')
    )
```

Reshaping Data – Change the layout of a data set



pd.melt(df)
Gather columns into rows.

df.pivot(columns='var', values='val')
Spread rows into columns.

pd.concat([df1,df2])
Append rows of DataFrames

pd.concat([df1,df2], axis=1)
Append columns of DataFrames

df=df.sort_values('mpg')
Order rows by values of a column (low to high).

df=df.sort_values('mpg',ascending=False)
Order rows by values of a column (high to low).

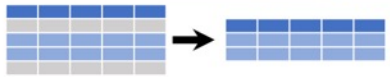
df=df.rename(columns = {'y':'year'})
Rename the columns of a DataFrame

df=df.sort_index()
Sort the index of a DataFrame

df=df.reset_index()
Reset index of DataFrame to row numbers, moving index to columns.

df=df.drop(['Length','Height'], axis=1)
Drop columns from DataFrame

Subset Observations (Rows)



df[df.Length > 7]
Extract rows that meet logical criteria.

df.drop_duplicates()
Remove duplicate rows (only considers columns).

df.head(n)
Select first n rows.

df.tail(n)
Select last n rows.

df.sample(frac=0.5)
Randomly select fraction of rows.

df.sample(n=10)
Randomly select n rows.

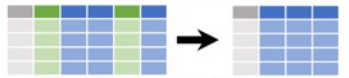
df.iloc[10:20]
Select rows by position.

df.nlargest(n, 'value')
Select and order top n entries.

df.nsmallest(n, 'value')
Select and order bottom n entries.

Logic in Python (and pandas)			
<	Less than	!=	Not equal to
>	Greater than	df.column.isin(*values*)	Group membership
==	Equals	pd.isnull(*obj*)	Is NaN
<=	Less than or equals	pd.notnull(*obj*)	Is not NaN
>=	Greater than or equals	&,|,~,^,df.any(),df.all()	Logical and, or, not, xor, any, all

Subset Variables (Columns)



df[['width','length','species']]
Select multiple columns with specific names.

df['width'] *or* **df.width**
Select single column with specific name.

df.filter(regex='*regex*')
Select columns whose name matches regular expression *regex*.

regex (Regular Expressions) Examples	
'\.'	Matches strings containing a period '.'
'Length$'	Matches strings ending with word 'Length'
'^Sepal'	Matches strings beginning with the word 'Sepal'
'^x[1-5]$'	Matches strings beginning with 'x' and ending with 1,2,3,4,5
''^(?!Species$).*'	Matches strings except the string 'Species'

df.loc[:,'x2':'x4']
Select all columns between x2 and x4 (inclusive).

df.iloc[:,[1,2,5]]
Select columns in positions 1, 2 and 5 (first column is 0).

df.loc[df['a'] > 10, ['a','c']]
Select rows meeting logical condition, and only the specific columns .

http://pandas.pydata.org/ This cheat sheet inspired by Rstudio Data Wrangling Cheatsheet (https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf) Written by Irv Lustig, Princeton Consultants

Source: https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf

Creating pd.DataFrame

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

In [1]:
```
import numpy as np
import pandas as pd
df = pd.DataFrame({"a": [4, 5, 6],
                   "b": [7, 8, 9],
                   "c": [10, 11, 12]},
                  index = [1, 2, 3])
df
```

Out[1]:

	a	b	c
1	4	7	10
2	5	8	11
3	6	9	12

```
import pandas as pd
df = pd.DataFrame({"a": [4, 5, 6],
                   "b": [7, 8, 9],
                   "c": [10, 11, 12]},
                  index = [1, 2, 3])
```

Source: https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf

Pandas DataFrame

```
type(df)
```

```
type(df)
```

```
pandas.core.frame.DataFrame
```

pandas

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
print('pandas imported')
```

```
s = pd.Series([1,3,5,np.nan,6,8])
s
```

```
dates = pd.date_range('20181001',
periods=6)
dates
```

Source: http://pandas.pydata.org/pandas-docs/stable/10min.html

pandas

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 print('pandas imported')
```

```
pandas imported
```

```
1 s = pd.Series([1,3,5,np.nan,6,8])
2 s
```

```
0    1.0
1    3.0
2    5.0
3    NaN
4    6.0
5    8.0
dtype: float64
```

```
1 dates = pd.date_range('20181001', periods=6)
2 dates
```

```
DatetimeIndex(['2018-10-01', '2018-10-02', '2018-10-03', '2018-10-04',
               '2018-10-05', '2018-10-06'],
              dtype='datetime64[ns]', freq='D')
```

pandas

```
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
df
```

```
1 df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
2 df
```

	A	B	C	D
2018-10-01	-0.336188	0.584621	-1.061433	-0.036278
2018-10-02	0.903683	-0.839723	-0.270219	-1.099606
2018-10-03	0.920208	-0.240353	-0.818598	-1.105489
2018-10-04	0.221045	-0.314589	0.042071	-1.447280
2018-10-05	0.946862	-1.570305	-1.009180	-0.375659
2018-10-06	-0.225148	0.510691	2.002372	-0.335005

```
df = pd.DataFrame(np.random.randn(3,5),
index=['student1','student2','student3']
, columns=list('ABCDE'))
df
```

```
1 df = pd.DataFrame(np.random.randn(3,5), index=['student1','student2','student3'], columns=list('ABCDE'))
2 df
```

	A	B	C	D	E
student1	-0.346884	-1.232934	-0.302072	-1.345084	-0.723880
student2	1.090955	-0.010483	1.280072	-0.253958	-0.030604
student3	0.325660	0.808956	-0.395820	-1.498926	1.603471

pandas

```
df2 = pd.DataFrame({ 'A' : 1.,
'B' : pd.Timestamp('20181001'),
'C' : pd.Series(2.5,index=list(range(4)),dtype='float32'),
'D' : np.array([3] * 4,dtype='int32'),
'E' : pd.Categorical(["test","train","test","train"]),
'F' : 'foo' })
df2
```

```
1 df2 = pd.DataFrame({ 'A' : 1.,
2 'B' : pd.Timestamp('20181001'),
3 'C' : pd.Series(2.5,index=list(range(4)),dtype='float32'),
4 'D' : np.array([3] * 4,dtype='int32'),
5 'E' : pd.Categorical(["test","train","test","train"]),
6 'F' : 'foo' })
7 df2
```

	A	B	C	D	E	F
0	1.0	2018-10-01	2.5	3	test	foo
1	1.0	2018-10-01	2.5	3	train	foo
2	1.0	2018-10-01	2.5	3	test	foo
3	1.0	2018-10-01	2.5	3	train	foo

pandas

df2.dtypes

```
df2.dtypes
```

```
A             float64
B    datetime64[ns]
C             float32
D               int32
E            category
F              object
dtype: object
```

Python Data Analysis and Visualization

Python Pandas

http://pandas.pydata.org/

Python matplotlib

Source: https://matplotlib.org/

Python seaborn

seaborn

Source: https://seaborn.pydata.org/

Python plotly

Source: https://plotly.com/python/

Python bokeh

Source: https://bokeh.org/

Python matplotlib

matplotlib
Version 3.3.4

Fork me on GitHub

Installation Documentation Examples Tutorials Contributing Search

home | contents » Matplotlib: Python plotting modules | index

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Matplotlib makes easy things easy and hard things possible.

Create

- Develop publication quality plots with just a few lines of code
- Use interactive figures that can zoom, pan, update...

Customize

- Take full control of line styles, font properties, axes properties...
- Export and embed to a number of file formats and interactive environments

Extend

- Explore tailored functionality provided by third party packages
- Learn more about Matplotlib through the many external learning resources

Latest stable release
3.3.4: docs | changelog

Last release for Python 2
2.2.5: docs | changelog

Development version
docs

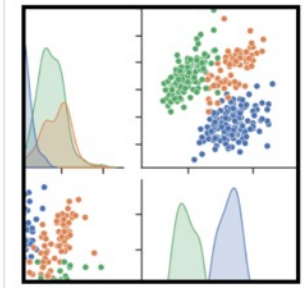
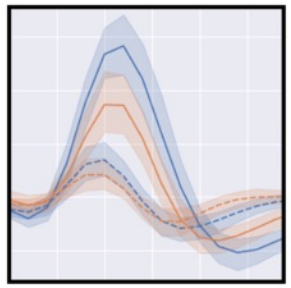
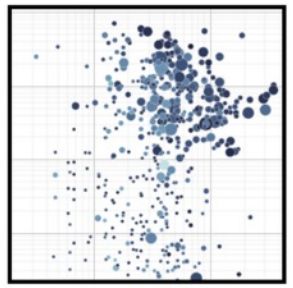
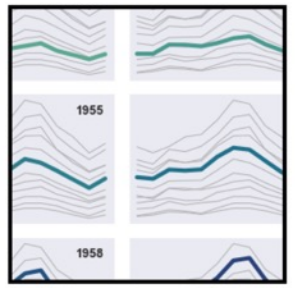
Support Matplotlib

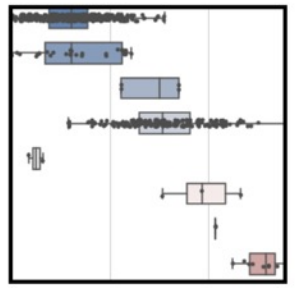
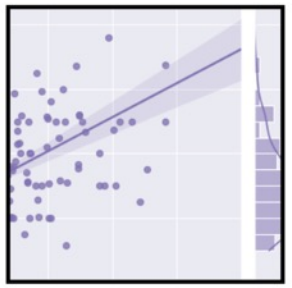
https://matplotlib.org/

Python Seaborn

seaborn 0.11.1 Gallery Tutorial API Site Page Search

seaborn: statistical data visualization

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

For a brief introduction to the ideas behind the library, you can read the introductory notes. Visit the installation page to see how you can download the package and get started with it. You can browse the example gallery to see what you can do with seaborn, and then check out the tutorial and API reference to find out how.

To see the code or report a bug, please visit the GitHub repository. General support questions are most at home on stackoverflow or discourse, which have dedicated channels for seaborn.

Contents

- Introduction
- Release notes
- Installing
- Example gallery
- Tutorial
- API reference

Features

- Relational: API | Tutorial
- Distribution: API | Tutorial
- Categorical: API | Tutorial
- Regression: API | Tutorial
- Multiples: API | Tutorial
- Style: API | Tutorial
- Color: API | Tutorial

© Copyright 2012-2020, Michael Waskom. Created using Sphinx 3.3.1.

Back to top

https://seaborn.pydata.org/

plotly

Python Plotly Graphing Library

plotly | Graphing Libraries

Star 9,085 DO MORE WITH DASH

Search...

Quick Start

- Getting Started
- Is Plotly Free?
- Figure Reference
- API Reference
- Dash
- GitHub
- community.plotly.com

Examples

- Fundamentals
- Basic Charts
- Statistical Charts
- Artificial Intelligence and Machine Learning
- Scientific Charts
- Financial Charts
- Maps
- 3D Charts

Plotly Python Open Source Graphing Library

Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.
Plotly.py is free and open source and you can view the source, report issues or contribute on GitHub.

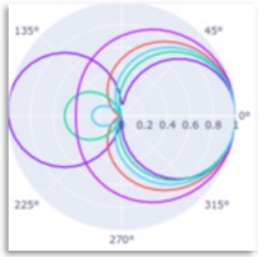
Our recommended IDE for Plotly's Python graphing library is Dash Enterprise's Data Science Workspaces, which has both Jupyter notebook and Python code file support.
Find out if your company is using Dash Enterprise.

Install Dash Enterprise on Azure | ***Install Dash Enterprise on AWS***

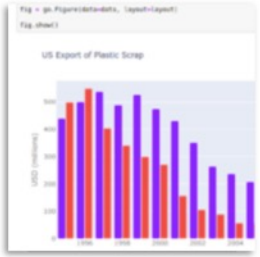
Fundamentals

More Fundamentals »

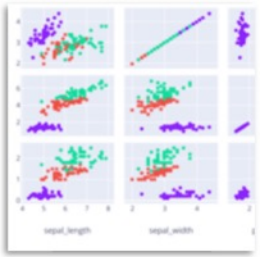
The Figure Data Structure



Creating and Updating Figures



Displaying Figures



Plotly Express

Analytical Apps with Dash

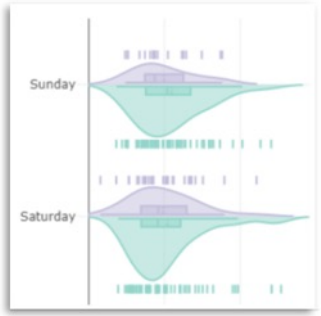
https://plotly.com/python/

plotly

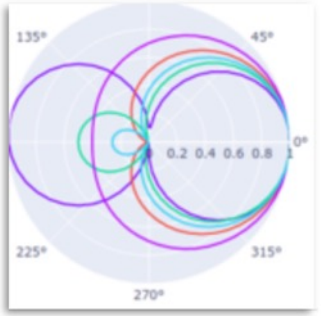
Python Plotly Graphing Library

Fundamentals

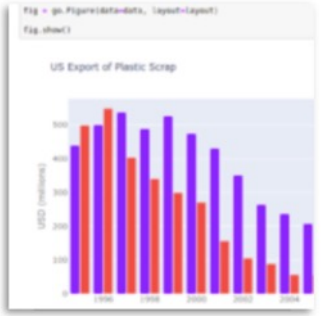
More Fundamentals »



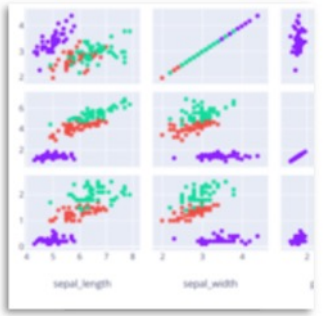
The Figure Data Structure



Creating and Updating Figures



Displaying Figures

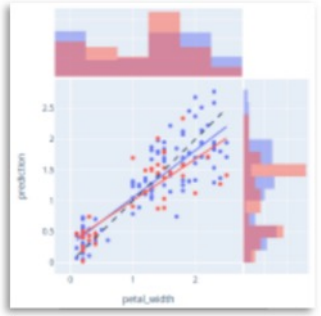


Plotly Express

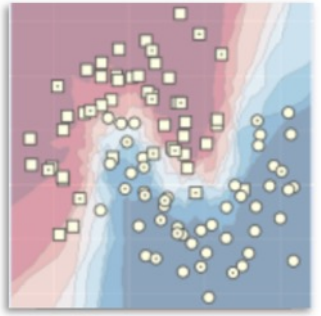
Analytical Apps with Dash

Artificial Intelligence and Machine Learning

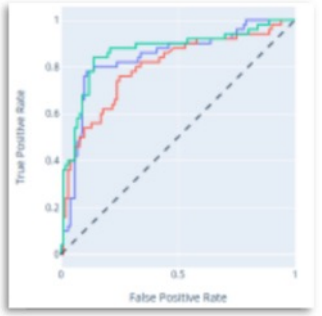
More AI and ML »



ML Regression



kNN Classification



ROC and PR Curves

PCA Visualization

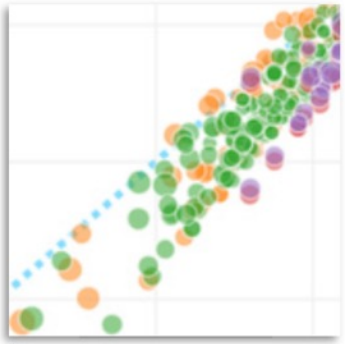
AI/ML Apps with Dash

https://plotly.com/python/

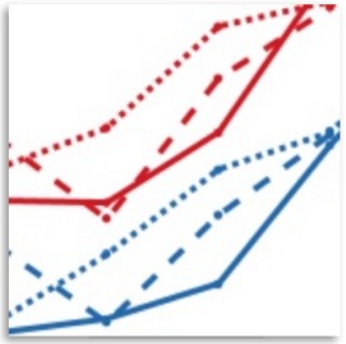
Python Plotly Graphing Library

Basic Charts

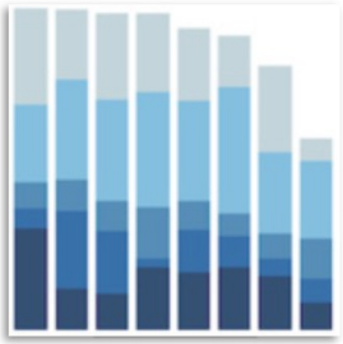
More Basic Charts »



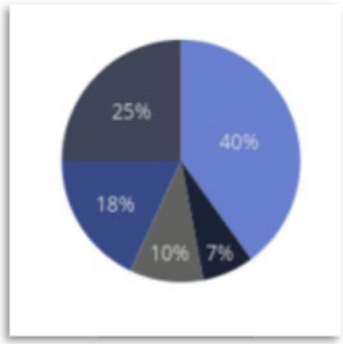
Scatter Plots



Line Charts



Bar Charts

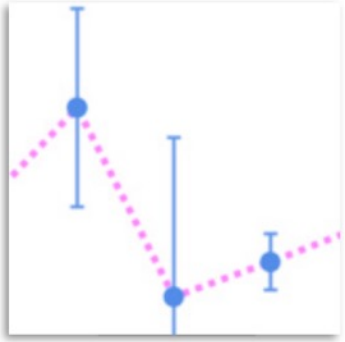


Pie Charts

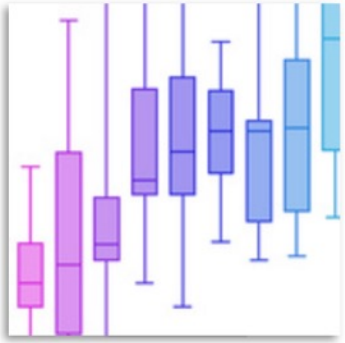
Bubble Charts

Statistical Charts

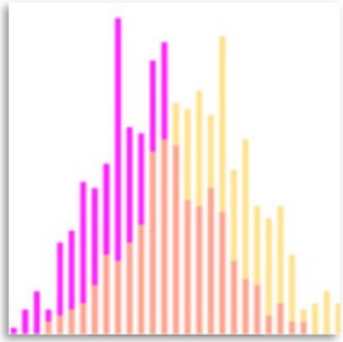
More Statistical Charts »



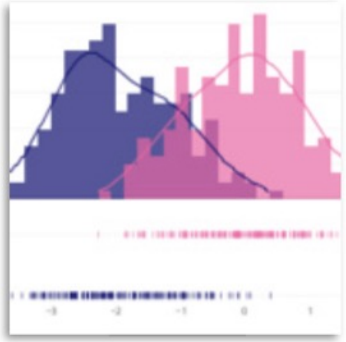
Error Bars



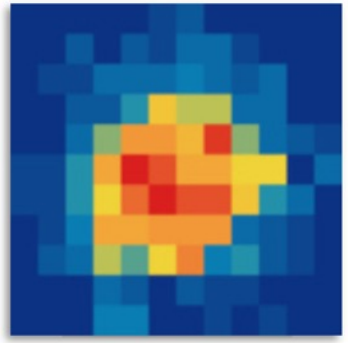
Box Plots



Histograms



Distplots



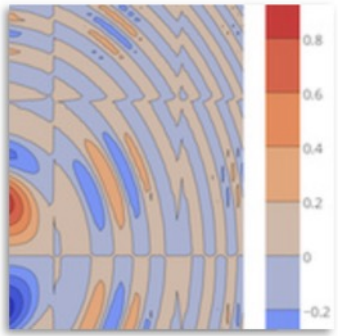
2D Histograms

https://plotly.com/python/

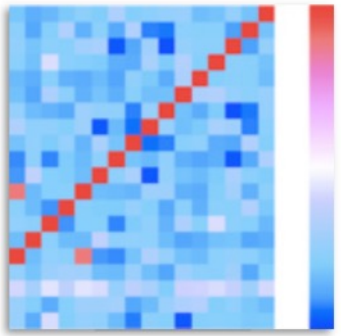
Python Plotly Graphing Library

Scientific Charts

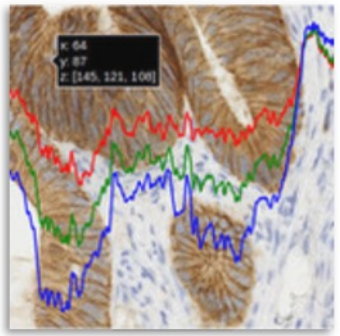
More Scientific Charts »



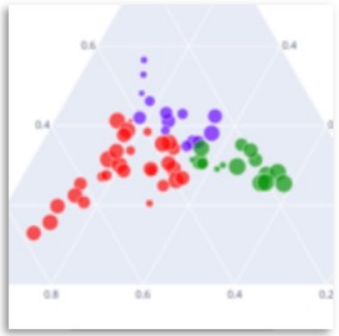
Contour Plots



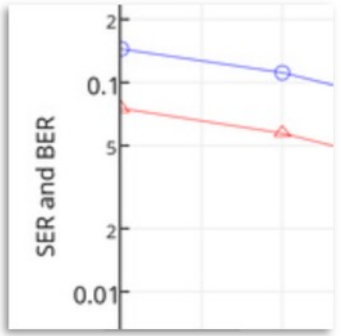
Heatmaps



Imshow



Ternary Plots



Log Plots

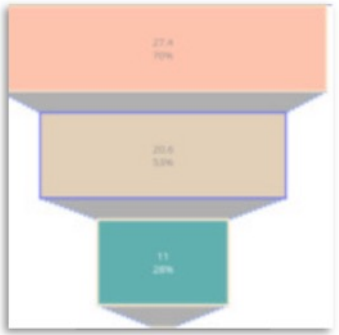
Financial Charts

More Financial Charts »

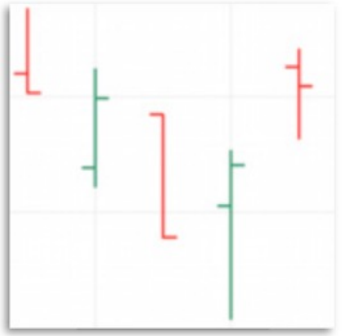
Time Series and Date Axes

Candlestick Charts

Waterfall Charts



Funnel Chart



OHLC Charts

https://plotly.com/python/

Python Plotly Graphing Library

Maps

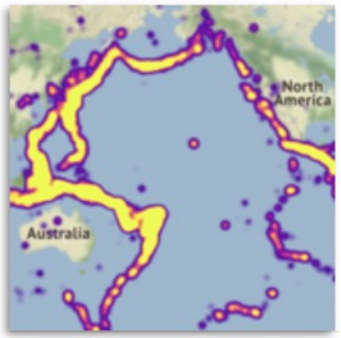
More Maps »

Mapbox Choropleth Maps

Lines on Mapbox

Filled Area on Maps

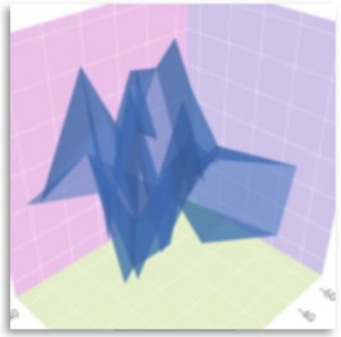
Bubble Maps



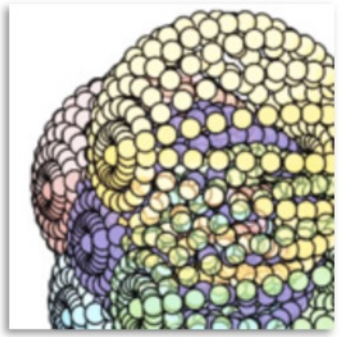
Mapbox Density Heatmap

3D Charts

More 3D Charts »

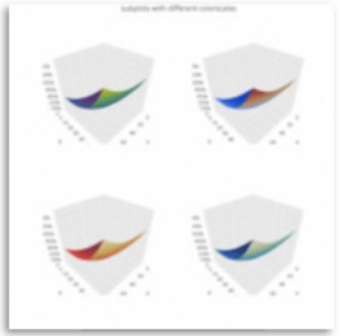


3D Axes

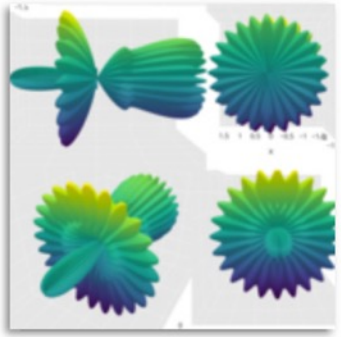


3D Scatter Plots

3D Surface Plots



3D Subplots



3D Camera Controls

https://plotly.com/python/

Python Plotly Graphing Library

Subplots

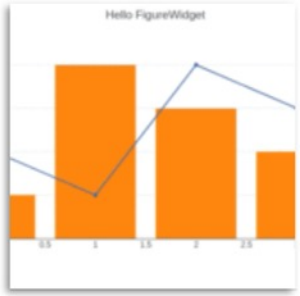
Mixed Subplots

Map Subplots

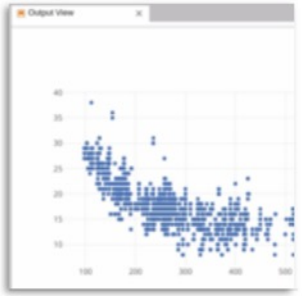
Table and Chart Subplots

Figure Factory Subplots

Jupyter Widgets Interaction

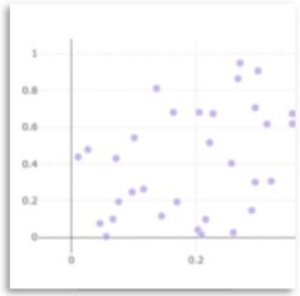


Plotly FigureWidget Overview



Jupyter Lab with FigureWidget

Interactive Data Analysis with FigureWidget ipywidgets



Click Events

https://plotly.com/python/

Python Bokeh

https://bokeh.org/

Iris flower data set

setosa

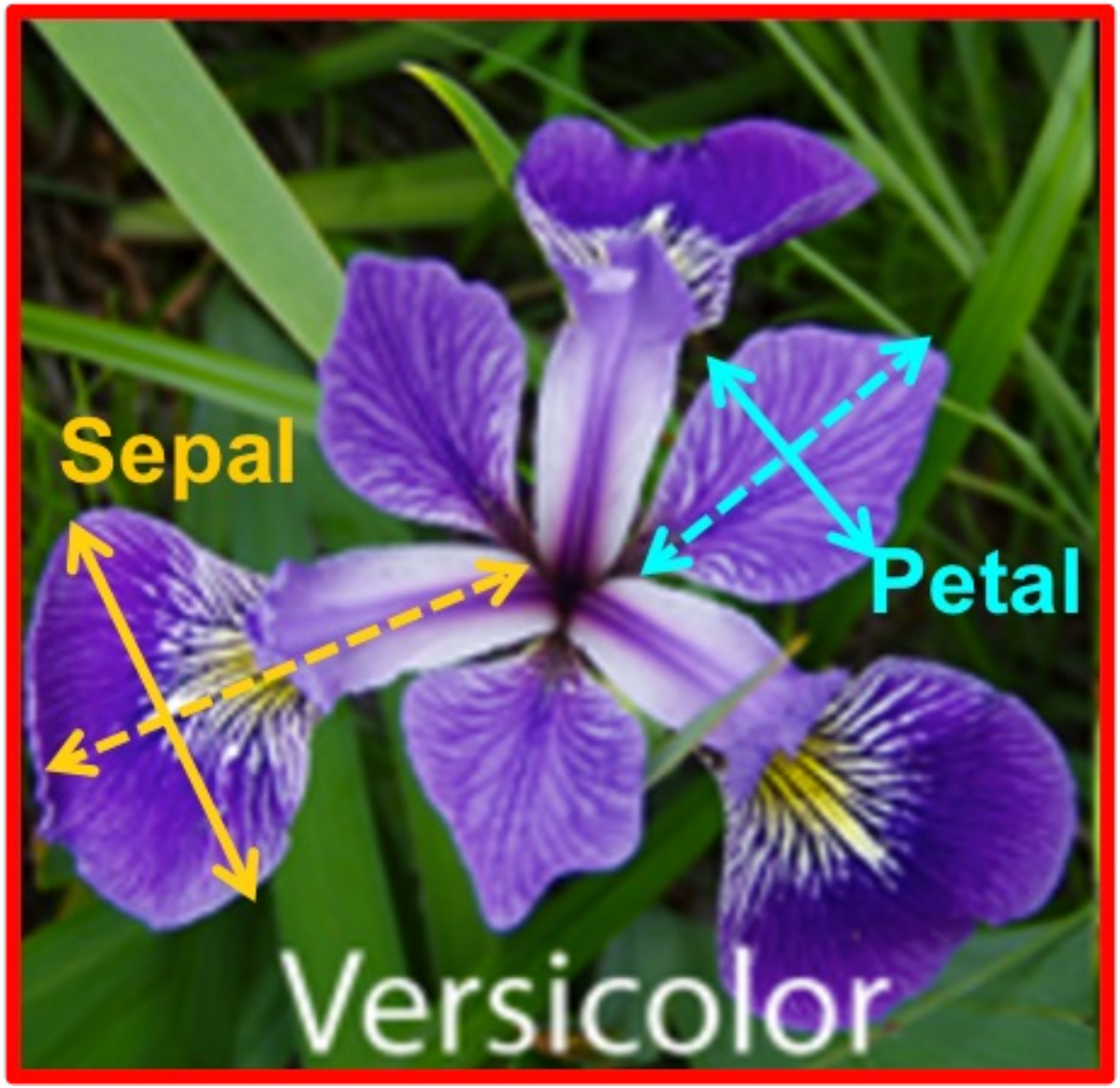
versicolor

virginica

Source: https://en.wikipedia.org/wiki/Iris_flower_data_set

Source: http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/

Iris Classfication



Source: http://suruchifialoke.com/2016-10-13-machine-learning-tutorial-iris-classification/

iris.data

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

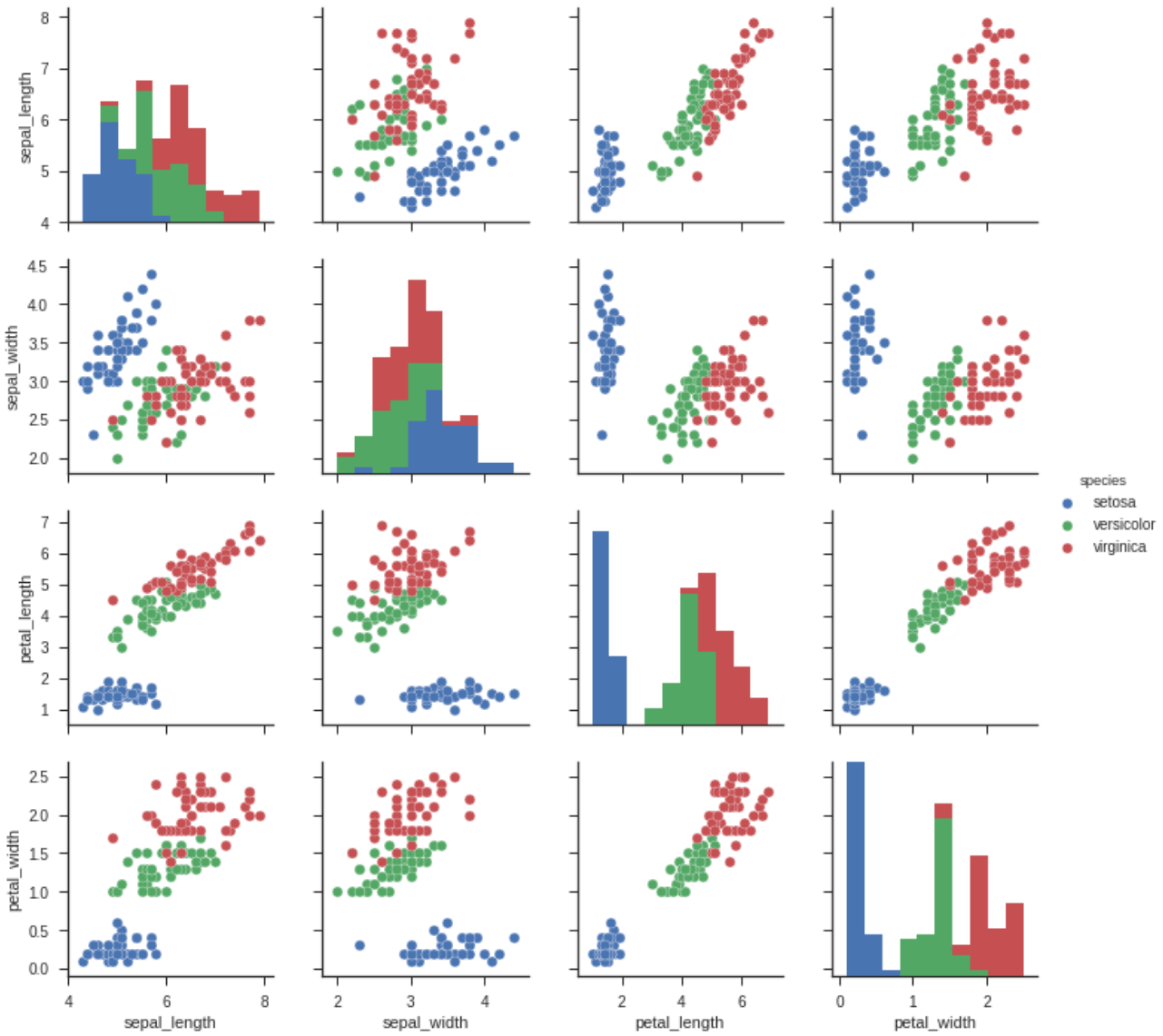
```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
5.4,3.9,1.3,0.4,Iris-setosa
5.1,3.5,1.4,0.3,Iris-setosa
5.7,3.8,1.7,0.3,Iris-setosa
5.1,3.8,1.5,0.3,Iris-setosa
5.4,3.4,1.7,0.2,Iris-setosa
5.1,3.7,1.5,0.4,Iris-setosa
4.6,3.6,1.0,0.2,Iris-setosa
5.1,3.3,1.7,0.5,Iris-setosa
4.8,3.4,1.9,0.2,Iris-setosa
5.0,3.0,1.6,0.2,Iris-setosa
```

setosa

virginica

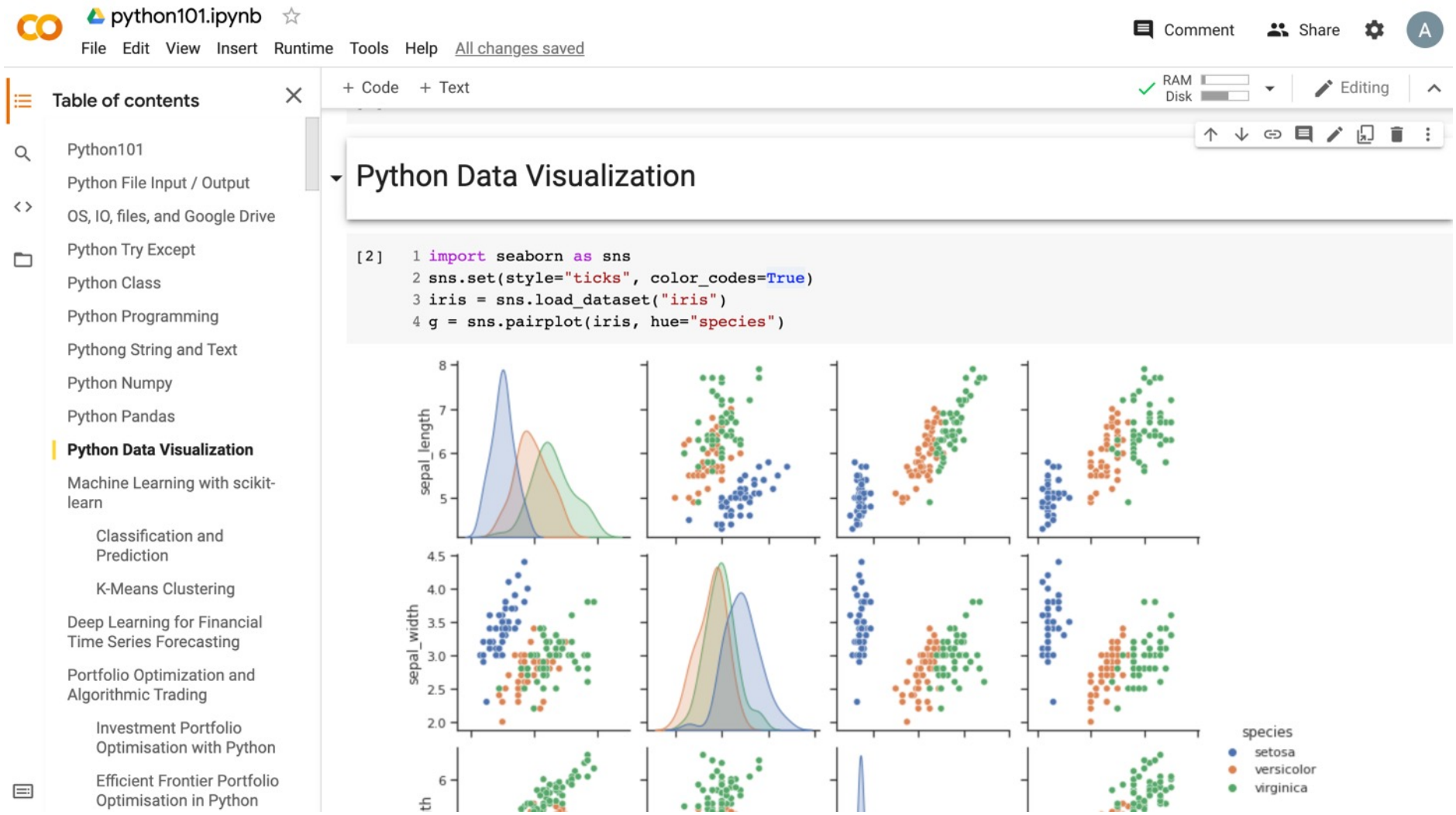
versicolor

Iris Data Visualization



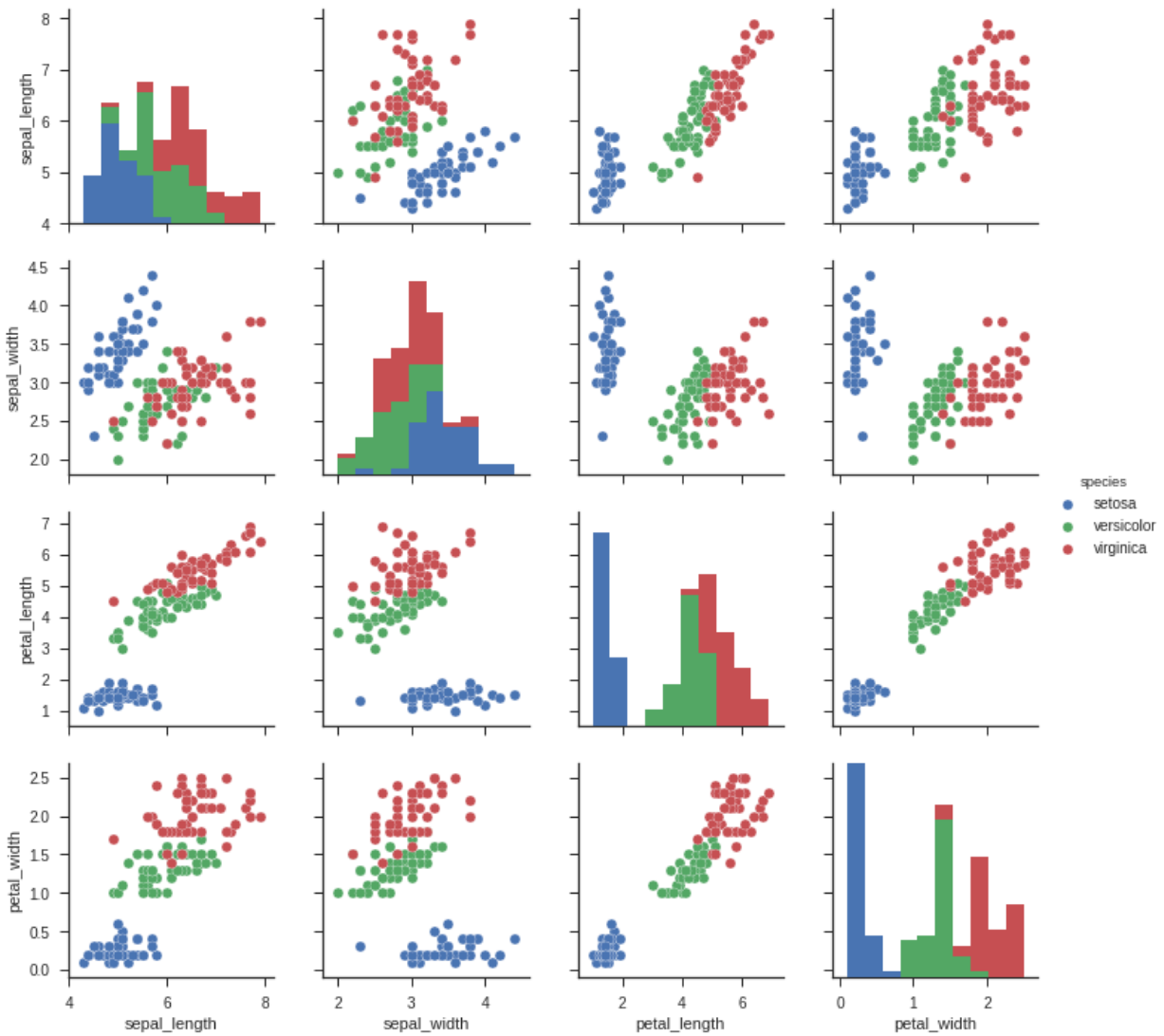
Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html

Data Visualization in Google Colab



https://tinyurl.com/aintpupython101

```
import seaborn as sns
sns.set(style="ticks", color_codes=True)
iris = sns.load_dataset("iris")
g = sns.pairplot(iris, hue="species")
```



Source: https://seaborn.pydata.org/generated/seaborn.pairplot.html

```
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
```

```
# Import Libraries
import numpy as np
import pandas as pd
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix
print('imported')
```

```
imported
```

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)
print(df.head(10))
```

```
# Load dataset
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']
df = pd.read_csv(url, names=names)
print(df.head(10))
```

	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

df.tail(10)

```
print(df.tail(10))
```

	sepal-length	sepal-width	petal-length	petal-width	class
140	6.7	3.1	5.6	2.4	Iris-virginica
141	6.9	3.1	5.1	2.3	Iris-virginica
142	5.8	2.7	5.1	1.9	Iris-virginica
143	6.8	3.2	5.9	2.3	Iris-virginica
144	6.7	3.3	5.7	2.5	Iris-virginica
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

df.describe()

```
print(df.describe())
```

```
       sepal-length  sepal-width  petal-length  petal-width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

```
print(df.info())
print(df.shape)
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
sepal-length    150 non-null float64
sepal-width     150 non-null float64
petal-length    150 non-null float64
petal-width     150 non-null float64
class           150 non-null object
dtypes: float64(4), object(1)
memory usage: 5.9+ KB
None
```

```
print(df.shape)
```

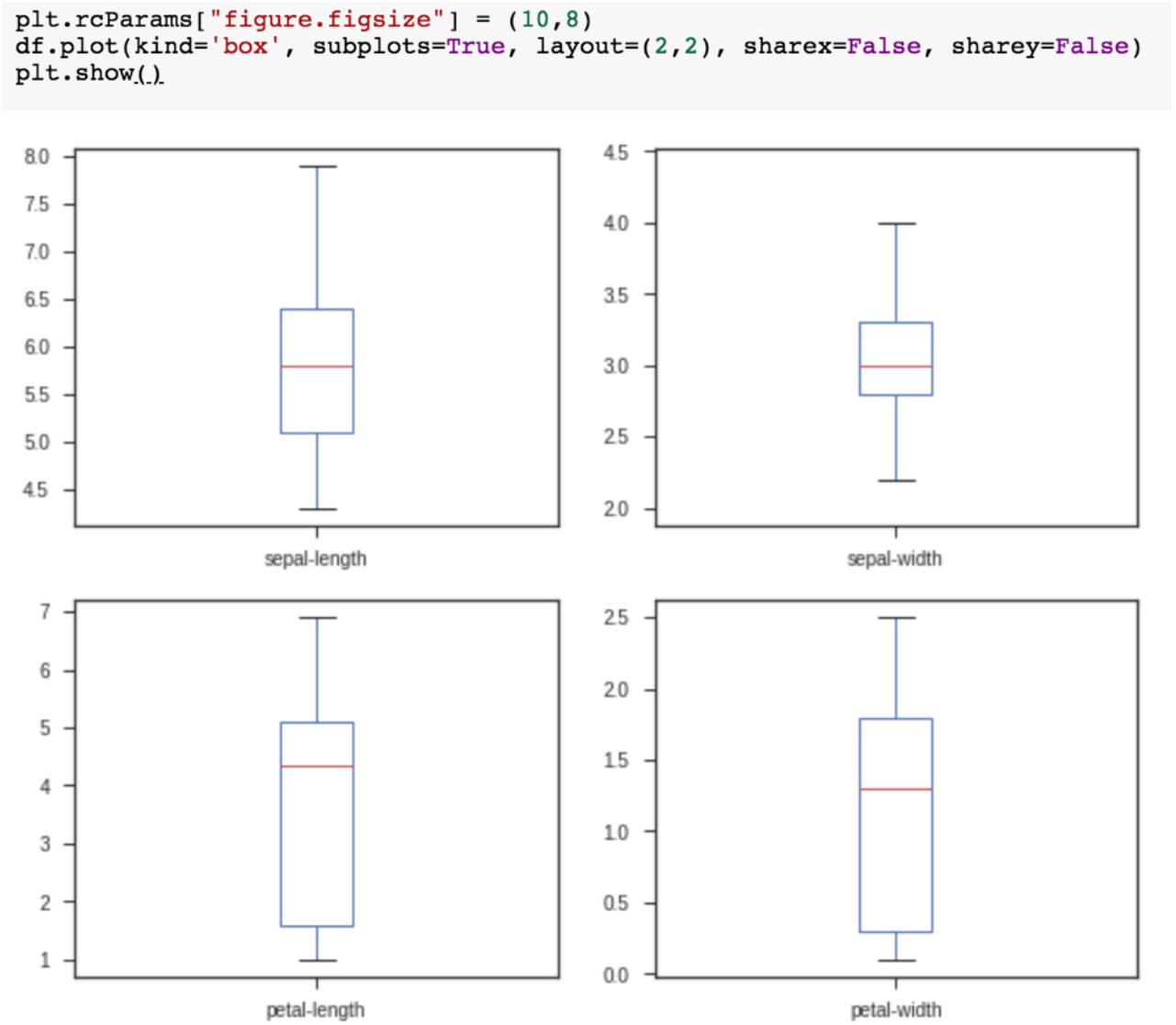
```
(150, 5)
```

```
df.groupby('class').size()
```

```
print(df.groupby('class').size())
```

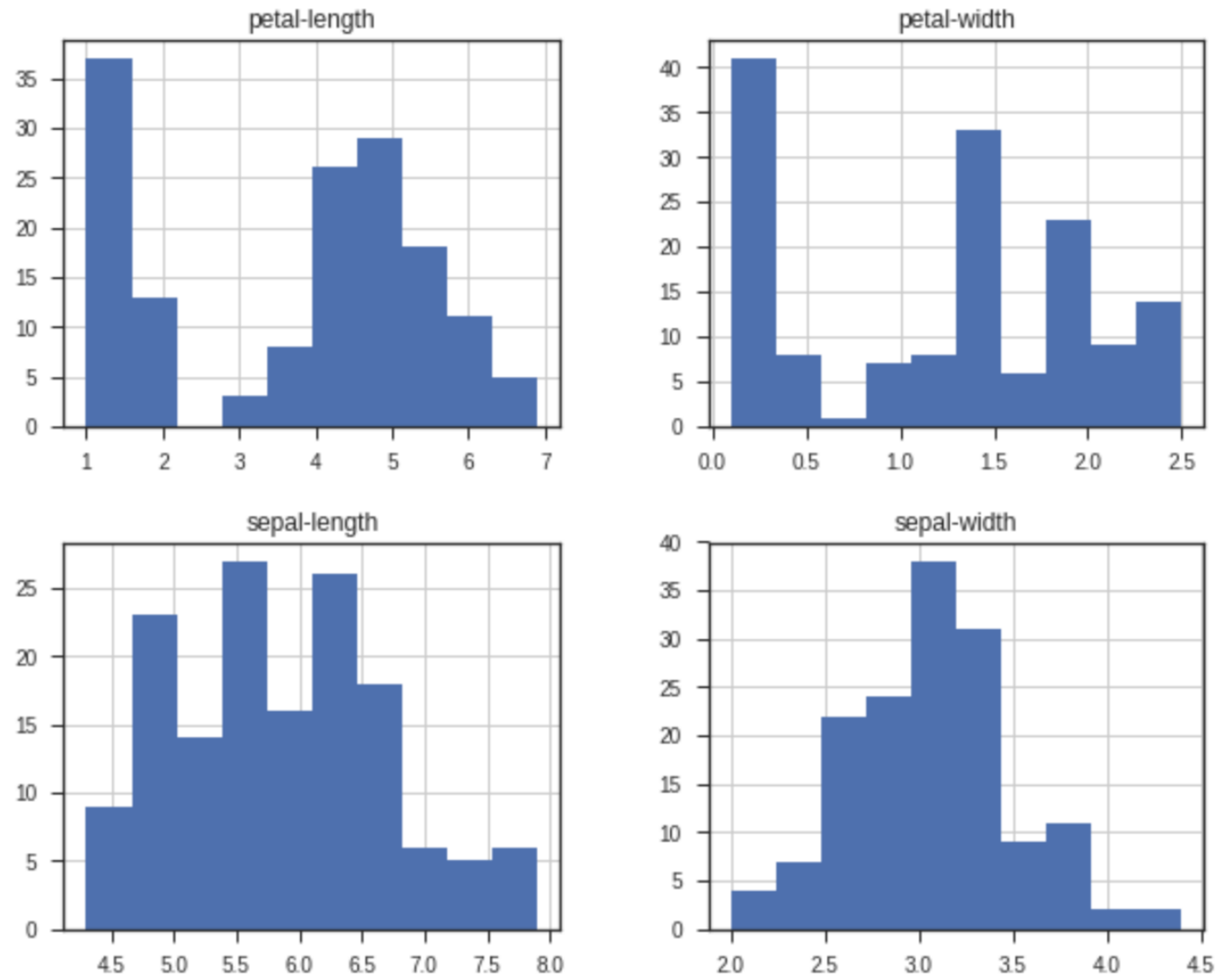
```
class
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
dtype: int64
```

```
plt.rcParams["figure.figsize"] = (10,8)
df.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
plt.show()
```



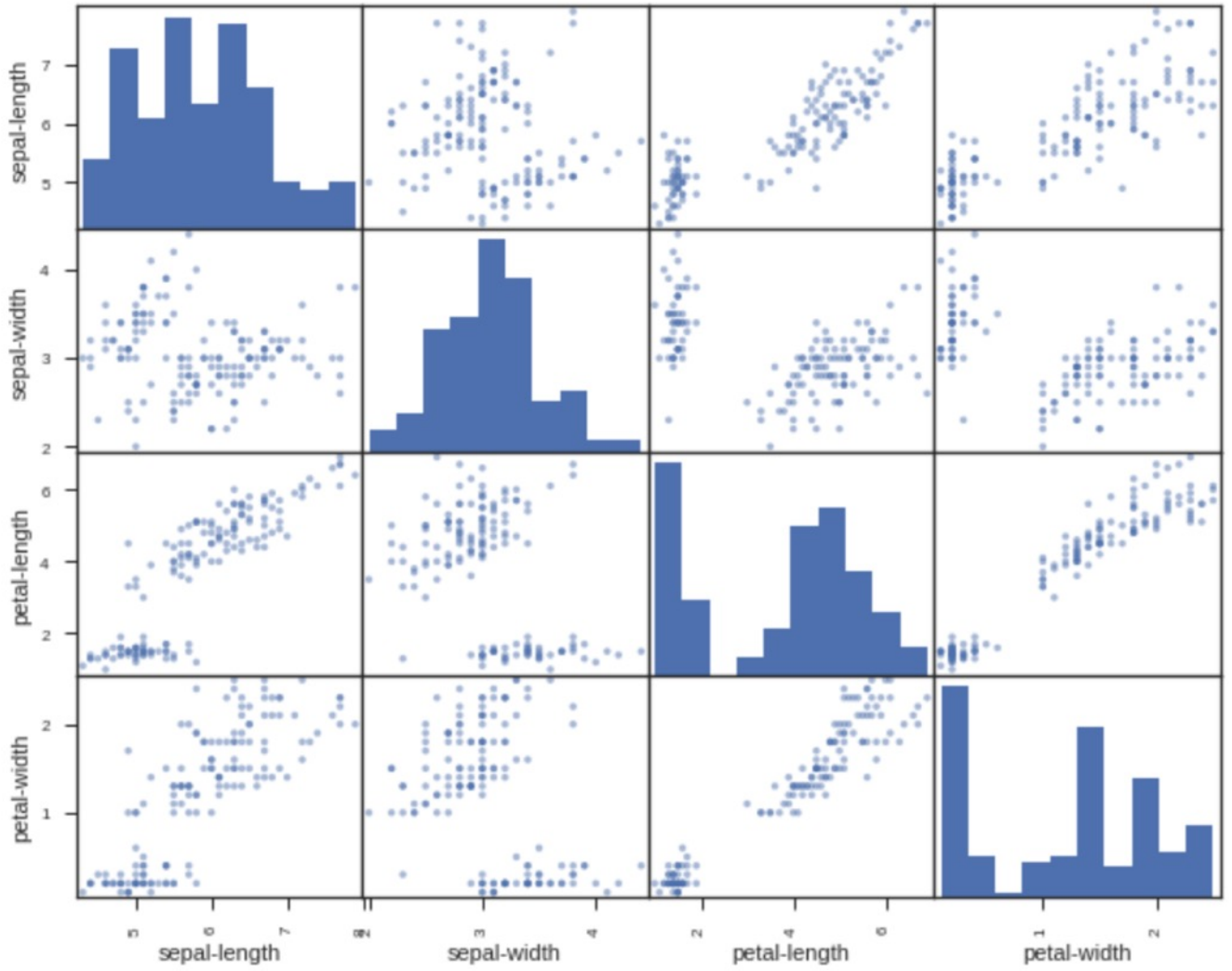
```
df.hist()
plt.show()
```

```
df.hist()
plt.show()
```



```
scatter_matrix(df)
plt.show()
```

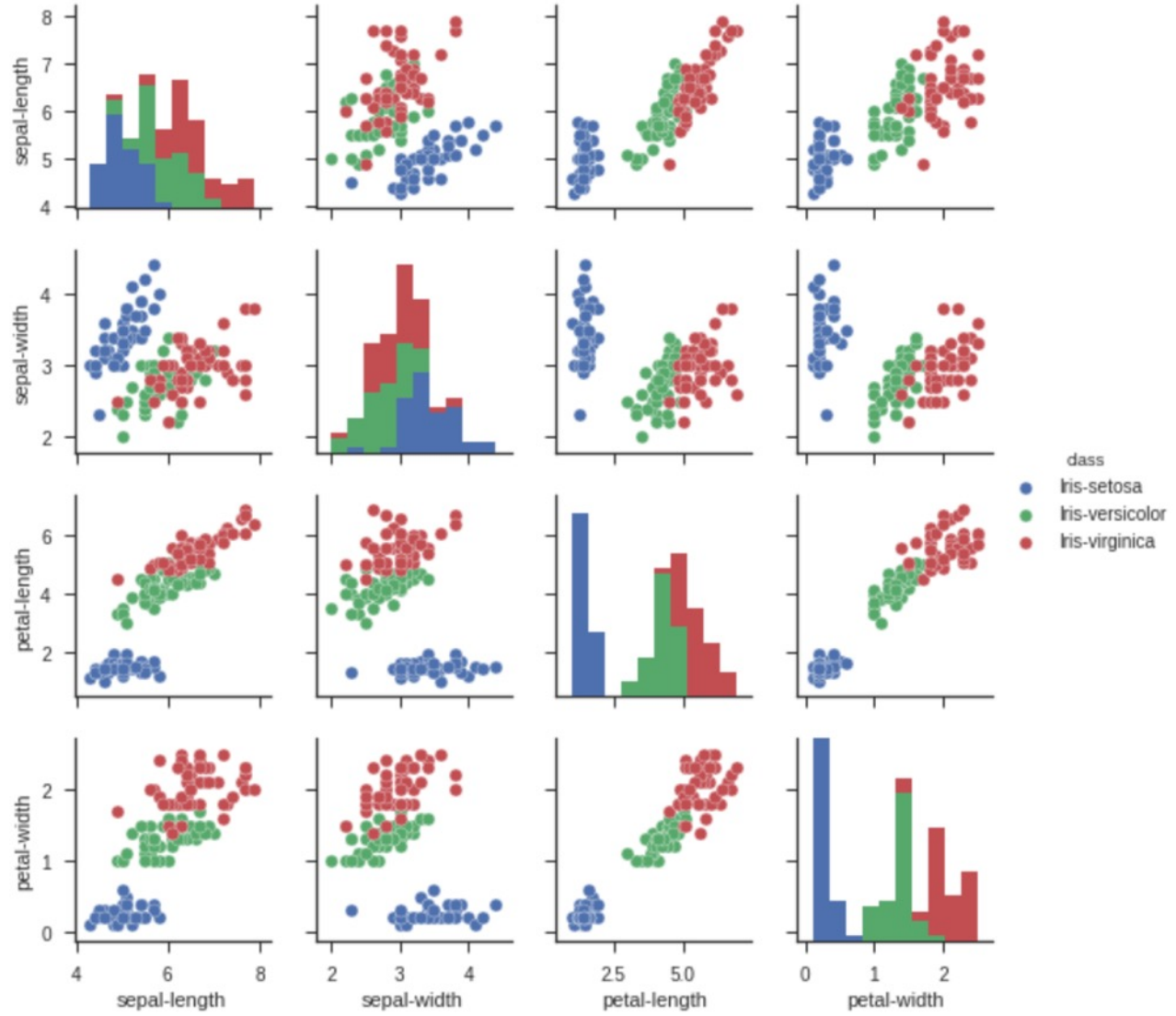

```
scatter_matrix(df)
plt.show()
```

sns.pairplot(df, hue="class", size=2)

```
sns.pairplot(df, hue="class", size=2)
```

```
<seaborn.axisgrid.PairGrid at 0x7f1d21267390>
```



Wes McKinney (2022), "Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter", 3rd Edition, O'Reilly Media.

wesm / pydata-book Public

Notifications | Fork 14.1k | Star 19k

Code | Issues 2 | Pull requests 1 | Actions | Projects | Wiki | Security | Insights

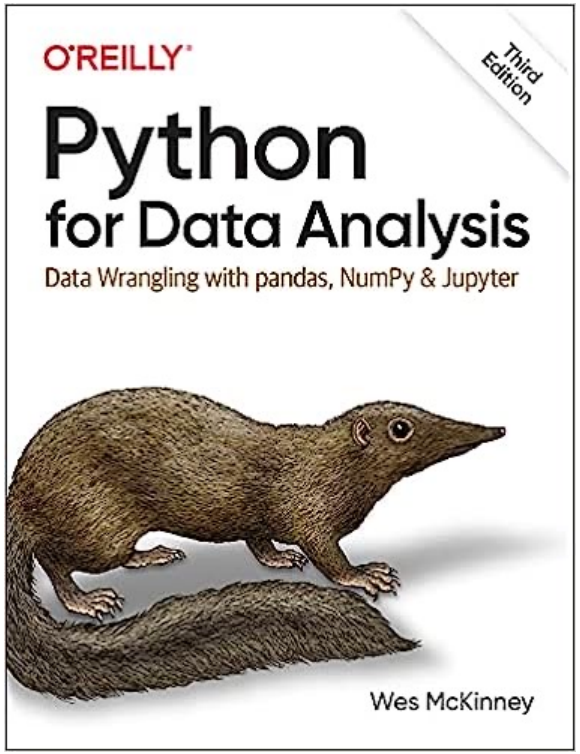
3rd-edition | 3 branches | 0 tags | Go to file | Code

wesm Upload cleaner notebook files without internal build toolchai... ... f1757b8 3 days ago 70 commits

datasets	Add fec.parquet	10 months ago
examples	Simplifying datasets	10 months ago
.gitignore	Add gitignore	8 years ago
COPYING	Update COPYING	4 months ago
README.md	Update notebooks in advance of publication	7 months ago
appa.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
appb.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch02.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch03.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch04.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch05.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago
ch06.ipynb	Upload cleaner notebook files without internal build toolchai...	3 days ago

About

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



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

Wes McKinney (2022), "Python for Data Analysis: Data Wrangling with pandas, NumPy, and Jupyter", 3rd Edition, O'Reilly Media.

3rd-edition · pydata-book / ch04.ipynb · Go to file · ···

wesm Upload cleaner notebook files without internal build toolchain instru... ··· Latest commit f1757b8 3 days ago History

3 contributors

1224 lines (1224 sloc) | 21.9 KB · Raw · Blame

In [1]:
```
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 [2]:
```
import numpy as np

my_arr = np.arange(1_000_000)
my_list = list(range(1_000_000))
```

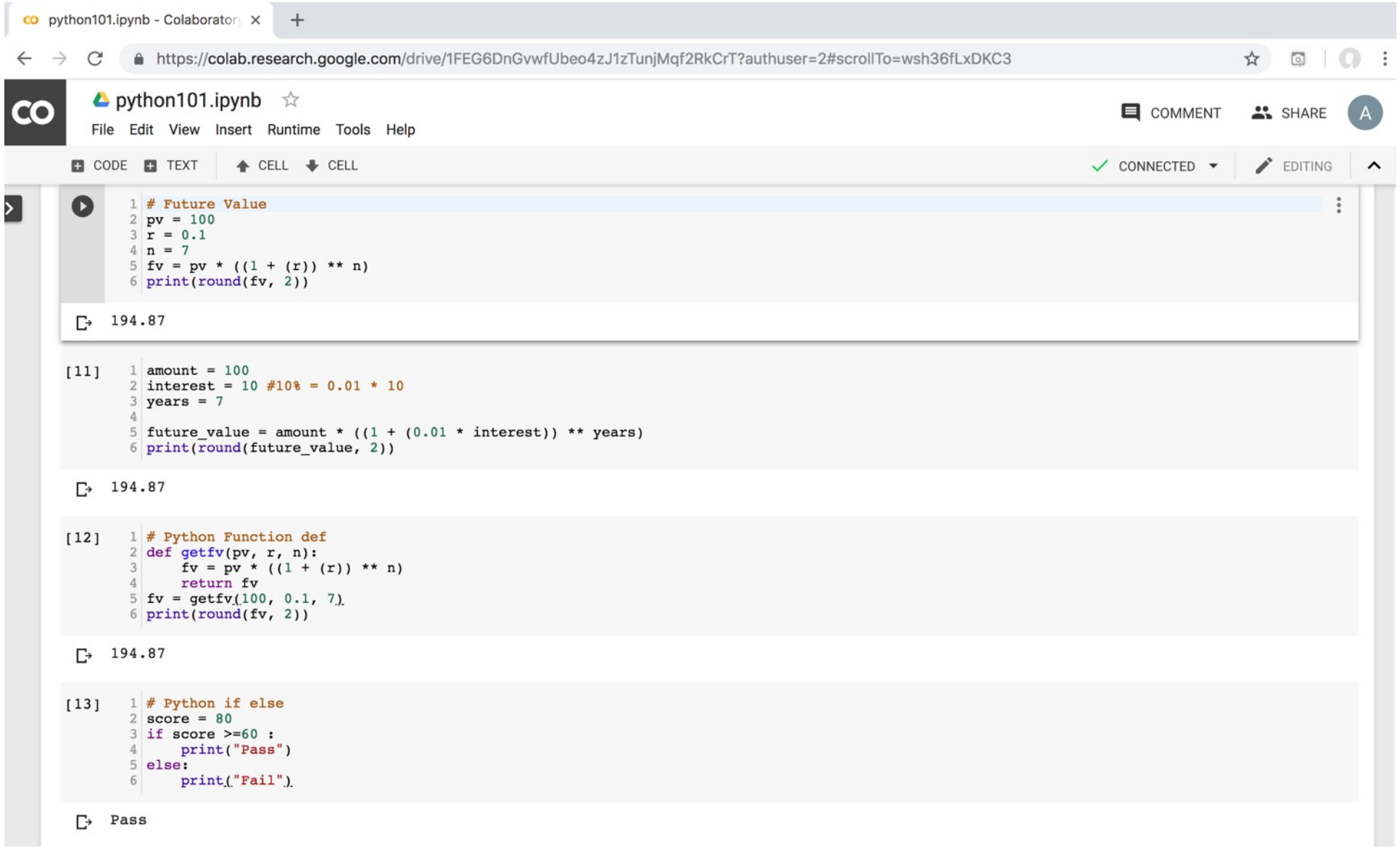
In [3]:
```
%timeit my_arr2 = my_arr * 2
%timeit my_list2 = [x * 2 for x in my_list]
```

In [4]:
```
import numpy as np
data = np.array([[1.5, -0.1, 3], [0, -3, 6.5]])
data
```

Source: https://github.com/wesm/pydata-book/blob/3rd-edition/ch04.ipynb

Python in Google Colab (Python101)

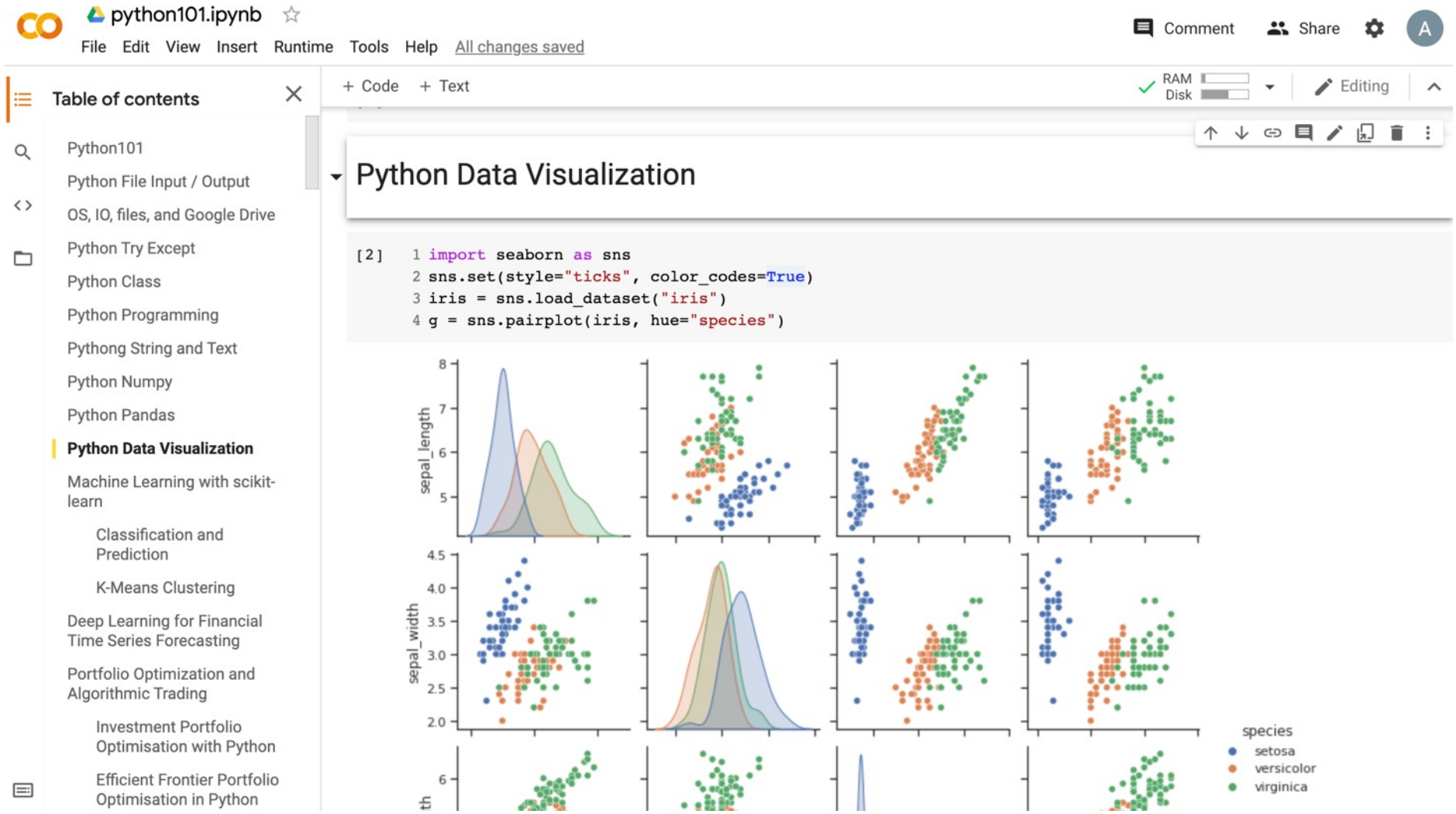
https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



https://tinyurl.com/aintpupython101

Python in Google Colab (Python101)

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



https://tinyurl.com/aintpupython101

Papers with Code
State-of-the-Art (SOTA)

Computer Vision

Semantic Segmentation

185 benchmarks

3397 papers with code

Image Classification

390 benchmarks

2778 papers with code

Object Detection

269 benchmarks

2559 papers with code

Contrastive Learning

2 benchmarks

1119 papers with code

Image Generation

208 benchmarks

1097 papers with code

▸ See all 1415 tasks

Natural Language Processing

Language Modelling

458 benchmarks

2248 papers with code

Question Answering

181 benchmarks

1818 papers with code

Machine Translation

78 benchmarks

1721 papers with code

Sentiment Analysis

87 benchmarks

1040 papers with code

Text Generation

242 benchmarks

931 papers with code

▸ See all 664 tasks

https://paperswithcode.com/sota

Summary

- Data Science and Sustainability
- Data Collection and Analysis for Sustainability
- Implementing Data-Driven Sustainability Strategies
- ESG Data Science Foundation with Python

Source: Dipanjan Sarkar (2019), Text Analytics with Python: A Practitioner's Guide to Natural Language Processing, Second Edition. APress.

References

- Cino Robin Castelli, Cyril Shmatov (2022), Quantitative Methods for ESG Finance, Wiley
- Simon Thompson (2023), Green and Sustainable Finance: Principles and Practice in Banking, Investment and Insurance, 2nd Edition, Kogan Page.
- Chrissa Pagitsas (2023), Chief Sustainability Officers At Work: How CSOs Build Successful Sustainability and ESG Strategies, Apress.
- 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.
- 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.
- 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
- Min-Yuh Day (2024), Python 101, https://tinyurl.com/aintpupython101