

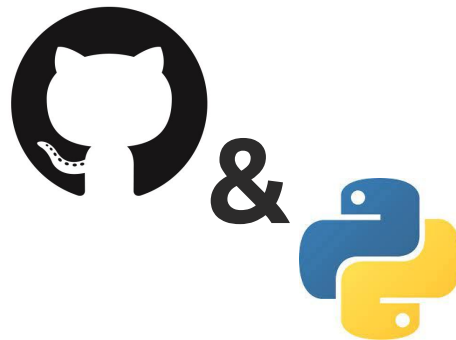
Data Science Survival Skills

Exercise 3

Description of the exercise

Welcome to the third exercise of Data Science Survival Skills. With this activity you will learn:

- What is GitHub.
- How to create an account and your first repository.
- How to set packages in your repo.
- Use the packages and a virtual environment.



GitHub: Overview

GitHub is a tool that helps us track our codes. You can commit and push changes to keep a record of what you are doing, work without affecting the original program, work simultaneously with different people on the same project, and even deploy your codes or websites.



GitHub: Overview

We can decide if we want to use Git with either a GUI or the command line.

Some of the main commands of GitHub are: **clone**, **status**, **branch**, **checkout**, **add**, **commit**, **push** and **pull**.



Anaconda

It is a Python distribution platform that allows us to install and manage Python libraries together with several Virtual Environments. Its installation comes with various packages and elements such as Jupyter Notebook, Spyder, Pycharm, and its GUI: Anaconda Navigator.



Prerequisites

To carry out the tasks of this exercise, you need to install two essential components, namely Git and Anaconda. Their installation is straightforward, so you only have to follow the instructions.

- [Link to Git](#)
- [Link to Anaconda](#)



1. Create your first repository

1. Task: Create your first repository

The first step is to set an account on [GitHub](#). After following the instructions, the subsequent procedure is to create your first repo!

To do it, you can press the green button 'New' on the feed or follow:

Profile → Repositories →



1. Task: Create your first repository


As a repository name, use your IdM-ID (this will help us later). We recommend always writing a short description; it helps everyone who checks your work (and yourself) to identify the contents of your repo.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

Repository name *

 haaguileraa ▾

/ idm_id ✓

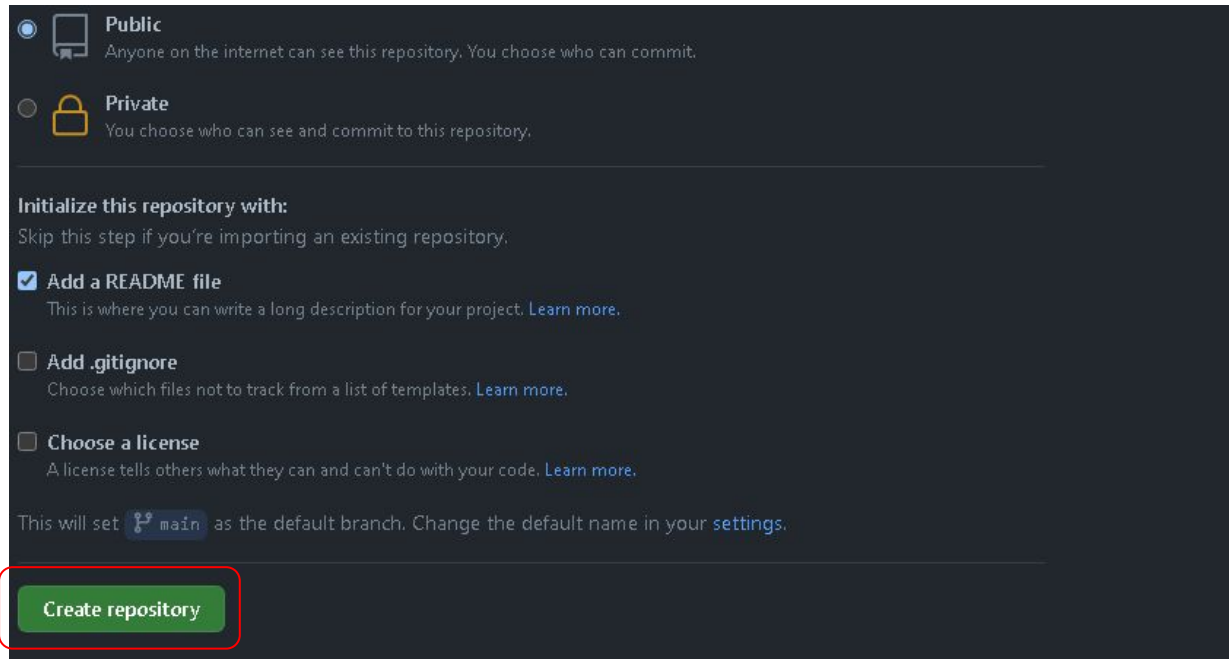
Great repository names are short and memorable. Need inspiration? How about [automatic-fortnight?](#)

Description (optional)

This is a repository containing the files of DSSS.

1. Task: Create your first repository

For this example, it is necessary to create a public repository. You can check the 'Add a README file' cell and leave 'Add .gitignore' and 'Choose a license' without a checkmark.



☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

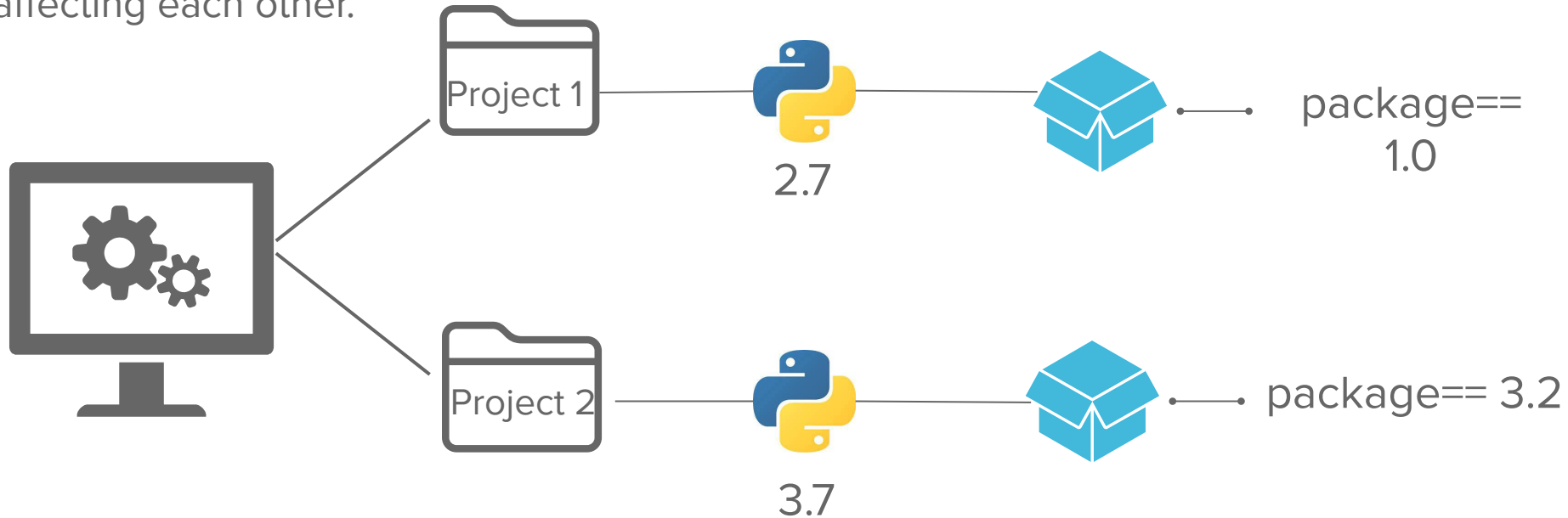
This will set `main` as the default branch. Change the default name in your [settings](#).

Create repository

2. Virtual Environment

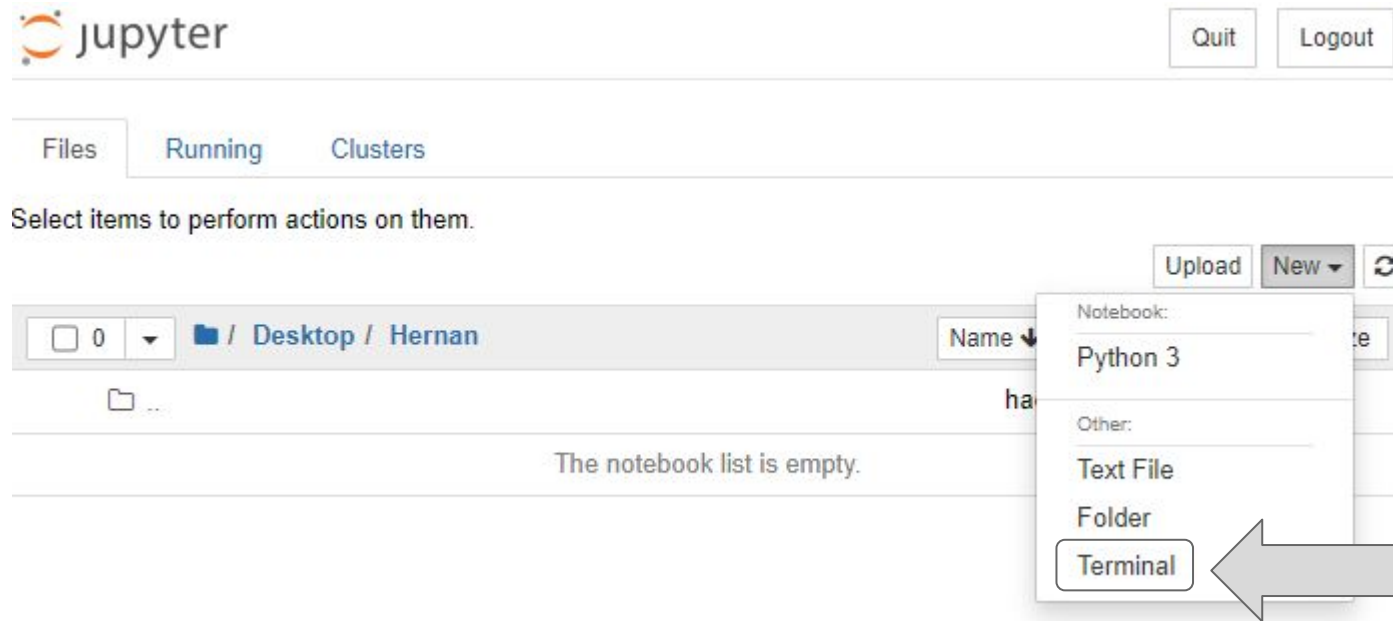
Virtualenv

It allows us to have isolated the packages needed to work in a specific project; i.e., these packages don't affect the base Python installation. Using a virtual environment, you can use different versions of the same package for several programs without affecting each other.



Jupyter Notebooks and venv

At this point, you should open Jupyter Notebook using Anaconda Navigator and select a web browser to run the notebooks. Subsequently, you will choose under "New" the option "Terminal."



Activating Venv

To create and activate your virtual environment we need to write the following on the command prompt:

conda create -n myvenv python=3.7

You will see the request “*Proceed ([y]/n)?*” from the command prompt. Press ‘y’ and then enter. This will create the virtual environment under the name “myvenv”. Finally, you can activate the venv by writing

conda activate myvenv

After the activation of the venv, you will see its name in front of the command line.

You can deactivate it using: ***conda deactivate***

Using venv

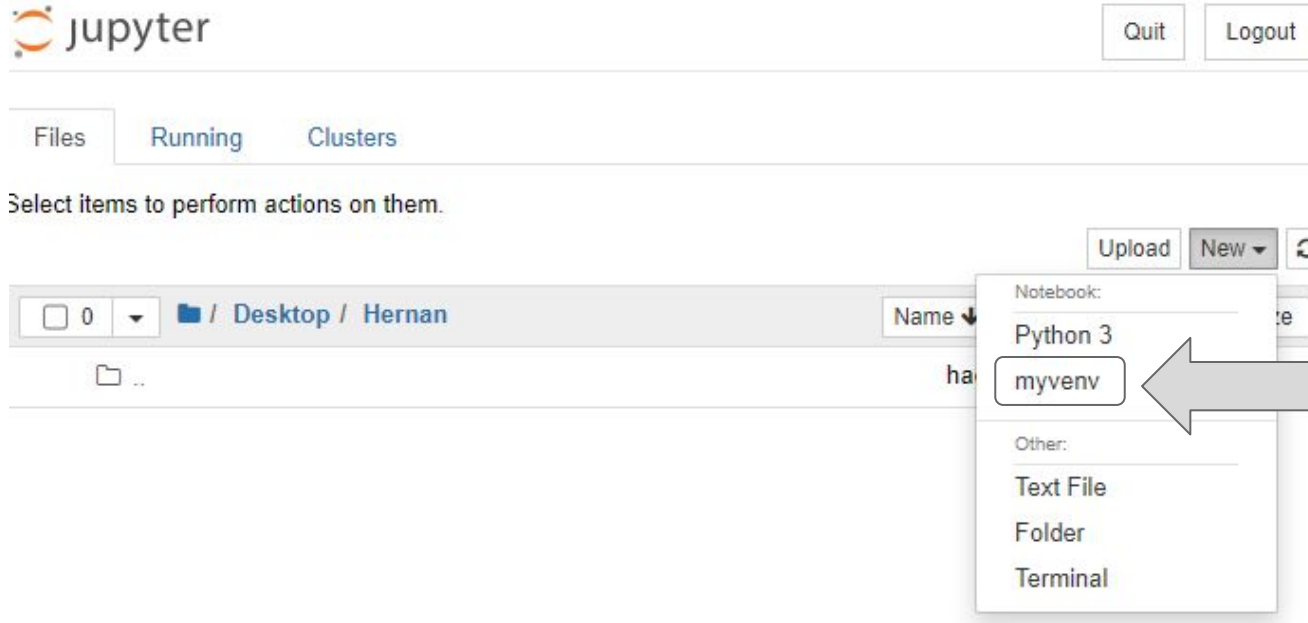
Now we want to use this virtual environment within Jupyter Notebook. To do it, we can install and specify the kernel using the commands:

```
pip install --user ipykernel
```

```
ipython kernel install --user --name=myvenv
```

Using venv

If there are no errors, you can refresh the tab of the notebook and find the venv like:

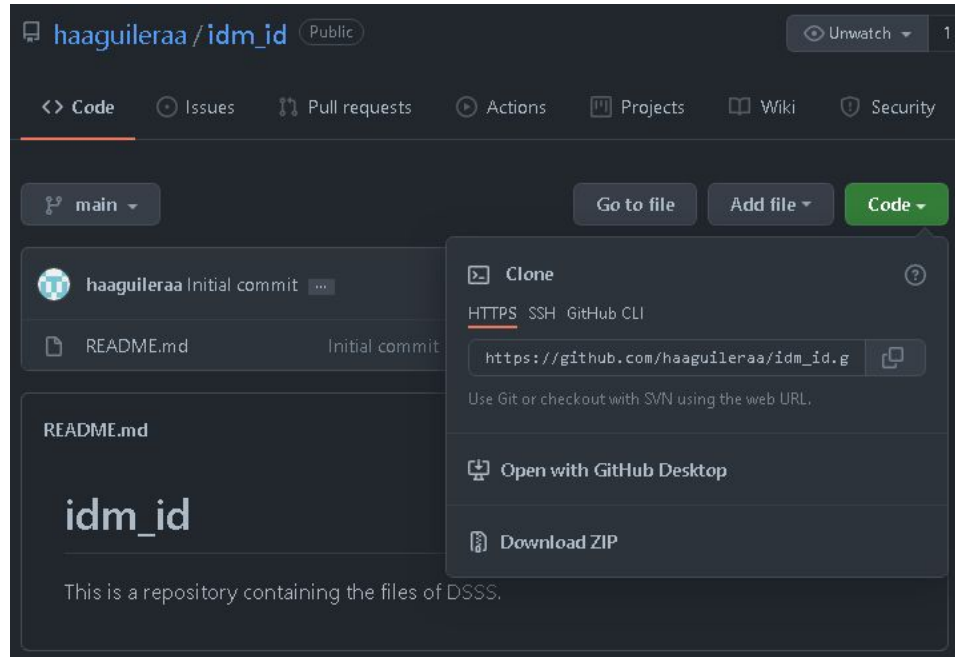


This new file will be running using the virtual environment that we just created, and you can install the packages inside the notebook as we did in Google Colab.

3. Working with GitHub

Cloning your repo

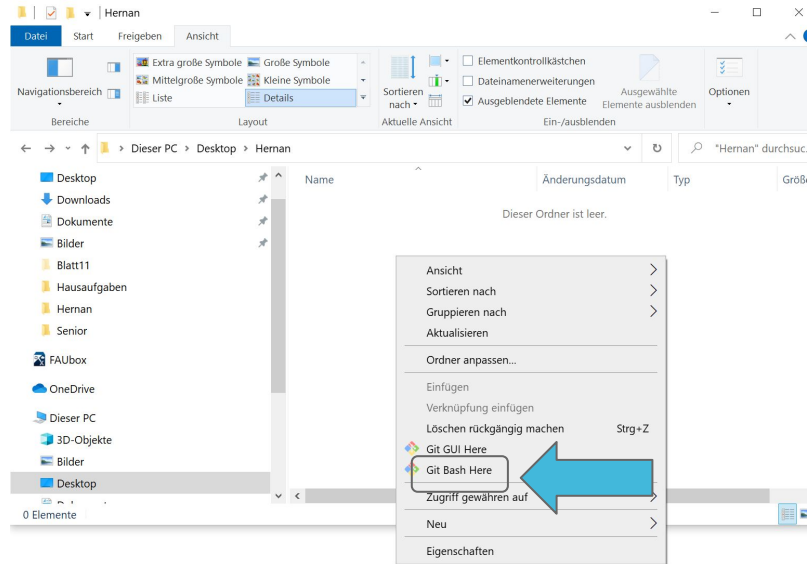
To clone the repository that you created before, you need its link to download. GitHub gives you three options to do it: HTTPS, SSH or GitHub CLI. In this case we will copy the HTTPS.



Cloning your repo

Go to a desired folder and open a git bash there

On Windows: right click -> Git Bash Here



You will find afterwards a folder containing the files from your repo. Open this folder and create a new Jupyter Notebook file named “git_config”

Cloning your repo

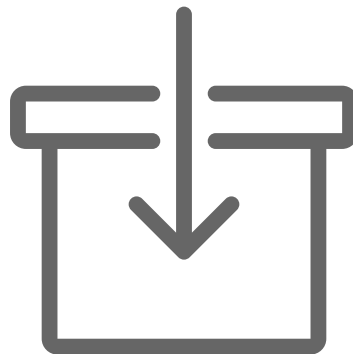
Write there the link that we copy before as:

```
git clone HTTPS_link
```

You will find afterwards a folder containing the files from your repo. To access it via command line, write:

```
cd name_folder/
```

and leave this Bash open.



Add, commit, push, pull...

Let us now create our first file! The first step is to use Jupyter's file explorer and move inside our just downloaded folder. There, you will create a new Notebook using our virtual env and name it as "mypackage."

Task: Copy the code of the next chapter and follow the instructions of slide 29.

In order to grade this task, we will install it and check it directly in our virtual environment.

We will now push this file to your repo!



Add, commit, push...

Go back to the Git Bash and write

git add -A

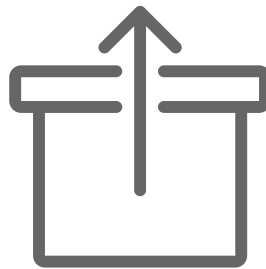
This line will add all our files to the queue and prepares them to commit. Then

git commit -m “comments: committing my first package”

This command will prepare all your files to be pushed. Finally,

git push

In this case, you will push the files committed in your repository. You need then to follow the instructions for adding your credentials to access GitHub.



Problem with credentials

If your computer has already an account you can solve it going to:

Control Panel\ All Control Panel Items\Credential Manager

or

Systemsteuerung\Alle Systemsteuerungselemente\Anmeldeinformationsverwaltung

There, you can edit the credentials used to access GitHub.



Git pull, status

Another necessary feature is the command ***git status***. It assists us to check some information about our working branch, i.e., the state of the working directory.

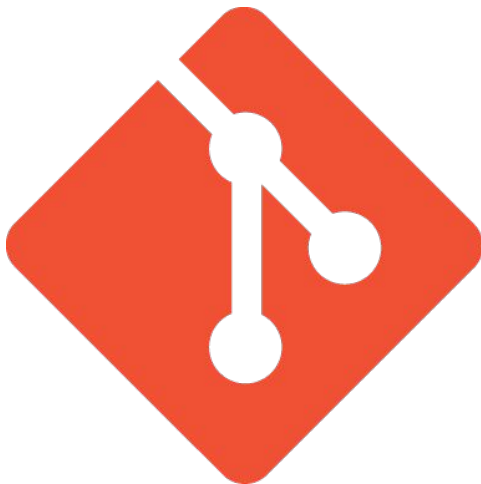
Now, using the command ***git pull***, you will retrieve the information from remote changes in your current working repository; thus, it will update your files.



Working with branches

Branching and merging is a helpful feature of Git, and we will need it in our projects. [This Video on youtube](#) explains how branching works; you can also follow the full tutorial to learn more about Git.

You can find more information about Git [here](#).



4. Creating your own installable package

Structure of the repository

To create a pip-installable package, we need to follow the next structure:

.

└─ idm_id # Folder containing our set of functions

└─ └─ __init__.py

└─ └─ function.py

└─ setup.py

└─ requirements.txt

└─ README.md

└─ .gitignore

`__init__.py`

File used to indicate that this folder contains a Python package.

```
from .function import *
```

function.py

File containing our functions **(this is your task)**:

```
import numpy as np
from ipywidgets import interact, fixed
from PIL import Image

def imshow(X, resize=None):
    """
    You should create a way to resize an image from an array X.
    The use of widgets is optional but you can take a look to interact.
    We should be able to install this package in Google Colab from your Git
    repo.

    """
    pass
```

setup.py

You have to generate a set-up to ensure the functioning of your package:

```
from distutils.core import setup
from setuptools import find_packages

setup(name='idm_id',
      version='0.1',
      author='DSSS',
      author_email='name@fau.de',
      packages=find_packages(),
      install_requires=['numpy', 'Pillow', 'ipywidgets'])
```

requirements.txt

You can also install packages using this file and the command line:

```
pip install -r requirements.txt
```

```
##### You can write here which packages are necessary to
run your code #####
##You can specify the version using '=='. For example:
numpy==1.21.2
numpy
Pillow
ipywidgets
```

Other options to specify the version of a package are:

```
Numpy >=1.0 , <1.21.2 # For 1.0 or greater but not 1.21.2 or higher
Numpy ~= 1.21.2    #For a compatible release
```

Among others ([Documentation](#)).

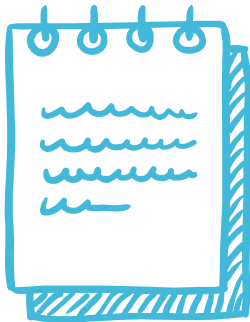
.gitignore

Git is a helpful file that specifies which folders and scripts Git shouldn't push into your repository but, instead, you are using locally.

More info: <https://git-scm.com/docs/gitignore>

Evaluation of results

We should be able to run `! pip install git+link_of_your_repo` on Google Colab or Jupyter Notebook and use the function that you created previously.



And we are done!

Thank you