Investment Management

SETTING UP THE WORKSPACE

Python scripts can both be a great replacement and a great addition to using spreadsheets. Thanks to Python it is possible to make calculations over thousands of rows without breaking your spreadsheet and it is also possible to include new cutting-edge machine learning techniques for analysing data and text which are not available in MS Excel.

You can deploy Python locally (on your computer) as well as on a cloud server via a web browser. To level the playing field, in this course we opt for the latter approach. Should you choose to run Python on your local machine, there are numerous tutorials detailing how to set up your computer for Python development and get started with programming. Among others, you may want to have a look at this guide and also check out a detailed DataCamp tutorial explaining how to install, run and use **Jupyter Notebooks**. For most practical exercises in this course, we will be using **Colab Notebooks**, which are Jupyter notebooks hosted by **Google Colaboratory**. To find out more about the Jupyter project, see jupyter.org.

1. Python via a web browser

Python is a great tool for most applications involving data analysis. However, running Python scripts locally, on your computer, may prove tricky and there are several advantages to deploying it via a web browser:

- **No need for installation**. Local installation of a complete Python environment might be both complex and costly to support and maintain; making Python available via a web browser makes deployment much more efficient in certain scenarios.
- **Use of remote hardware**. Your local computer might not be able to perform complex, compute- and memory-intensive analytics tasks. At the same time, the use of shared servers with multiple cores, afforded by a cloud-based approach, makes such tasks possible and more efficient.
- **Collaboration**. Working with a team on the same servers makes collaboration simpler and increases efficiency. Also, different Python configurations, versions and modules installed on personal machines can often make it difficult to execute code written by others and, therefore, performs pertinent data analysis operations.

2. Why Use Google Colaboratory?

If you are new to data analytics, you might have never heard about **Google Colaboratory**, or **"Colab"** for short, before. Those with some prior Python experience would have most likely worked with Jupyter notebook, which is a widely used open-source application where you can create and share documents that contain live code, equations, visualizations and narrative text. Oftentimes, Google Colab is used as a tool to create, run and share Jupyter notebooks easier without installing anything on your computer. This research tool has several useful features for sharing works related to data analysis, machine learning and research. It is free of charge; you only need a Google account like GMail to use the service. Consider watching Introduction to Colab to find out more about this service.

Google Colab is a cloud-based platform for machine learning and other data science applications, created for research and education. It comes with a GPU (graphics processing unit) which is totally free and allows you to create and run Jupyter-like Python notebooks online. Colab has numerous common Python libraries pre-installed by default and thus requires little time to setup your project. You can save notebooks created in Colab to Google Drive (or GitHub) and open Jupyter notebooks from Google Drive or upload your existing Jupyter notebooks. You can also share your notebooks saved in your Google Drive, just like Google Sheets or Docs. Your shared notebook will include its full content (code, text, etc.) except for the virtual machine and custom libraries you have used. If you need to share this as well, you include additional cells in your notebook that automatically install and load the pertinent custom files.

Therefore, the main advantage of using Colab is that there is no requirement to install Python on your computer as all calculations are made in the cloud, with the calculation capacity provided by Google. Also, since you can easily share and collaborate your work with others, you can improve your Python skills and easily use the available libraries for your applications. If you used Jupyter notebooks before to run your code, then the platform will be quite familiar to you.

Colab can be used with common browsers, such as Chrome and Firefox. As you do not have to install any software to work on your scripts, you can simply run your notebooks using the browsers of your choice.

For convenience, all exercises accompanying this course are hosted on Google Colab. Each notebook contains the necessary code, explanatory text, and all custom modules and libraries.
To learn more about Colab, check out the tutorials and examples on the Colab welcome page, and read the Frequently Asked Questions (FAQ) here.

3. Differences between Google Colab and local Jupyter

Unlike Jupyter, Google Colab allows you to immediately run scripts without any prior installation or configuration. The steps involved are very similar to uploading and share your Google Docs or Sheets.

As it allows you to use the cloud service, you can take advantage of its GPU and computation capabilities. This is particularly important when it comes to machine learning. At the same time, you cannot use the service for applications requiring long-running computations, such as cryptocurrency mining. Note also that Colab is intended for interactive usage. If you require to continuously run scripts for your project, you need to use a local runtime through Colab's UI.

For those with prior Jupyter experience, working in Colab may feel dissimilar to working in Jupyter Notebook. In particular:

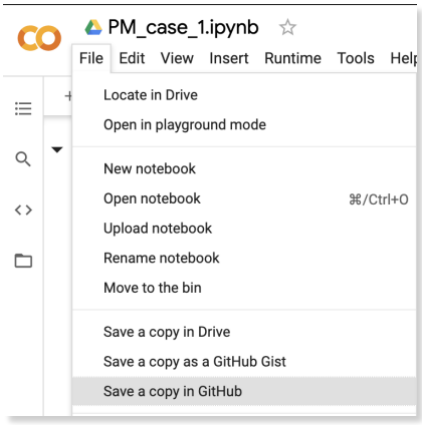
- some standards terms are named differently in Colab (e.g. "runtime" instead of "kernel", "text cell" instead of "markdown cell");
- "Command" and "Edit" modes in Colab and Jupyter work differently;
- most menu items in Colab are also different and some actions can only be done using keyboard shortcuts (see **Tools → Keyboard shortcuts**);
- notebooks cannot be downloaded in some formats available in Jupyter, such as an HTML webpage or a Markdown file;
- while you can upload a dataset to use within your Colab notebook, it will be automatically deleted once you end your session. Furthermore, Colab automatically disconnects the notebook if it is left idle for more than 30 minutes;
- when working on a notebook with several collaborators, edits are not visible to all collaborators in real-time as there may be a delay of up to 30 seconds. Therefore, your edits may get lost if multiple people are editing the notebook at the same time.

4. Ways to create and share a Colab project

Creating a new notebook:

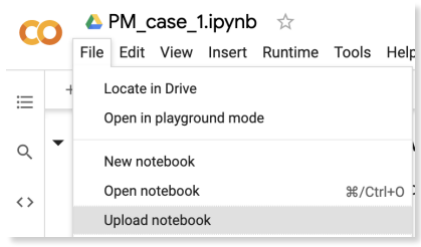
- register and/or sign in to your Google account, such as a GMail account;
- head over to Colab, click on **New Notebook** or **File → New Notebook** and choose Python 3 to create a new notebook;
- by default the new notebook is saved in your Google Drive under "**Colab Notebooks"**. Since the notebook is in your Google Drive, you can move it around and share it with others in the same way as any other Google Drive documents.

Sharing a notebook on GitHub (if you use this service):



- click on **File → Save a copy in Github**, you will be prompted to log in to your Github account and authorize app. Once done, you would be able to save your notebooks to a chosen repository on GitHub. Each time you save, you can add a commit message for GitHub.
- Once the notebook is saved to GitHub, you will get a "**View in Colaboratory**" link. The link allows you to open the notebook directly in Colab.
- Anyone opening the shared notebook would not need to do any additional setup to run the script.

Opening an existing Jupyter Notebook in Colab:



- If you have an existing Jupyter Notebook simply click on **File → Upload notebook** to open it in Colab.
- If the notebook has already been saved in your Google Drive, open it by clicking **File → Open notebook → Google Drive**.
- If your notebook is hosted on GitHub, open it by clicking **File → Open notebook → GitHub → enter a GitHub URL**.
- Once loaded, you can continue to work on your notebook. The Jupyter and Colab's user interfaces are very similar, with some minor differences. Colab text cells offer more text formatting capabilities. You can also access the table of contents for your notebook by selecting **View → Table of contents**.

5. Useful Colab tips

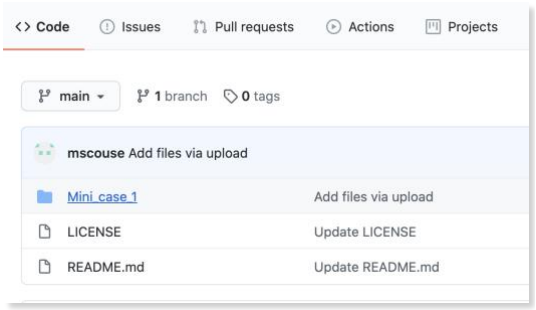
Uploading and downloading data files to Colab

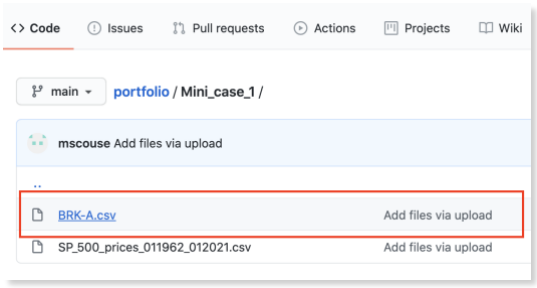
- To perform data analysis, the first step is to load a file containing the pertinent data – such as a CSV or Excel file - into Colab. There are several ways to do so.
- **Method 1**: upload a CSV file from your GitHub repository, if one is available:

  (1) create a new "Code" cell in your Colab notebook and from there import a widely used Python library for data manipulation and analysis, called Pandas, as shown below

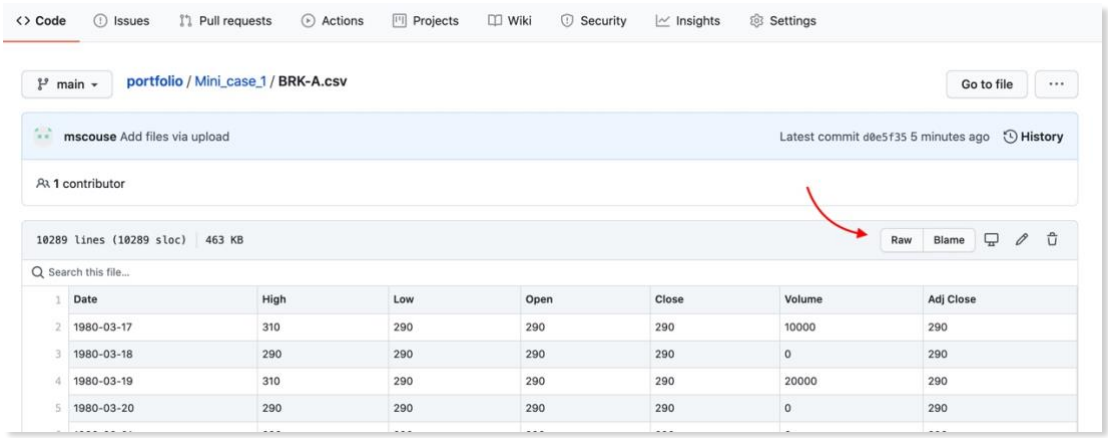

```
[ ]  import pandas as pd
```

(2) In your GitHub account, select the pertinent repository containing the data and click the dataset in that repository.





(3) Click on **View Raw**. Copy the link to the raw dataset and store it as a string variable called `url` in Colab as shown below




```
[ ]  # Copy the link to your GitHub dataser and store it as a string titled "url"
     # url = 'copied_raw_GH_link'

     url = 'https://raw.githubusercontent.com/mscouse/portfolio/main/Mini_case_1/BRK-A.csv?token=AHBGPJTWQVBNDG7WDY6L7PDAAHVTY'
```

(4) The final step is to load the `url` into Pandas using the `read_csv` method to get a dataframe containing your data and check it

```
[32] # Load the dataset into Pandas. The dataset will be stored as a Pandas Dataframe "df"
     df = pd.read_csv(url)
     df.head()
```

- **Method 2**: upload a CSV file from your local drive:

  (1) To upload the data from your local drive, type in the following code in a new "Code" cell in Colab:

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

  (2) Once executed, the code will prompt you to select a file containing your data. Click on "**Choose Files**" then select and upload the file. Wait for the file to be 100% uploaded. You should see the name of the file once it is uploaded:

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

```
Choose Files  BRK-A.csv
• BRK-A.csv(text/csv) - 790959 bytes, last modified: 14/06/2019 - 100% done
Saving BRK-A.csv to BRK-A (1).csv
{'BRK-A.csv': b'Date,Open,High,Low,Close,Adj Close,Volume\n1980-03-17,290.00000
```

  (3) Type in the following code to import the data into a Pandas dataframe. The "`filename.csv`" should match the name of the uploaded file, including the .csv extension – "`BRK-A.csv`" in this example:

```
# Load the dataset into Pandas. The dataset will be stored as a Pandas Dataframe "df2"
df2 = pd.read_csv('BRK-A.csv')
df2.head()
```

index	Date	Open	High	Low	Close	Adj Cl
0	1980-03-17	290.0	310.0	290.0	290.0	
1	1980-03-18	290.0	290.0	290.0	290.0	
2	1980-03-19	290.0	310.0	290.0	290.0	
3	1980-03-20	290.0	290.0	290.0	290.0	
4	1980-03-21	290.0	290.0	290.0	290.0	

Show 25 per page

- **Method 3**: upload data files from your Google drive:

  (1) To upload data stored on Google Drive, create a dedicated "**data**" folder in the location of your choosing. This is where you should upload your data files.

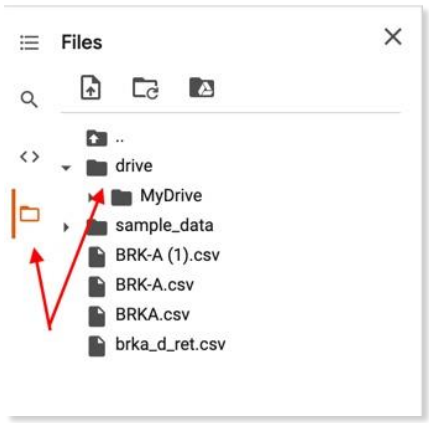
  (2) From a Colab notebook, type in the following code in a new "Code" cell:

```
from google.colab import drive
drive.mount('/content/drive')
```

  (3) Once executed, the commands in the cell above will bring you to the Google Authentication step. A screen with the "**Google Drive wants to access your Google Account**" message will appear, asking you to permit access.

  (4) After you allow permission, copy the verification code provided and paste it in the required box in the Colab notebook. You should now be able to access

your Google Drive files by clicking on the folder icon on the top left of the notebook and locating a folder named "Drive" → "MyDrive".



(5) Within the "MyDrive" folder, locate the **data** folder you created earlier and find the data file you want to import.

(6) Right-click on your data file and select **Copy Path**. Store this **copied path** into a `path` variable and use the following code to import the data into a Pandas dataframe:

```
path="copied path"
df3=pd.read_excel(path)
df3.head()
```

Enabling GPU acceleration

- For complex computations, you may want to take advantage of Colab's free GPU.
- To enable GPU acceleration, select **Runtime/change runtime type**, select **GPU for hardware acceleration**. You can run your training models on GPU for free.

Preventing Colab from disconnecting

- One drawback of Google Colab is that it automatically disconnects the notebook if left idle for more than 30 minutes.
- To prevent Colab from disconnecting, (1) open your project in Google Chrome, (2) open your Chrome DevTools by pressing `F12` or `ctrl+shift+i` on Linux or `command+option+i` on Mac and (3) enter the following JavaScript snippet in your **console**:

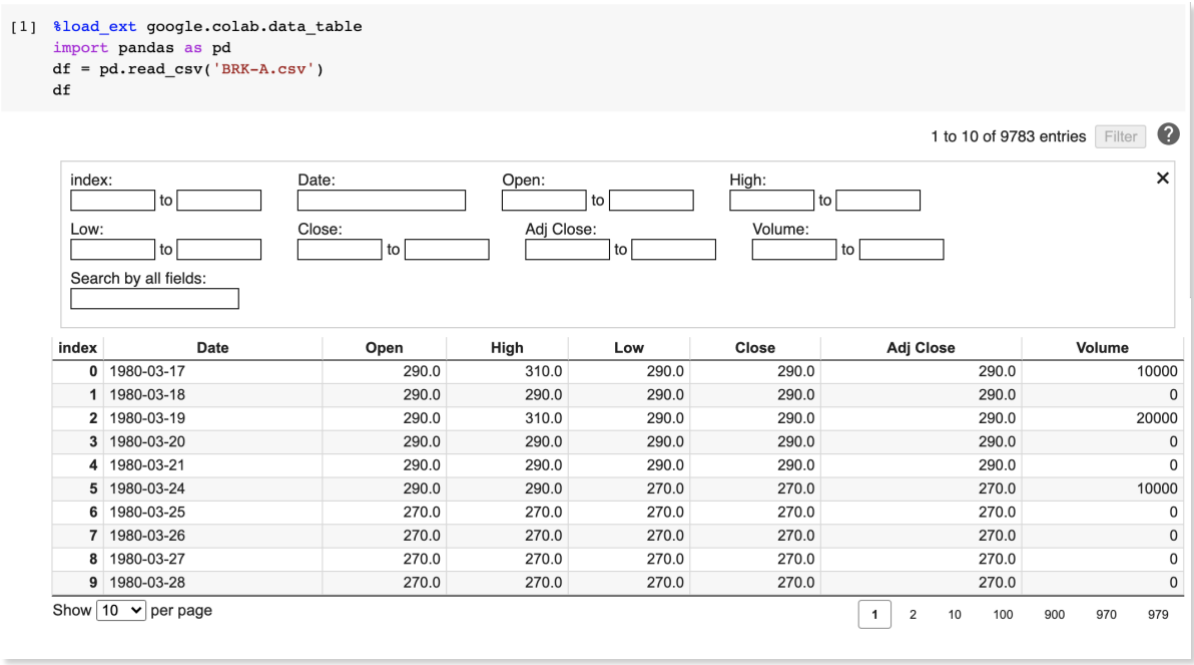
```
function KeepClicking(){
console.log("Clicking");
document.querySelector("colab-connect-button").click()}
setInterval(KeepClicking,60000)
```

- The JavaScript function above makes a click on the connect-button every 60 seconds. This ensures the notebook is not idle and you should not be disconnected.

Displaying Pandas Dataframes as interactive tables

- Colab includes an extension that renders Pandas dataframes into interactive tables that can be filtered, sorted, and explored dynamically.

- The extension can be enabled with `%load_ext google.colab.data_table` and disabled with `%unload_ext google.colab.data_table`:



```
[1] %load_ext google.colab.data_table
    import pandas as pd
    df = pd.read_csv('BRK-A.csv')
    df
```

1 to 10 of 9783 entries Filter

index: to Date: Open: to High: to
Low: to Close: to Adj Close: to Volume: to
Search by all fields:

index	Date	Open	High	Low	Close	Adj Close	Volume
0	1980-03-17	290.0	310.0	290.0	290.0	290.0	10000
1	1980-03-18	290.0	290.0	290.0	290.0	290.0	0
2	1980-03-19	290.0	310.0	290.0	290.0	290.0	20000
3	1980-03-20	290.0	290.0	290.0	290.0	290.0	0
4	1980-03-21	290.0	290.0	290.0	290.0	290.0	0
5	1980-03-24	290.0	290.0	270.0	270.0	270.0	10000
6	1980-03-25	270.0	270.0	270.0	270.0	270.0	0
7	1980-03-26	270.0	270.0	270.0	270.0	270.0	0
8	1980-03-27	270.0	270.0	270.0	270.0	270.0	0
9	1980-03-28	270.0	270.0	270.0	270.0	270.0	0

Show 10 per page 1 2 10 100 900 970 979

Changing display mode

- If you prefer to use a "dark" display mode, you can activate it by clicking **Tools** → **Settings** → **Theme:**