KADI

TRAINING ON 28TH APRIL 2025, KAPITI FARM:

Prepared by Levis Muse, Silas Issa and Mathew Mutuku

Table of Contents

Table of Contents **1**
MORNING SESSION: **2**
Recap for Sunday 2
GIT 2
Terminal 2
Configuration: 3
Exercise 5
Deleting files 5
Branches 6
COLLABORATIONS 7
SECRET FILES 7
AFTER TEA SESSION : **9**
VIRTUAL ENVIRONMENTS 9
Creating an environment 9
PACKAGES 10
EXTENSIONs 12
Jupyter 12
AutoDoc 12
LUNCH BREAK **13**
AFTERNOON SESSION: **13**
DATA 13
OPENING WITH JUPYTER NOTEBOOK 13
Kernel selection 13
Alternate way Jupyter 14
CLONING DATA FROM REPO 14
BRANCHING 14
PYTHON FUNCTIONS 15
Open () 15
Help () 16
Exercise 16
Evening Session **17**
DATA TYPES 17
Loop Functions 17
Example 1 18
Example 2 18

MORNING SESSION:

Morning walk

Recap for Sunday

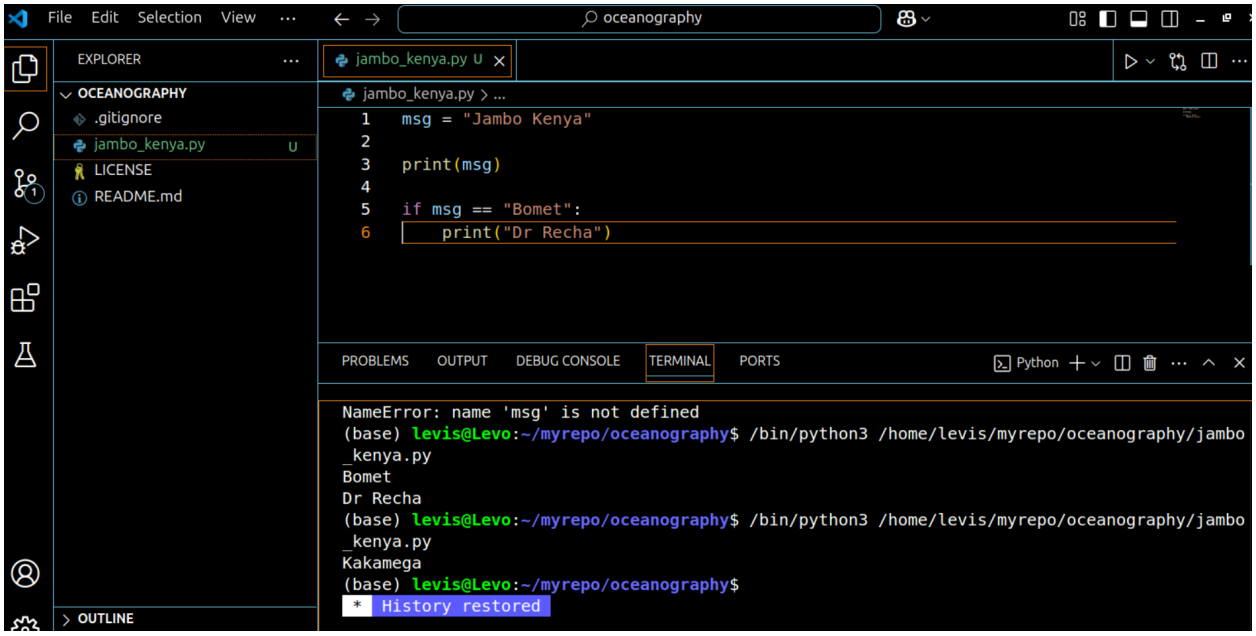
GIT

The purpose was to integrate GIT into the visual studio workflow.

Terminal

The layout of visual studio was explained with its main components including the explorer, terminal, source control and extensions.

Accessing and using the terminal is shown



The terminal gives an additional feature to run commands in visual studio.
The following command

> git status

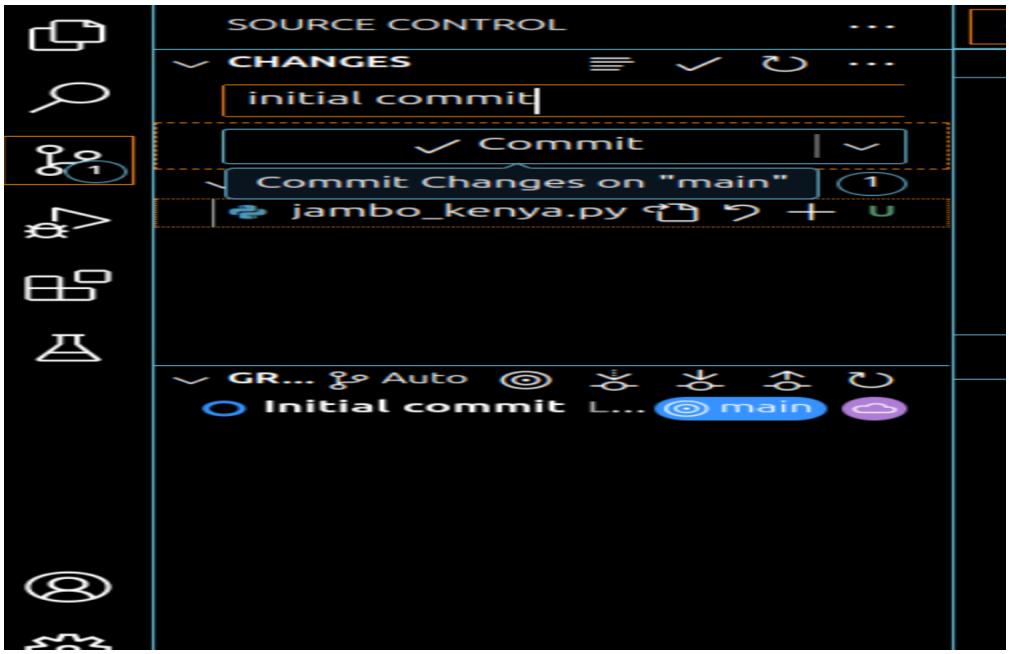
```
(.venv) (base) levis@Levo:~/myrepo/oceanography$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
(.venv) (base) levis@Levo:~/myrepo/oceanography$
```

Source control

The ability to track changes and be able to work on pending changes to the remote repo.

- ➢ Make sure the file changes have been saved
- ➢ Stage a file changes
- ➢ Message for accompany the changes
- ➢ Commit a file
- ➢ Sync changes
- ➢ Once completed look at the Graph for the branch



Click ✚ next to your file name to stage

Then write initial commit message in the message box: write meaningful commit messages

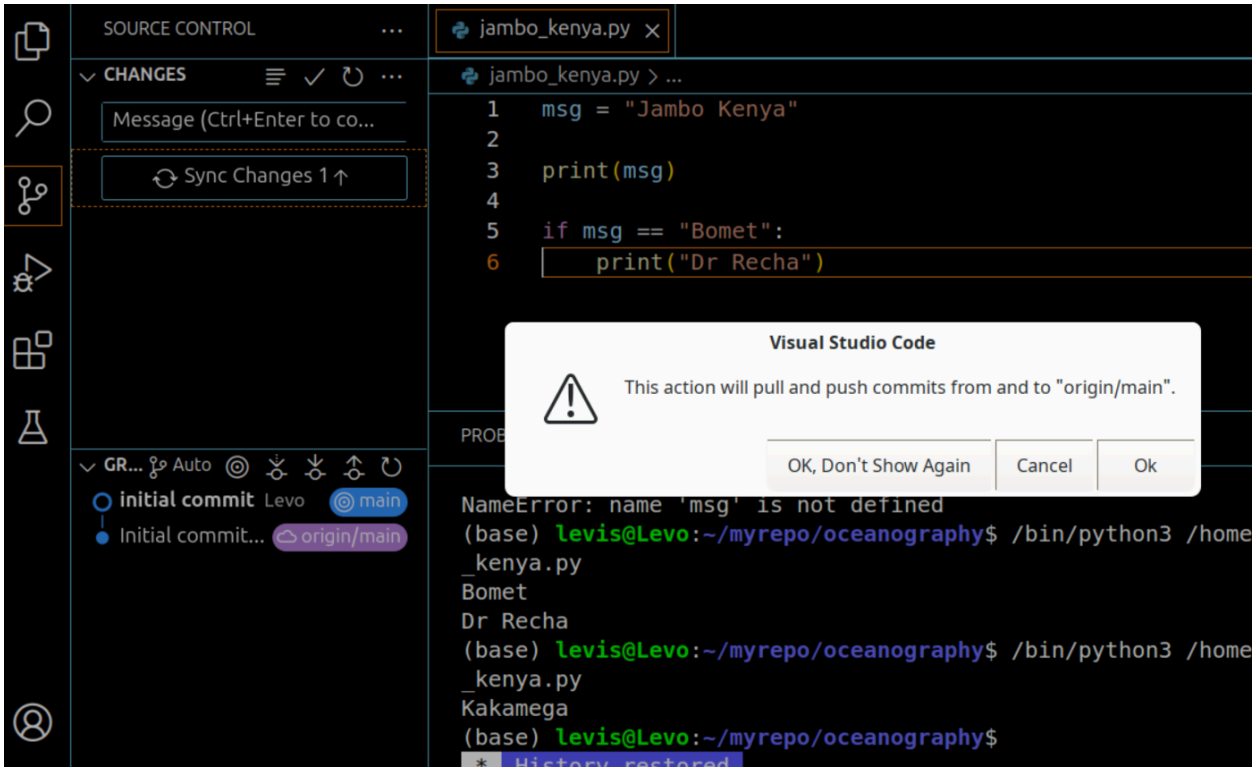
Commit the file

Configuration:

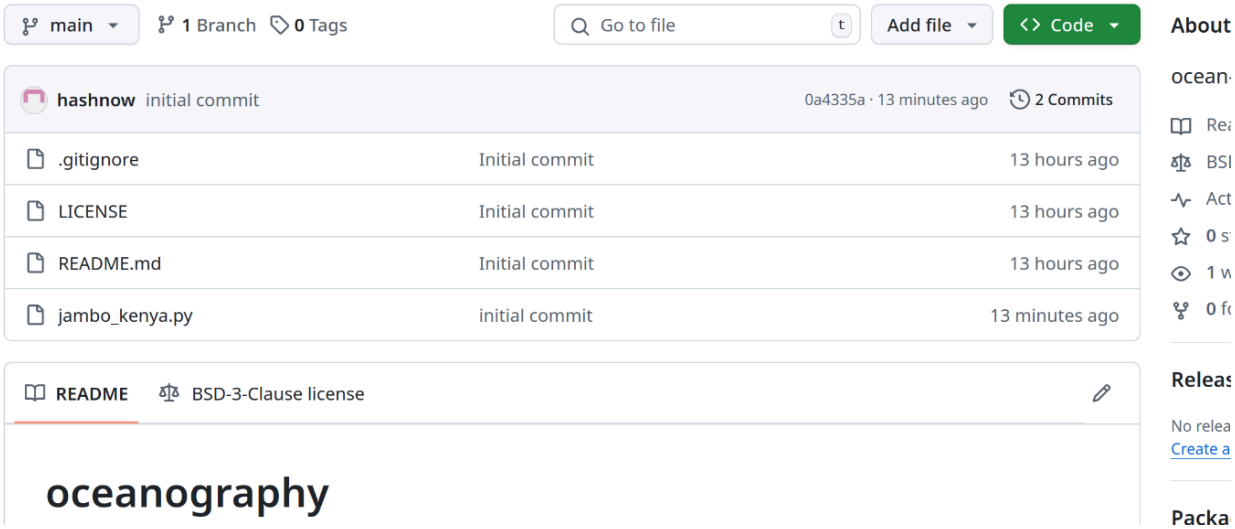
On the terminal configure the following

- **Set username:** Use the command `git config --global user.name "Levo"`.
- **Set email:** Use the command `git config --global user.email "levisjmuse@gmail.com"`.

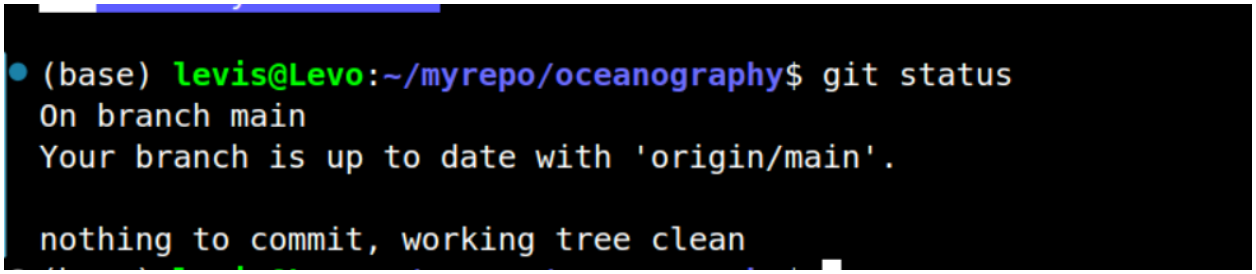
Sync changes



Visit the website of GIT.
You will see the file uploaded in your remote repository



Git status



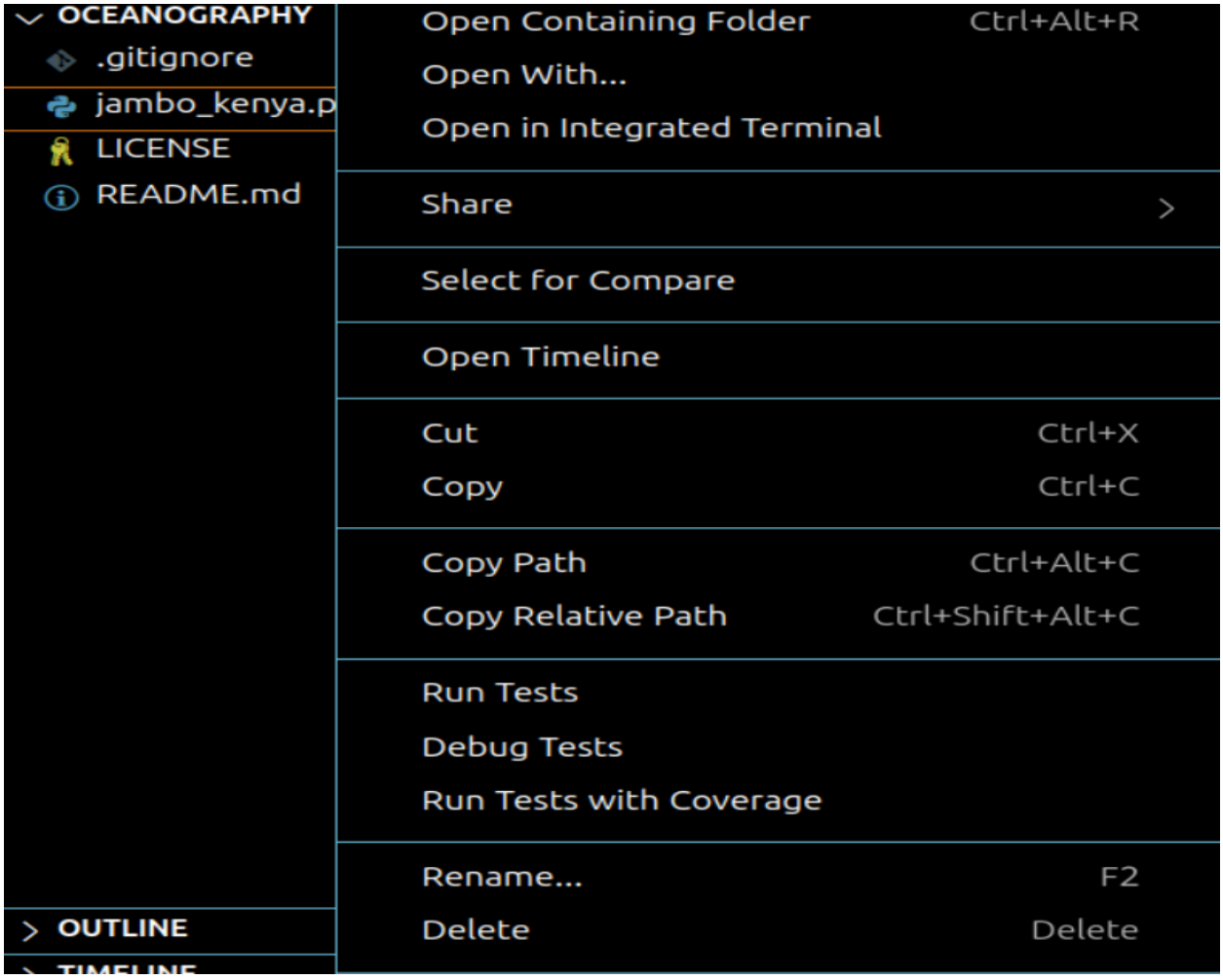
Using the terminal commands
>git commit - m “feat(text.txt)”
> git push

Exercise

1. Clone python data gym repository: https://github.com/KADI-Project/Python-Data-Gym
2. Forke a classmate repo
3. Create and merge a branch

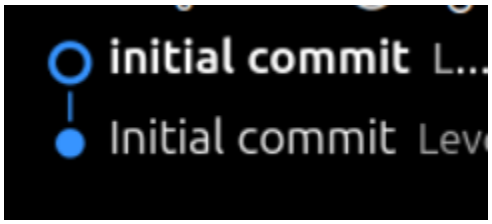
Deleting files

- Move to trash
- Delete



Check the source control to commit the changes

Branches



- Create new branches
- Merge branches
- Main branch

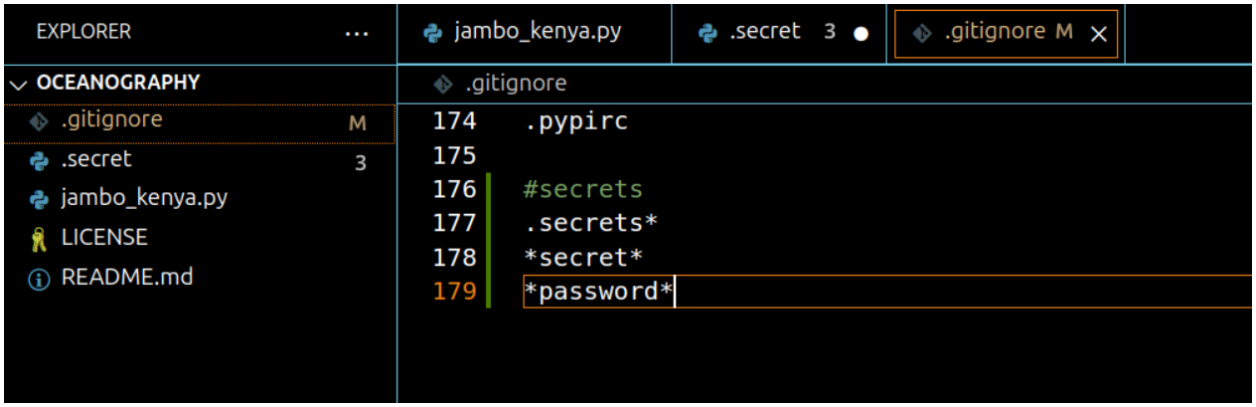
COLLABORATIONS

Wiki page
Github handles
https://github.com/hashnow/
Watch: if changes are made
Create a fork
Security details

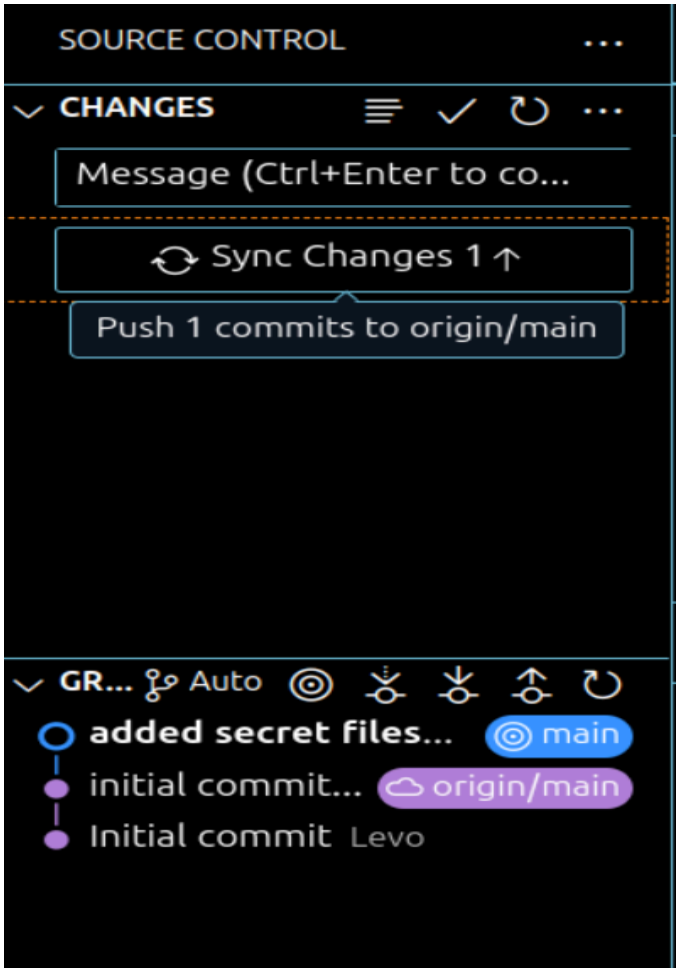
SECRET FILES

File name .secret

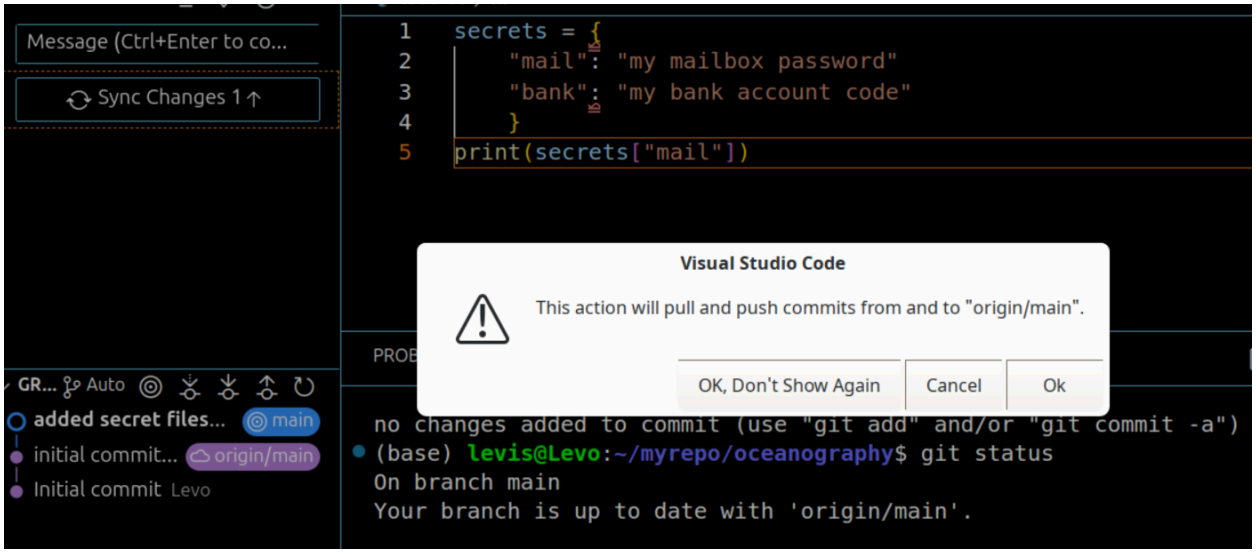
Open gitignore
Add the secret file words



Commit the changes in the source control
Observe the graph



Confirm the changes



> git status

Do not commit .secrets or .venv

AFTER TEA SESSION :

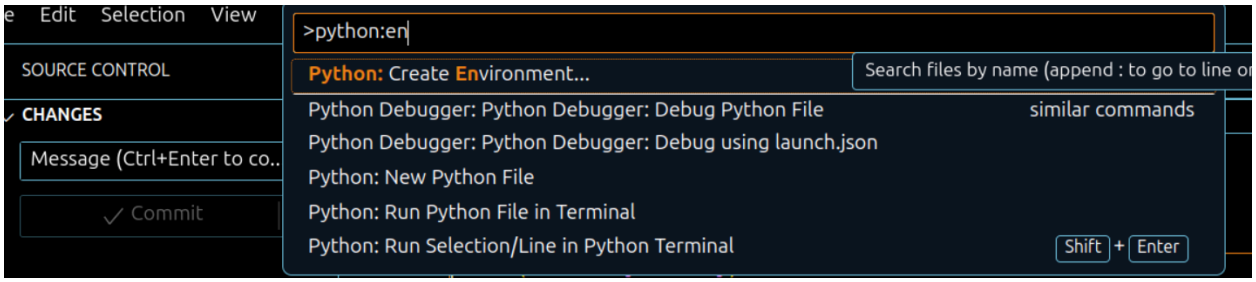
VIRTUAL ENVIRONMENTS

One project = one environment
Create little boma with everything that you need: that is an environment with all the libraries that you need.

Creating an environment

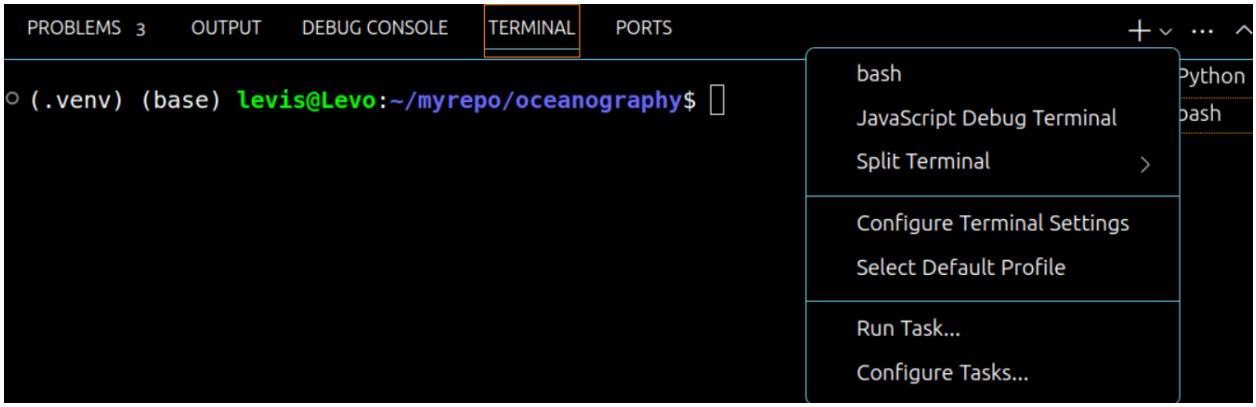
1. Venv
2. Conda

On the search box type: >python :envir
Select python: creat



Choose “venv” and complete the process
Activate the environment
linux : Source .venv/bin/activate
Poweshell (PS): .venv\Scripts\Activate.ps1

Access different shells

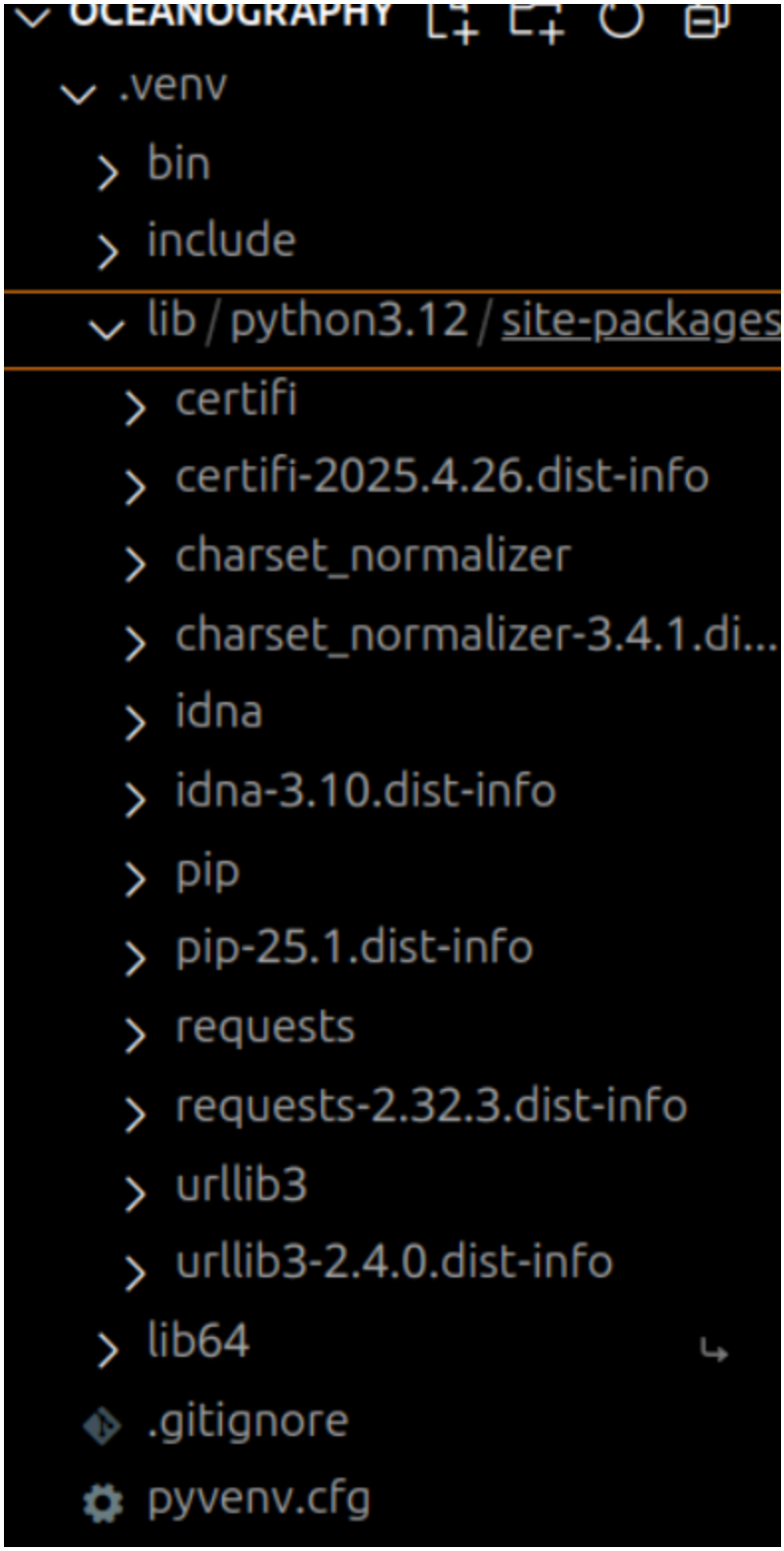


> deactivate – to leave the environment
Make sure you have (.venv)

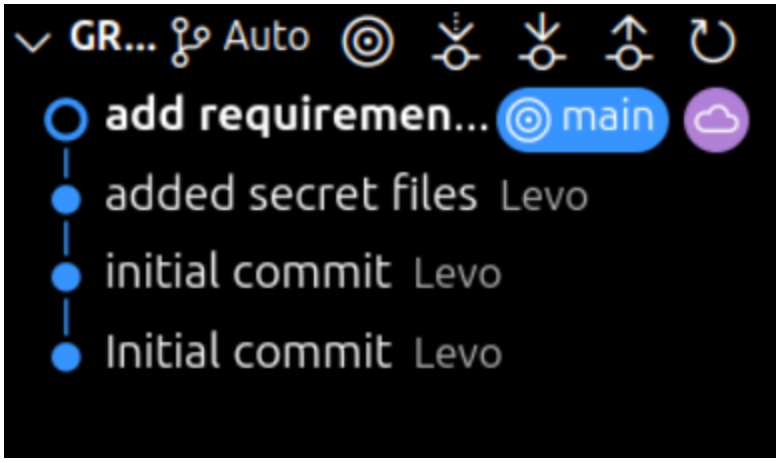
PACKAGES

Use: pip install
>pip install requests
> pip install polars
> pip install matplotlib
> pip install pandas
> pip freeze > requirements.txt

In the venv/lib path see libraries installed



In the source control commit and sync the changes for requirements file



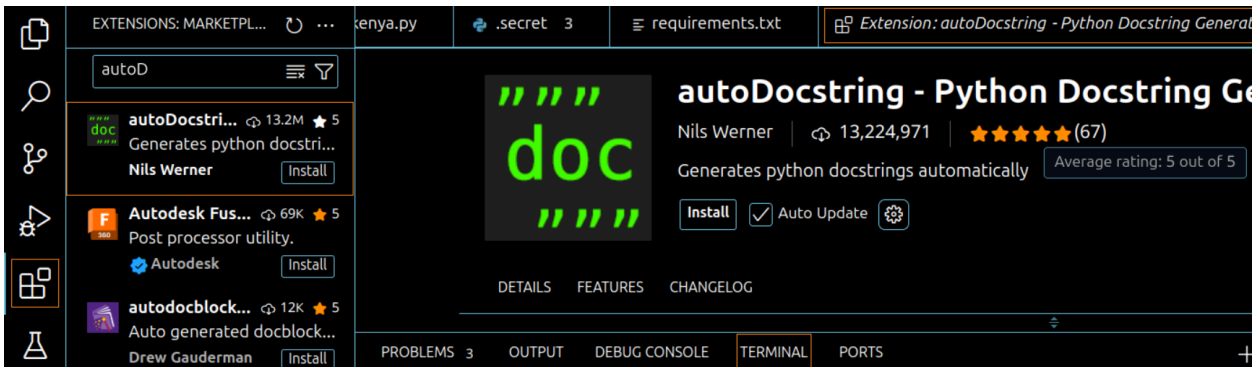
EXTENSIONs

Jupyter

Install jupyter from the extension

AutoDoc

Install AutoDoc



LUNCH BREAK

AFTERNOON SESSION:

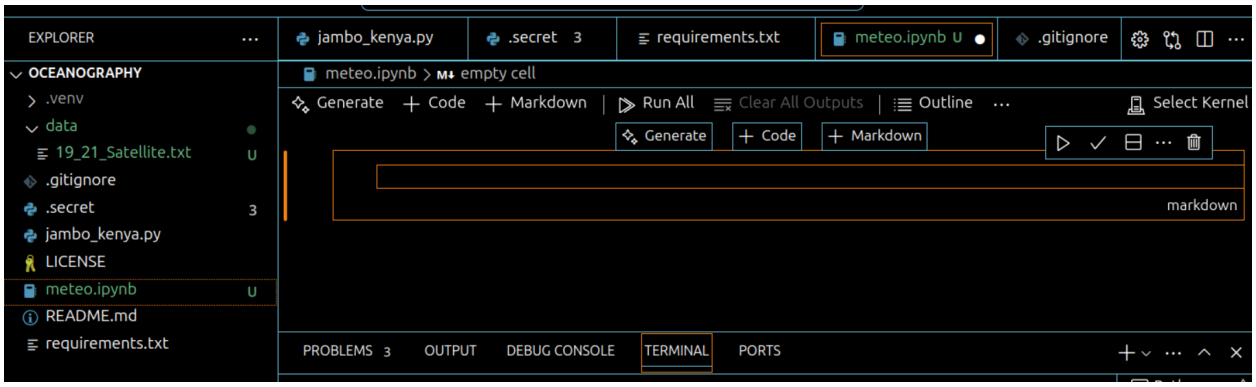
DATA

Create a data folder

- Data formats
  - CSV, TXT
  - Excel
  - netcdf
- Reading csv files
  -
-

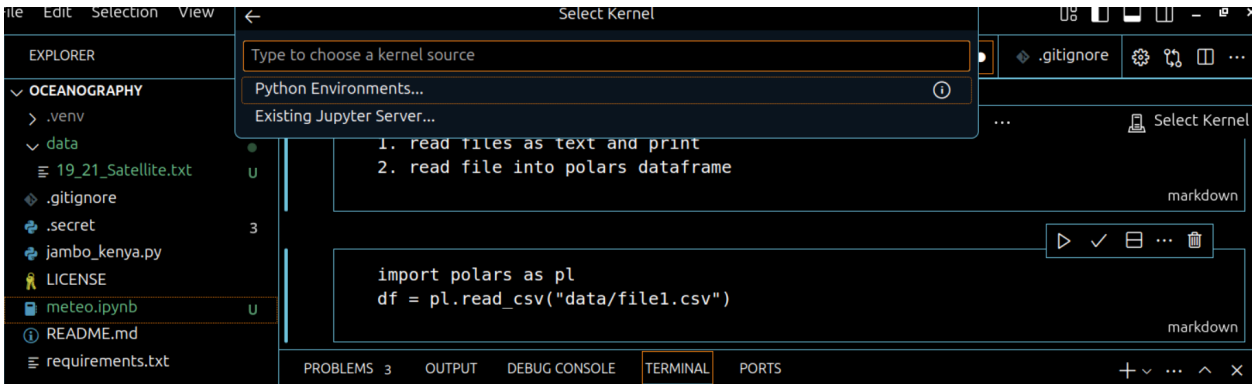
OPENING WITH JUPYTER NOTEBOOK

Save a file with ipynb extensions
meteo.ipynb



Kernel selection

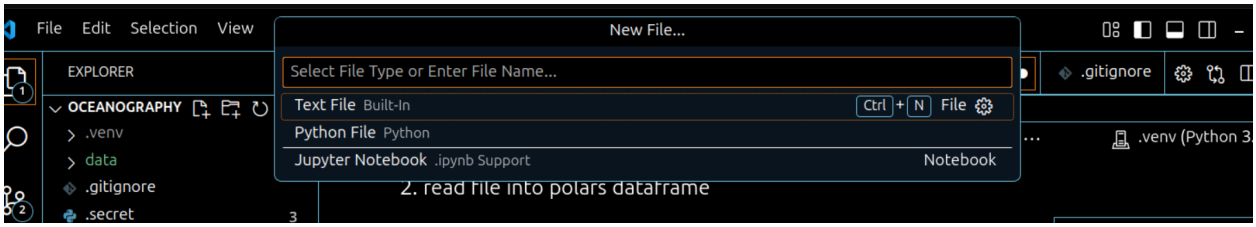
> Python environment



1. Markdown – text
2. Code – writing code

Alternate way Jupyter

Another way to open jupyter notebook
FILE -> NEW FILE ->



CLONING DATA FROM REPO

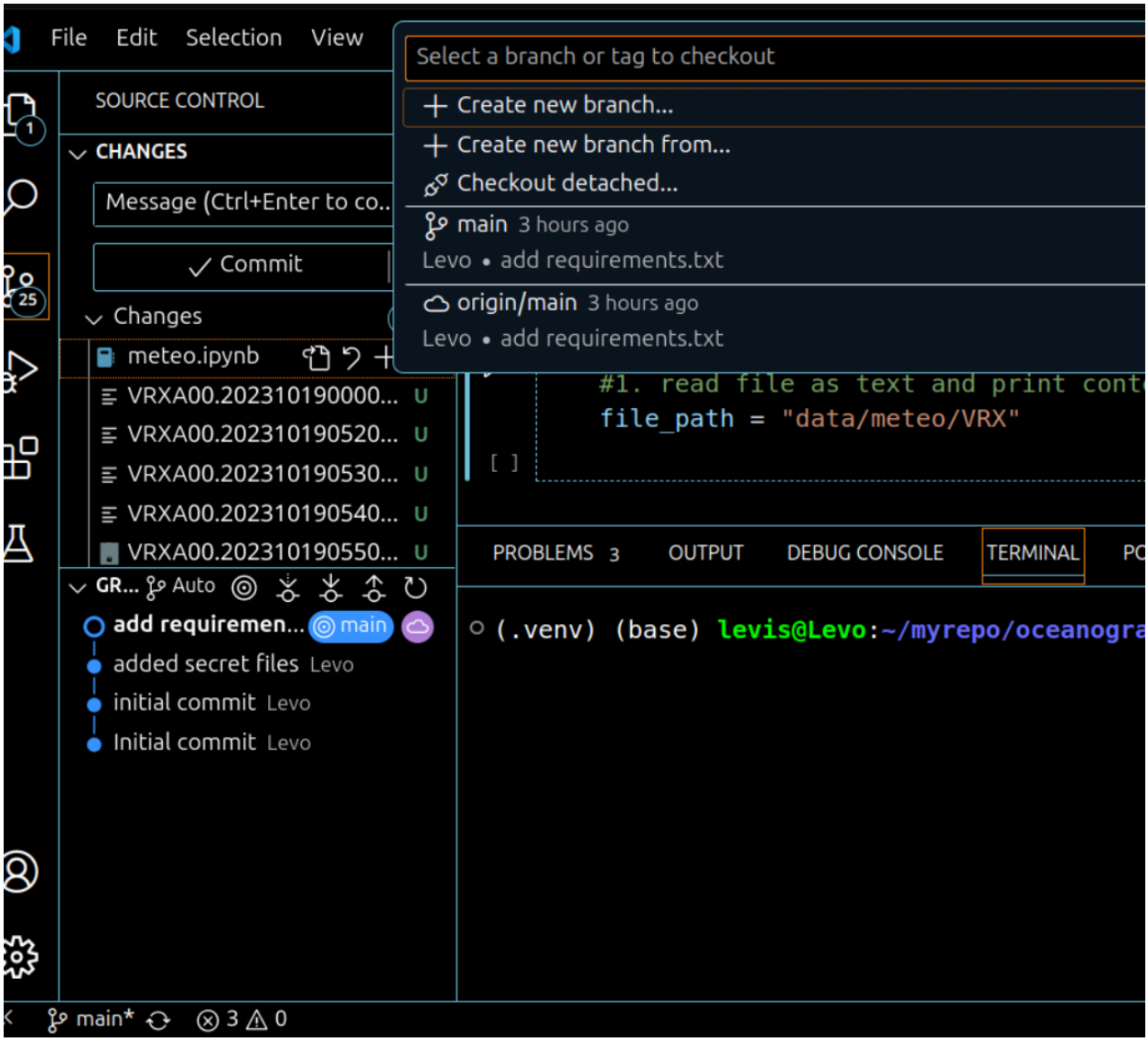
Procedure

1. FILE
2. NEW WINDOW
3. CLONE GIT REPO
4. Clone from github
5. Choose the repo /name/path/

BRANCHING

Procedure to branch a collaborators work

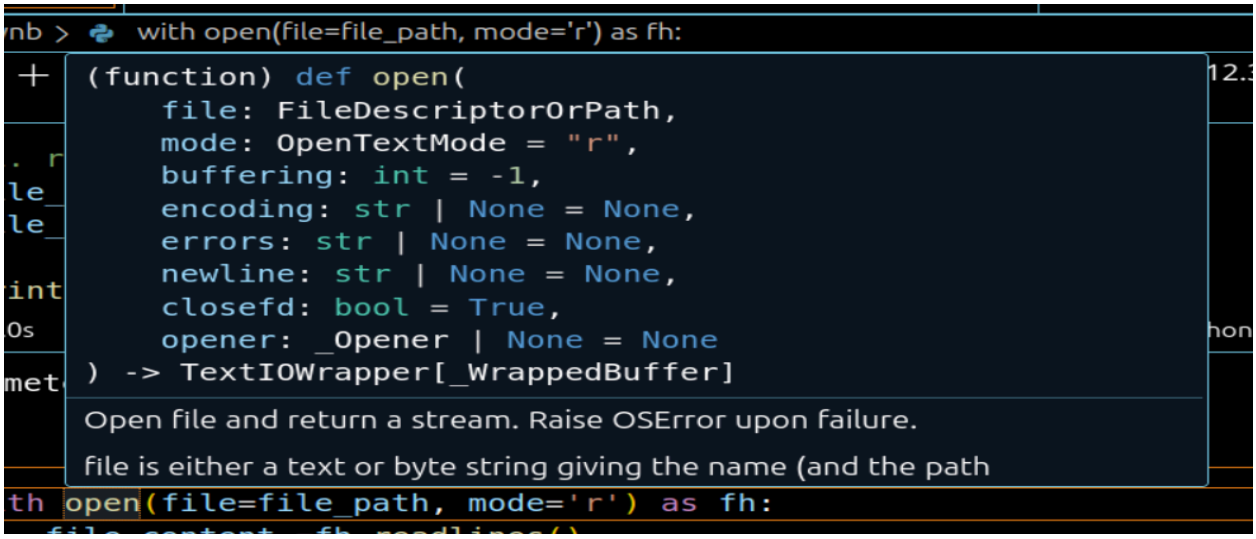
1. SELECT main branch bottom left
2. Create new branch
3. Branch name



PYTHON FUNCTIONS

Open ()

Read file as text and print content



Encoding options: utf8, ascii
Mode: “r” read

Help ()

For more information use the help function
> help(open)

Example code on printing the header

Exercise

1. Extract the header element for temperature(tre200s0)
2. Extract the corresponding temperature reading
3. Print the result as follows: tre200s

Solution

```
temperature = header_list[2]
print(temperature)
values = file_content[4]
values = values.split()*
print(values)
temperature_value = values[2]
temperature_value = float(temperature_value)
print(int(temperature_value))
```

33] ✓ 0.0s Python

```
tre200s0
['187', '202310190530', '007.3', '098.6', '0662.9', '003.4', '329.6', '00000.00', '006.8', '100.0', '0662.7', '00
7
```

```
print(temperature, " : ", (temperature_value))
```

29] ✓ 0.0s Pytho

```
tre200s0  :  7.3
```

Evening Session

DATA TYPES

1. FLOAT → float()
2. INTEGER → int()

Loop Functions

Creating

Loop are important if you have repetitive task

- For loop

Example 1

```
a = 1
for i in range(10):
    a = a + i
    if a > 20:
        break
    print(i,":", a)
```

```
35]   ✓ 0.0s

0 : 1
1 : 2
2 : 4
3 : 7
4 : 11
5 : 16
```

Example 2

```
import os
path = "data/meteo"
files = os.listdir(path)
datetimes = list()
temperatures = list()
for file in files:
    if ".zip" in file:
        continue
    else:
        #print(file)
        datetime = file.replace("VRXA00.","")
        datetimes.append(datetime)
        with open(file=os.path.join(path,file)) as fh:
            file_content =fh.readlines()

            temp = file_content[4].split()[2]

            temperatures.append(temp)

print(temperatures)
print(datetimes)
```

```
✓ 0.0s
['008.0', '007.0', '007.8', '008.5', '008.1', '/', '/', '008.6', '/', '008.6', '005.1', '/', '007.3']
['202310190630', '202310190540', '202310190600', '202310190650', '202310190640', '202310190520', '202401111540', '202310190610
```

The order is not preserved but the match should be preseved