

Tutorial III

Python Programming with Jupyter Notebooks

Google Colab
&

Python for Data Science



Google Colab

- Google is quite aggressive in AI research. Over many years, Google developed AI framework called **TensorFlow** and a development tool called **Colaboratory**.
- **TensorFlow** is an *open-sourced* end-to-end platform that makes it easy for you to build and deploy Machine Learning models.
- It is an *entire ecosystem* to help you solve challenging, real-world problems with machine learning.

Google Colab

- Colaboratory is now known as **Google Colab** or simply **Colab**.
- The introduction of Colab has eased the learning and development of machine learning applications.
- Another feature that Google offers to the developers is the use of GPU (graphical processing unit). Colab supports GPU and is free.
- The reasons for making it free for public was to make its software *a standard in the academics for teaching machine learning and data science*.

Google Colab

- A **Jupyter Notebook** (formerly known as IPython Notebook) is an interactive web application for creating and sharing computational documents..
- The project was first named IPython and later renamed Jupyter in 2014. It is a fully open-source product, and users can use every functionality available for free. .
- A Jupyter Notebook is a mutable file saved in **ipynb** format.

Google Colab

- Colab is a free **Jupyter Notebook** *environment* that runs entirely in the cloud.
- Most importantly, *it does not require a setup* and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs.
- Colab *supports many popular machine learning libraries* which can be easily loaded in your notebook.



<https://jupyter.org/>

What Colab Offers You?

- As a programmer, you can perform the following using Google Colab
 - **Write and execute code in Python**
 - **Document your code** that supports mathematical equations
 - Create/Upload/**Share** notebooks
 - *Import/Save notebooks from/to Google Drive*
 - **Import/Publish notebooks from GitHub**
 - **Import external datasets** e.g. from Kaggle
 - *Integrate Machine Learning Packages* like PyTorch, TensorFlow, Keras, OpenCV ...
 - **Free Cloud** service with free GPU

How Google Colab Works?

As Colab *implicitly uses Google Drive* for storing your notebooks, ensure that you are logged in to your Google Drive account before proceeding further.

Step 0: Open the following URL in your browser:
<https://colab.research.google.com>

Ein Colab-Abo optimal nutzen
PRO+ Datei Bearbeiten Anzeige Einfügen Laufzeit Tools Hilfe

+ Code + Text In Google

Notebook öffnen

Beispiele >

Zuletzt geöffnet >

Google Drive >

GitHub >

Hochladen >

Suche: Notebooks durchsuchen

Titel	Zuletzt geöffnet	Zuerst geöffnet	
Ein Colab-Abo optimal nutzen	12:26	12. Jan. 2022	
KETE_Tryouts-DBE.ipynb	4. Dezember	4. Dezember	
Kopie von CS231n-PyTorch-Tutorial.ipynb	29. November	29. November	
CS231n-PyTorch-Tutorial.ipynb	29. November	29. November	
BERT_test.ipynb	22. November	23. Oktober	

[+ Neues Notebook](#) [Abbrechen](#)

```
[ ] gpu_info = !nvidia-smi
gpu_info = '\n'.join(
    if gpu_info.find('fail') > 0:
        print('Not connected to GPU')
    else:
        print(gpu_info)
```


Step 1: Click on the **NEW NOTEBOOK** link
at the bottom left of the opened window

The screenshot shows the Google Colab interface with a 'Notebook öffnen' (Open Notebook) dialog box open. The dialog box has a sidebar on the left with options: 'Beispiele', 'Zuletzt geöffnet' (highlighted), 'Google Drive', 'GitHub', and 'Hochladen'. The main area of the dialog shows a search bar 'Notebooks durchsuchen' and a table of notebooks.

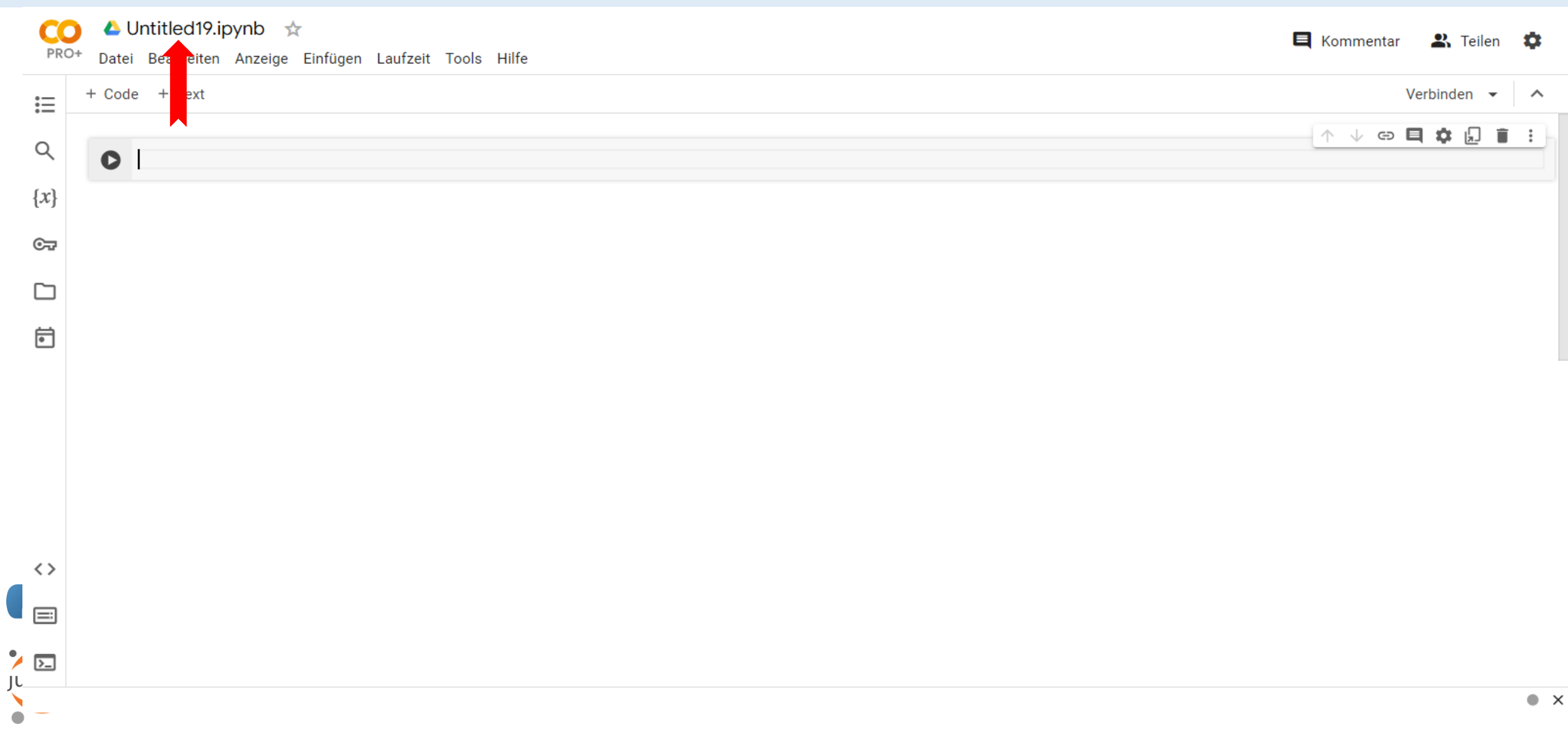
Titel	Zuletzt geöffnet	Zuerst geöffnet	
Ein Colab-Abo optimal nutzen	12:26	12. Jan. 2022	
KETE_Tryouts-DBE.ipynb	4. Dezember	4. Dezember	
Kopie von CS231n-PyTorch-Tutorial.ipynb	29. November	29. November	
CS231n-PyTorch-Tutorial.ipynb	29. November	29. November	
BERT_test.ipynb	22. November	23. Oktober	

At the bottom left of the dialog box, there is a blue button labeled '+ Neues Notebook' with a red arrow pointing to it. At the bottom right, there is a button labeled 'Abbrechen'.

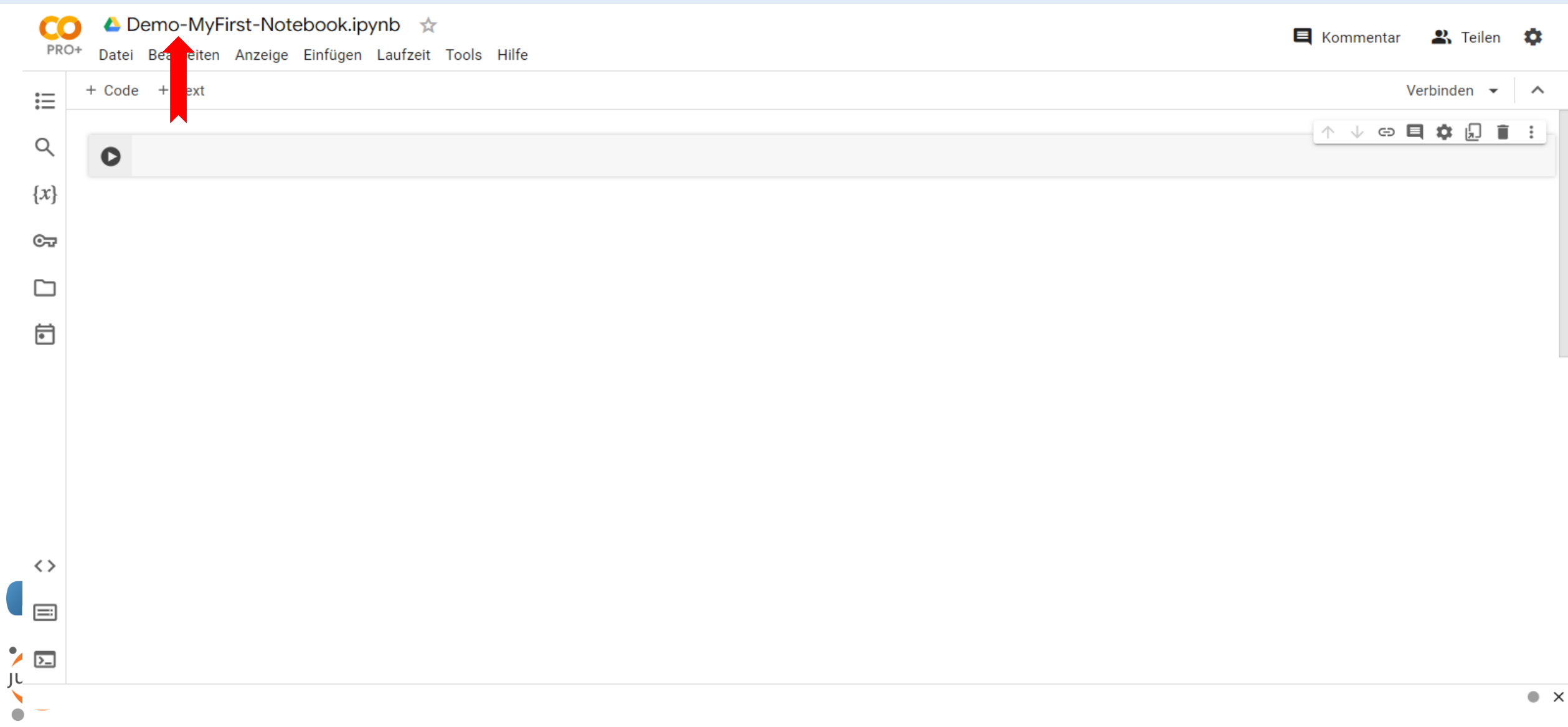
Step 3: Click on the **connect** at the top (right) of the screen

The screenshot displays the CO+ PRO+ web interface. At the top left, the logo 'CO+ PRO+' is visible next to the document title 'Untitled19.ipynb' and a star icon. Below this is a menu bar with options: 'Datei', 'Bearbeiten', 'Anzeige', 'Einfügen', 'Laufzeit', 'Tools', and 'Hilfe'. On the top right, there are icons for 'Kommentar' (comment), 'Teilen' (share), and a settings gear. Below these icons is a 'Verbinden' (connect) button with a dropdown arrow, which is highlighted by a large red arrow pointing to it. The main workspace is a large white area with a toolbar at the top containing icons for undo, redo, link, comment, settings, copy, and delete. The left sidebar contains icons for a menu, search, and various file/folder management tools. The bottom left corner shows a vertical toolbar with icons for code, text, and other editing functions.

Step 4a: Click on the **untitledxx.ipynb** at the top (left) of the screen



Step 4b: Replace **untitledxx** with an appropriate notebook name



The screenshot displays the JupyterLab interface. At the top, the title bar shows the Jupyter logo, the text "Demo-MyFirst-Notebook.ipynb", and a star icon. Below the title bar is a menu bar with options: "Datei", "Bearbeiten", "Anzeige", "Einfügen", "Laufzeit", "Tools", and "Hilfe". On the right side of the title bar, there are icons for "Kommentar", "Teilen", and a settings gear. Below the menu bar, there is a toolbar with "+ Code" and "+ Text" buttons. A red arrow points to the text "untitledxx" in the toolbar. The main area of the interface is a large, empty white space. On the left side, there is a sidebar with icons for "Code", "Text", "File", "Run", and "Help". On the right side, there is a sidebar with icons for "Verbinden", "Up", "Down", "Link", "Comment", "Settings", "Copy", "Trash", and "More".

Step 5: Write and execute a single program statement

The screenshot shows a Jupyter Notebook interface with the title "Demo-MyFirst-Notebook.ipynb". The top bar includes a "PRO+" logo, a menu with "Datei", "Bearbeiten", "Anzeige", "Einfügen", "Laufzeit", "Tools", and "Hilfe", and a status message "Alle Änderungen wurden gespeichert". On the right, there are buttons for "Kommentar", "Teilen", and a settings gear. Below the top bar, there are tabs for "+ Code" and "+ Text". The main area contains a code cell with the text `print ("Hello World")`. To the left of the code cell, there is a green checkmark and "0 s". Below the code cell, the output "Hello World" is displayed. A red arrow points to the output. On the right side of the code cell, there is a toolbar with icons for up, down, link, comment, settings, print, and delete. The bottom status bar shows a green checkmark, "0 s", and "Abgeschlossen um 12:31".

0 s

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

Hello World

0 s Abgeschlossen um 12:31

Step 6: Click on box at left side to import (upload) files

The screenshot displays the Google Colab environment. The top bar shows the Colab logo, the notebook title "Demo-MyFirst-Notebook.ipynb", and a star icon. Below this is a menu bar with options: Datei, Bearbeiten, Anzeige, Einfügen, Laufzeit, Tools, Hilfe, and a status message "Alle Änderungen wurden gespeichert". On the right side of the top bar are icons for "Kommentar", "Teilen", and a settings gear.

The left sidebar contains a "Dateien" (Files) section with a search icon, a file upload icon, a folder icon, and a calendar icon. Below these are icons for a code editor, a terminal, and a console. The file explorer shows a directory structure with a folder named "sample_data" highlighted. A red arrow points to the "upload" icon (a folder with an upward arrow) in the file explorer. Another red arrow points to the "sample_data" folder.

The main area of the interface shows a code cell with the following content:

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

Below the code cell, the output "Hello World" is displayed. The status bar at the bottom indicates "0 s" and "Abgeschlossen um 12:31".

Step 7a: Select (local) file to upload

The screenshot shows the JupyterLab interface with a file explorer window open. The file explorer is titled "Öffnen" and displays a list of files in the directory "DATA (Z:) > zWORK > zGithub > BINA-FS22-WORK > LB07-ProcessMining > R". The file "DATA_patients.csv" is selected, and a red arrow points to it. The file explorer also shows the file's metadata: Name, Änderungsdatum, Typ, and Größe. The file "DATA_patients.csv" is a Microsoft Excel file, 399 KB in size, and was last modified on 14.07.2022 at 15:36. The file explorer is open over a JupyterLab window showing a code cell with the text "print("Hello World")". The JupyterLab interface includes a top bar with the "Demo-MyFirst-Notebook.ipynb" title, a left sidebar with file explorer and file browser views, and a bottom status bar showing "0 s" and "Abgeschlossen um 12:31".

Step 7b: Show/Inspect uploaded file

CO

Demo-MyFirst-Notebook.ipynb

☆

PRO+ Datei Bearbeiten Anzeige Einfügen Laufzeit Tools Hilfe Alle Änderungen wurden gespeichert

Kommentar

Teilen

⚙

RAM

Laufwerk

⌵

⌴

Dateien

🔍

📁

📄

👁

{x}

+

..

▶

📁 .config

🔑

▶

📁 sample_data

▶

📁 .ipynb_checkpoints

📄 DATA_patients.csv

📄 README.md

📄 anscombe.json

📄 california_housing_test.csv

📄 california_housing_train.csv

📄 mnist_test.csv

📄 mnist_train_small.csv

+ Code

+ Text

Notebook

DATA_patients.csv

×

1 to 10 of 5442 entries

Filter

📄

	handling	patient	employee	handling_id	registration_type	time	.order
1	Registration	1	r1	1	start	2017-01-02 11:41:53	1
2	Registration	2	r1	2	start	2017-01-02 11:41:53	2
3	Registration	3	r1	3	start	2017-01-04 01:34:05	3
4	Registration	4	r1	4	start	2017-01-04 01:34:04	4
5	Registration	5	r1	5	start	2017-01-04 16:07:47	5
6	Registration	6	r1	6	start	2017-01-04 16:07:47	6
7	Registration	7	r1	7	start	2017-01-05 04:56:11	7
8	Registration	8	r1	8	start	2017-01-05 04:56:11	8
9	Registration	9	r1	9	start	2017-01-06 05:58:54	9
10	Registration	10	r1	10	start	2017-01-06 05:58:54	10

Show

10

▼

per page

1

2

10

100

500

540

545

Laufwerk

198.91 GB verfügbar

0 s

Abgeschlossen um 12:31

Step 8a: Write program to load/read data from a csv file

The screenshot displays a Jupyter Notebook interface with the title "Demo-MyFirst-Notebook.ipynb". The top navigation bar includes a "PRO+" logo, a menu with "Datei", "Bearbeiten", "Anzeige", "Einfügen", "Laufzeit", "Tools", and "Hilfe", and a status message "Alle Änderungen wurden gespeichert". On the right, there are links for "Kommentar", "Teilen", and a settings icon, along with a user profile picture. Below the navigation bar, the left sidebar shows a file explorer with a tree view containing folders like ".config" and "sample_data", and files such as "DATA_patients.csv", "README.md", "anscombe.json", "california_housing_test.csv", "california_housing_train.csv", "mnist_test.csv", and "mnist_train_small.csv". The main area of the notebook is titled "Notebook DATA_patients.csv" and contains three code cells. The first cell executes `print("Hello World")` and outputs "Hello World". The second cell imports the pandas library with `import pandas as pd`. The third cell reads the CSV file with `df = pd.read_csv("sample_data/DATA_patients.csv")` and then displays the first five rows of the dataframe using `df.head()`. The status bar at the bottom indicates "Laufwerk" with a progress bar and "198.91 GB verfügbar". The bottom right corner shows a green checkmark, "0 s", and "Abgeschlossen um 15:12".

PRO+ Datei Bearbeiten Anzeige Einfügen Laufzeit Tools Hilfe Alle Änderungen wurden gespeichert Kommentar Teilen

Dateien

- ..
- .config
- sample_data
 - .ipynb_checkpoints
 - DATA_patients.csv
 - README.md
 - anscombe.json
 - california_housing_test.csv
 - california_housing_train.csv
 - mnist_test.csv
 - mnist_train_small.csv

Notebook DATA_patients.csv

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

Hello World

```
[2] # Import Python Libraries
import pandas as pd
```

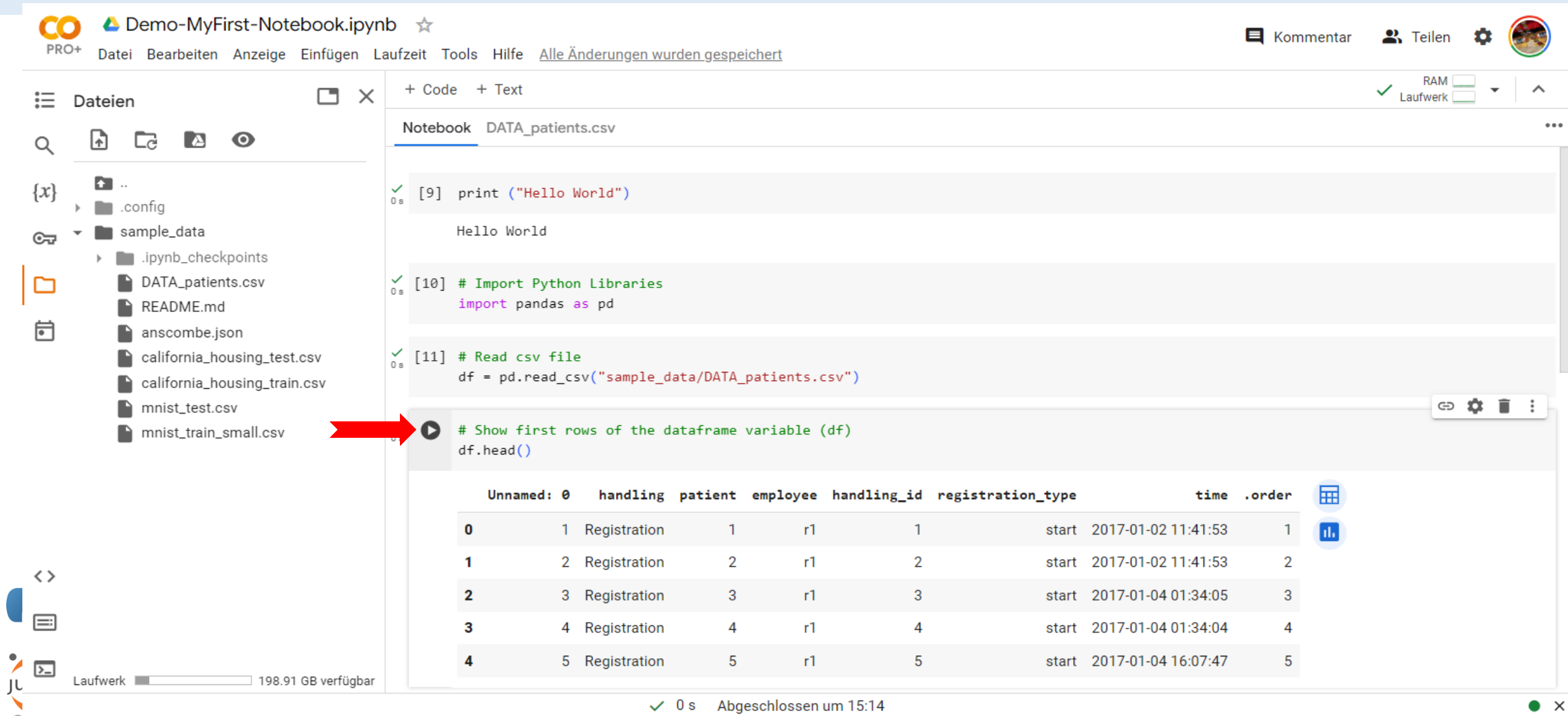
```
[7] # Read csv file
df = pd.read_csv("sample_data/DATA_patients.csv")
```

```
# Show first rows of the dataframe variable (df)
df.head()
```

Laufwerk 198.91 GB verfügbar

0 s Abgeschlossen um 15:12

Step 8b: Execute program to load/read data from a csv file



The screenshot shows a Jupyter Notebook interface with the following components:

- Top Bar:** Includes the Colab logo, the notebook name "Demo-MyFirst-Notebook.ipynb", and a star icon. On the right, there are links for "Kommentar", "Teilen", and a settings icon.
- Left Sidebar (Dateien):** Displays a file explorer with a tree view. The "sample_data" folder is expanded, showing files like "DATA_patients.csv", "README.md", "anscombe.json", "california_housing_test.csv", "california_housing_train.csv", "mnist_test.csv", and "mnist_train_small.csv". A red arrow points to the "DATA_patients.csv" file.
- Main Notebook Area:** Contains three code cells:
 - Cell [9]: `print ("Hello World")`. The output is "Hello World".
 - Cell [10]: `# Import Python Libraries`
`import pandas as pd`
 - Cell [11]: `# Read csv file`
`df = pd.read_csv("sample_data/DATA_patients.csv")`
Below the code, there is a button to execute the cell (a play icon) and a button to show the first rows of the dataframe variable (df).
- Output of Cell [11]:** A table showing the first 5 rows of the dataframe. The columns are: Unnamed: 0, handling, patient, employee, handling_id, registration_type, time, and .order.

	Unnamed: 0	handling	patient	employee	handling_id	registration_type	time	.order
0	1	Registration	1	r1	1	start	2017-01-02 11:41:53	1
1	2	Registration	2	r1	2	start	2017-01-02 11:41:53	2
2	3	Registration	3	r1	3	start	2017-01-04 01:34:05	3
3	4	Registration	4	r1	4	start	2017-01-04 01:34:04	4
4	5	Registration	5	r1	5	start	2017-01-04 16:07:47	5

At the bottom of the notebook, it shows "0 s" and "Abgeschlossen um 15:14".

What is Python?

- **Python** is an *interpreted*, high level and general-purpose programming language.
- **Python** is a *general-purpose* coding language—which means that, unlike HTML, CSS, and JavaScript, it can be used for other types of programming and software development besides web development.

Why Python?

- Python works on **different platforms** (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a **simple syntax** similar to the English language with influence from mathematics.
- Python has **syntax** that allows developers to *write programs with fewer lines* than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. *Thus prototyping can be very quick.*
- Python can be treated in a procedural way, an object-oriented way or a functional way.

What can Python do?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

Multi Words Variable Names

- Variable names with more than one word can be difficult to read. There are several techniques you can use to make them more readable:
- **Pascal Case**
 - Each word starts with a capital letter:
e.g. `MyVariableName` = "John"
- **Camel Case**
 - Each word, except the first, starts with a capital letter:
e.g. `myVariableName` = "John"
- **Snake Case**
 - Each word is separated by an underscore character:
e.g. `my_variable_name` = "John"

Data Type in Python

Built-in Data Types

- In programming, data type is an important concept.
- Variables can store data of different types, and different types can do different things.
- Python has the following data types built-in by default, in these categories:

Text Type:	str
Numeric Types:	int, float, complex
Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview

Data Type in Python

- You can get the data type of any object by using the **type()** function

```
x = 5  
print (type (x) )
```


Setting the (implicit) Data Type

- In Python, the data type is set when you assign a value to a variable:

Example	Data Type
<code>x = "Hello World"</code>	str
<code>x = 20</code>	int
<code>x = 20.5</code>	float
<code>x = 1j</code>	complex
<code>x = ["apple", "banana", "cherry"]</code>	list
<code>x = ("apple", "banana", "cherry")</code>	tuple
<code>x = range(6)</code>	range
<code>x = {"name" : "John", "age" : 36}</code>	dict
<code>x = {"apple", "banana", "cherry"}</code>	set
<code>x = frozenset({"apple", "banana", "cherry"})</code>	frozenset
<code>x = True</code>	bool
<code>x = b"Hello"</code>	bytes
<code>x = bytearray(5)</code>	bytearray
<code>x = memoryview(bytes(5))</code>	memoryview

Setting the Specific Data Type

If you want to specify the data type, you can use the following **constructor functions**:

Example	Data Type
<code>x = str("Hello World")</code>	str
<code>x = int(20)</code>	int
<code>x = float(20.5)</code>	float
<code>x = complex(1j)</code>	complex
<code>x = list(("apple", "banana", "cherry"))</code>	list
<code>x = tuple(("apple", "banana", "cherry"))</code>	tuple
<code>x = range(6)</code>	range
<code>x = dict(name="John", age=36)</code>	dict
<code>x = set(("apple", "banana", "cherry"))</code>	set
<code>x = frozenset(("apple", "banana", "cherry"))</code>	frozenset
<code>x = bool(5)</code>	bool
<code>x = bytes(5)</code>	bytes
<code>x = bytearray(5)</code>	bytearray
<code>x = memoryview(bytes(5))</code>	memoryview

Python Libraries for Data Science

Some popular Python toolboxes/libraries:

- Data Processing and Modeling

- NumPy
- SciPy
- Pandas
- SciKit-Learn
- TensorFlow
- Keras

- Data Mining

- scrapy

Visualization libraries

- matplotlib
- Seaborn
- plotly
- pydot

..... and many more

Python Libraries for Data Science

Loading Python Libraries

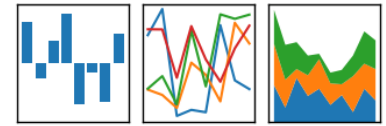
```
In [ ]: #Import Python Libraries  
import numpy as np  
import scipy as sp  
import pandas as pd  
import matplotlib as mpl  
import seaborn as sns
```

Hint: Press `Shift+Enter` to execute the current *jupyter cell*

Python Libraries for Data Science

NumPy

- NumPy (*Numerical Python*) is a perfect tool for scientific computing and performing basic and advanced *array operations*.
- introduces objects for *multidimensional arrays and matrices*, as well as functions that allow to easily perform advanced *mathematical and statistical operations* on those objects
- provides vectorization of mathematical operations on arrays and matrices which significantly improves the performance
- many other python libraries are built on NumPy



Python Libraries for Data Science

Pandas

- Pandas is a library created to help developers work with "*labeled*" and "*relational*" data.
- adds data structures and tools designed *to work with table-like data* (similar to Series and Data Frames in R)
- provides tools for *data manipulation*: reshaping, merging, sorting, slicing, aggregation etc.
- allows *handling missing data*

Python Libraries for Data Science

Reading data using pandas

```
In [ ]: #Read csv file  
df = pd.read_csv("Salaries.csv")
```

Note: The data import process.

There is a number of pandas commands to read other data formats:

```
pd.read_excel('myfile.xlsx', sheet_name='Sheet1', index_col=None, na_values=['NA'])  
pd.read_stata('myfile.dta')  
pd.read_sas('myfile.sas7bdat')  
pd.read_hdf('myfile.h5', 'df')
```

Python Libraries for Data Science

matplotlib

- A standard data science library that helps to *generate data visualizations* such as two-dimensional diagrams and graphs (histograms, line plots, pie charts, scatterplots).
- Matplotlib is one of those plotting libraries that are very useful in data science projects — it provides an object-oriented API for embedding plots into applications.
- relatively low-level; *some effort needed to create advanced visualization*

Python Libraries for Data Science

Seaborn

- Seaborn is *based on Matplotlib* and serves as a useful Python machine learning *tool for visualizing statistical models* – heatmaps and other types of visualizations that summarize data and depict the overall distributions.
- When using this library, you get to benefit from *an extensive gallery of visualizations* (including complex ones like time series, joint plots, and violin diagrams).
- provides *high level interface* for drawing attractive statistical graphics

Link: <https://seaborn.pydata.org/>

Python Libraries for Data Science

Plotly

- A web-based tool for data visualization that offers many useful out-of-box graphics – you can find them on the plotly website
- The library works very well in interactive web applications.
- Its creators are busy expanding the library with new graphics and features for supporting multiple linked views, animation, and crosstalk integration.

Python Libraries for Data Science

Scrapy

- One of the most popular Python data science libraries, Scrapy helps to *build crawling programs* (spider bots) that can retrieve structured data from the web – for example, URLs or contact info.
- It's a great tool for scraping data used in, for example, Python machine learning models.
- Developers use it for gathering data from APIs. This full-fledged framework follows the Don't Repeat Yourself principle in the design of its interface. As a result, the tool inspires users to write universal code that can be reused for building and scaling large crawlers.

Python Libraries for Data Science

SciPy:

- *A large collection of algorithms* for linear algebra, differential equations, numerical integration, optimization, statistics and more
- part of SciPy Stack
- built on NumPy

Python Libraries for Data Science

SciKit-Learn:

- This is an *industry-standard for data science projects* based in Python.
- Scikits is *a group of packages* in the SciPy Stack that were created for specific functionalities – for example, image processing. Scikit-learn uses the math operations of SciPy to expose a concise interface to the most common machine learning algorithms.
- Data scientists use it for handling *standard machine learning and data mining tasks* such as classification, clustering, regression, model selection, model validation, dimensionality reduction, and classification.
- built on NumPy, SciPy and matplotlib

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

Python Libraries for Data Science

TensorFlow

- TensorFlow is a popular Python framework for machine learning and deep learning, which was developed at Google Brain.
- It's the best tool for tasks like object identification, speech recognition, and many others. It helps in working with artificial neural networks that need to handle multiple data sets.
- The library includes various layer-helpers (tflearn, tf-slim, skflow), which make it even more functional.
- TensorFlow is constantly expanded with its new releases – including fixes in potential security vulnerabilities or improvements in the integration of TensorFlow and GPU.

Link: <https://www.tensorflow.org/>

Python Libraries for Data Science

Keras

- Keras is a *great library for building neural networks and modeling*.
- It's very *straightforward to use* and provides developers with a *good degree of extensibility*.
- The library takes advantage of other packages, (Theano or TensorFlow) as its backends. Moreover, Microsoft integrated CNTK (Microsoft Cognitive Toolkit) to serve as another backend.
- It's a *great pick if you want to experiment quickly* using compact systems – the minimalist approach to design really pays off!

Link: <https://keras.io/>