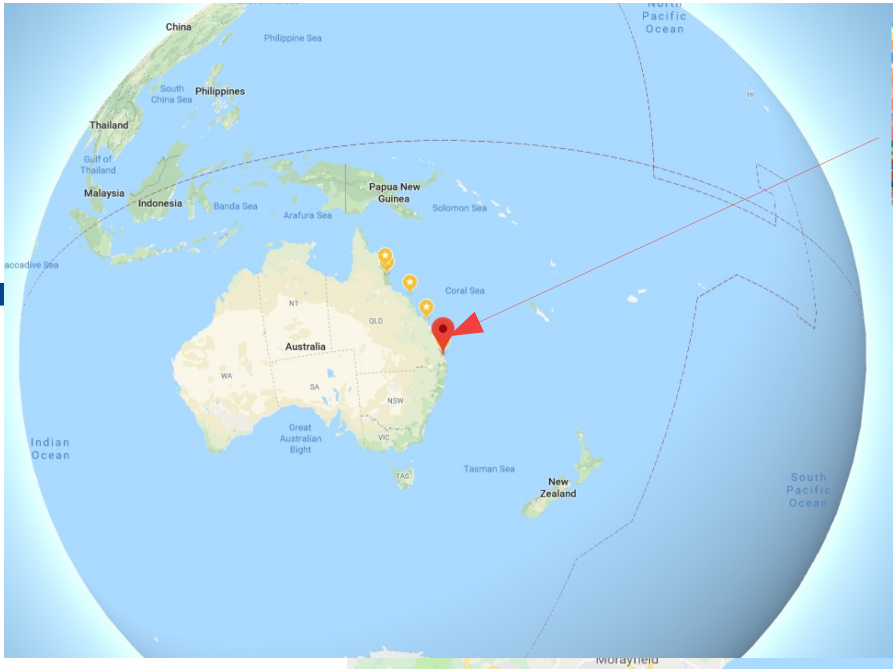
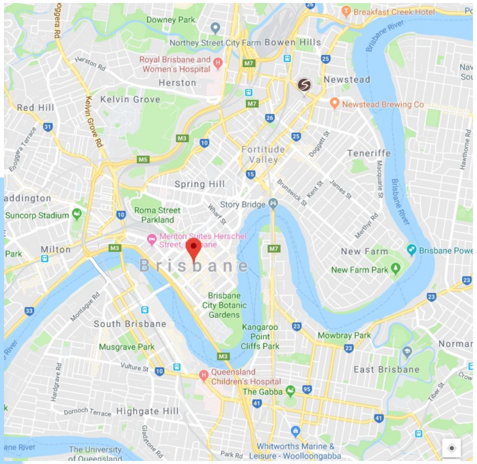
The University of Queensland
School of Earth and Environmental Sciences

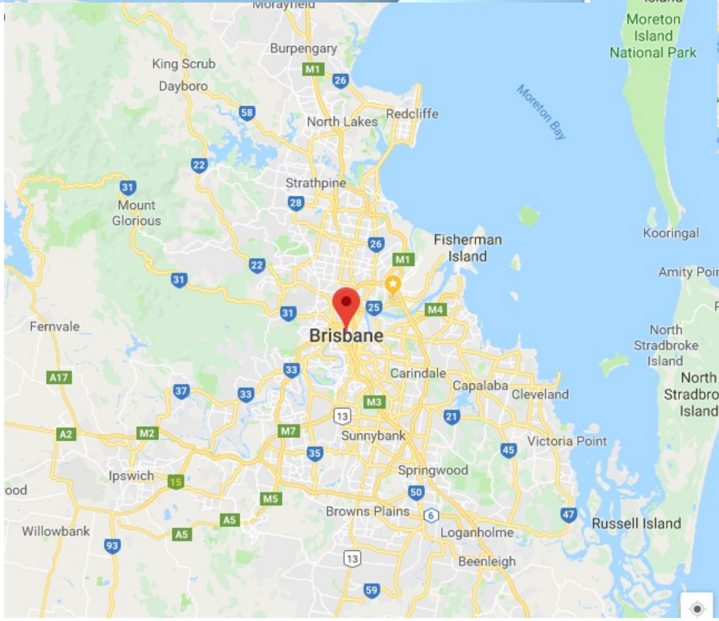
Geophysical Imaging with Finite Elements and Python

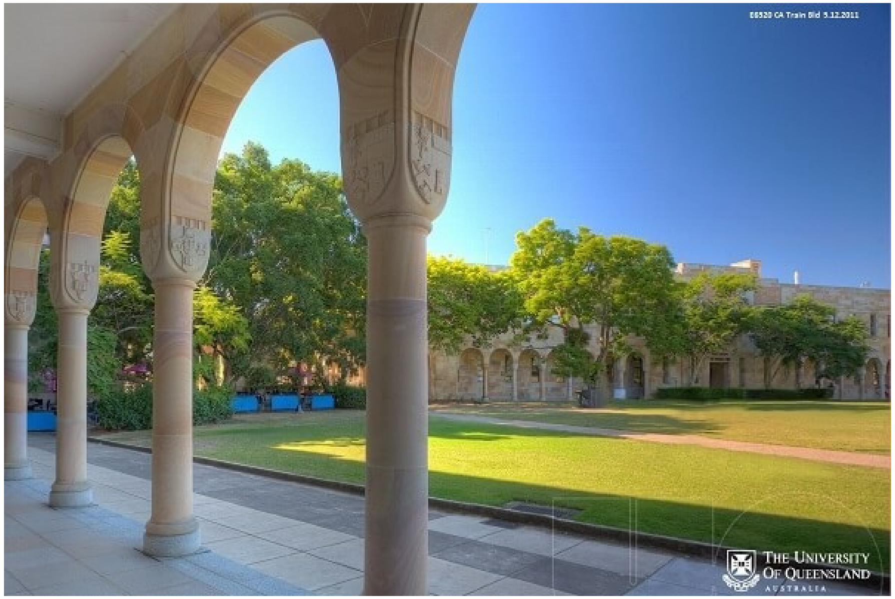
Lutz Gross
E-mail : l.gross@uq.edu.au



Where is Home?






E6520 CA Train Bld 5.12.2011
The University of Queensland Australia

Course Outline

- AIM: Hands-on introduction to
  - Python
  - Finite element method and application in geophysics
  - Concepts of inversion
- Content:

Week 1 (3/11): Introduction to python (incl. practical with Jupyter notebooks)

Week 2 (10/11): More on python programming (incl. practical with Jupyter notebooks)

Week 3 (17/11): Introduction to the Finite Element Method (FEM)

Week 4 (24/11): FEM modeling in geophysics using esys-escript in python

Week 5 (1/12): Parameter Fitting (incl. practical with Jupyter notebooks)

Week 6 (8/12): The Concept of Inversion

Week 7 (15/2): FEM-based Inversion in Geophysics

Week 8 (22/12): Advanced Inversion Techniques

We use Open Source software

- This is software with source code that anyone can inspect, modify, and enhance
  - But watch out for the license: Apache, GPL, etc
- Developments are carried by communities
  - Need help: https://www.python.org/about/help/
  - Get help and help others!
- All documentation is on-line:
  - Not happy? Fix it!

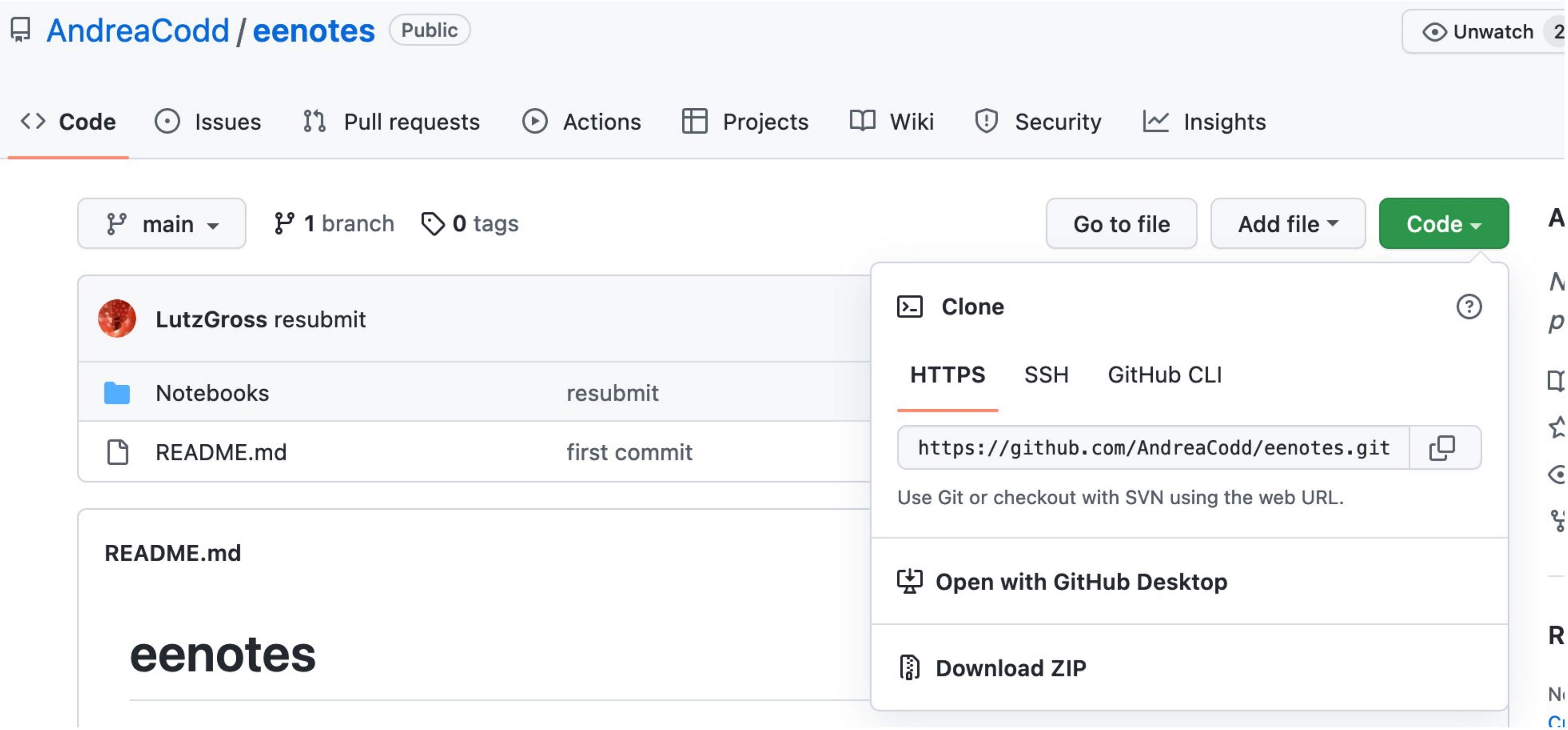
Software for the Course

- What software do I need for the course?
  - Python3
  - Arrays: Numpy
  - Visualization: Matplotlib
  - numerical algorithms: Scipy
  - FEM solver: esys-escript
- Jupyter notebook
  - Edit & run scripts via Web-based server
  - Integrating text & code

Beyond Jupyter Notebook

- Integrated Development Environment (IDE)
  - PyCharm
  - PyDev with Eclipse
  - Spyder IDE
  - Python IDLE
  - etc

Course Material on GitHub



AndreaCodd / eenotes Public

Unwatch 2

LutzGross resubmit

Notebooks	resubmit
README.md	first commit

README.md

eenotes

Clone

https://github.com/AndreaCodd/eenotes.git

Use Git or checkout with SVN using the web URL.

Software Access

A) Python distribution anaconda

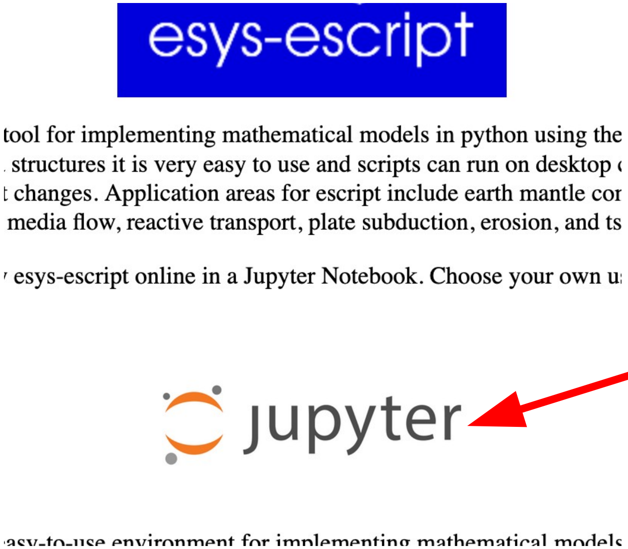
- Install via https://www.anaconda.com/
- Includes many python modules
- Limited esys-escript support :-(

B) install a web-server

- Via a Docker virtual machine
- See notes distributed

C) use cloud server

- Visit https://esys-escript.github.io/
- Click on Jupyter logo



esys-escript

tool for implementing mathematical models in python using the
structures it is very easy to use and scripts can run on desktop
t changes. Application areas for escript include earth mantle cor
media flow, reactive transport, plate subduction, erosion, and ts

esys-escript online in a Jupyter Notebook. Choose your own u

asy-to-use environment for implementing mathematical models

Click here

Create An Account

Sign in

Warning: JupyterHub seems to be served over an unsecured HTTP connection. We strongly recommend enabling HTTPS for JupyterHub.

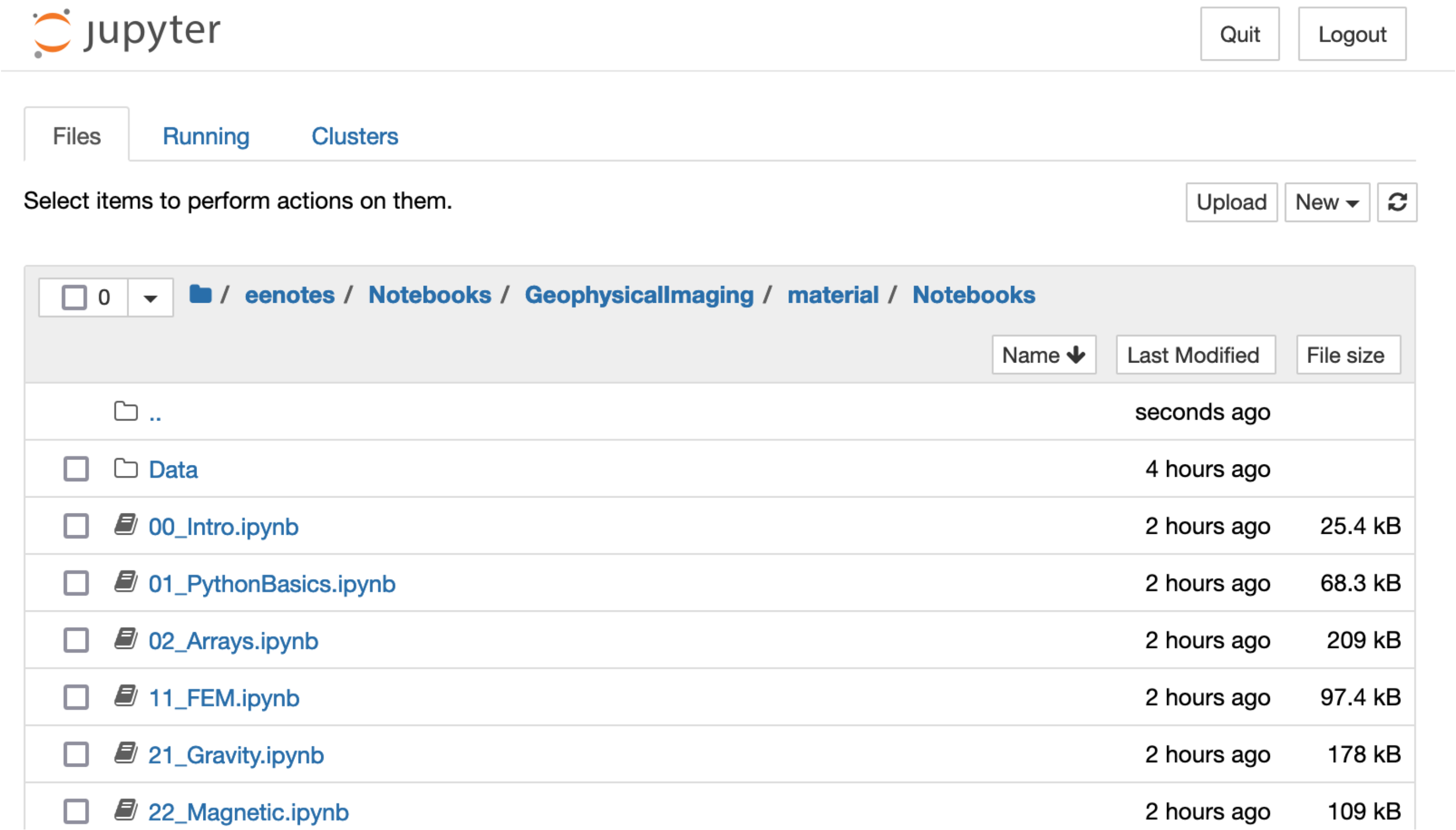
Username:

Password:

- Don’t forget your password!
- Download your data at one point ...

Navigate to Course Material

With preloaded course material

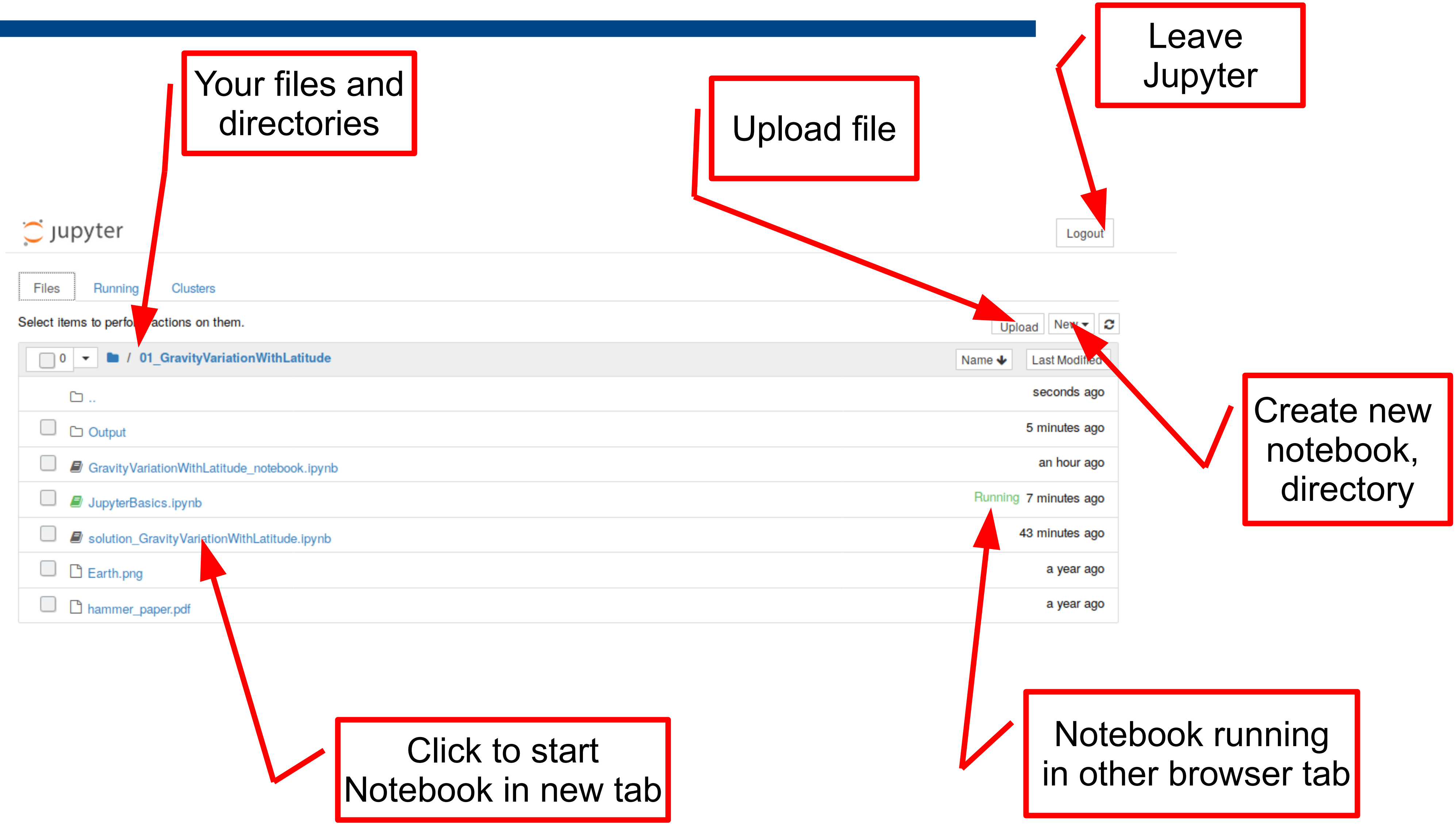


Select items to perform actions on them.

□ 0 / eenotes / Notebooks / GeophysicalImaging / material / Notebooks

Name	Last Modified	File size
..	seconds ago	
□ Data	4 hours ago	
□ 00_Intro.ipynb	2 hours ago	25.4 kB
□ 01_PythonBasics.ipynb	2 hours ago	68.3 kB
□ 02_Arrays.ipynb	2 hours ago	209 kB
□ 11_FEM.ipynb	2 hours ago	97.4 kB
□ 21_Gravity.ipynb	2 hours ago	178 kB
□ 22_Magnetic.ipynb	2 hours ago	109 kB

After login



Your files and directories

Upload file

Leave Jupyter

Create new notebook, directory

Click to start Notebook in new tab

Notebook running in other browser tab

Select items to perform actions on them.

□ 0 ▾ / 01_GravityVariationWithLatitude

..	seconds ago
□ Output	5 minutes ago
□ GravityVariationWithLatitude_notebook.ipynb	an hour ago
□ JupyterBasics.ipynb	Running 7 minutes ago
□ solution_GravityVariationWithLatitude.ipynb	43 minutes ago
□ Earth.png	a year ago
□ hammer_paper.pdf	a year ago

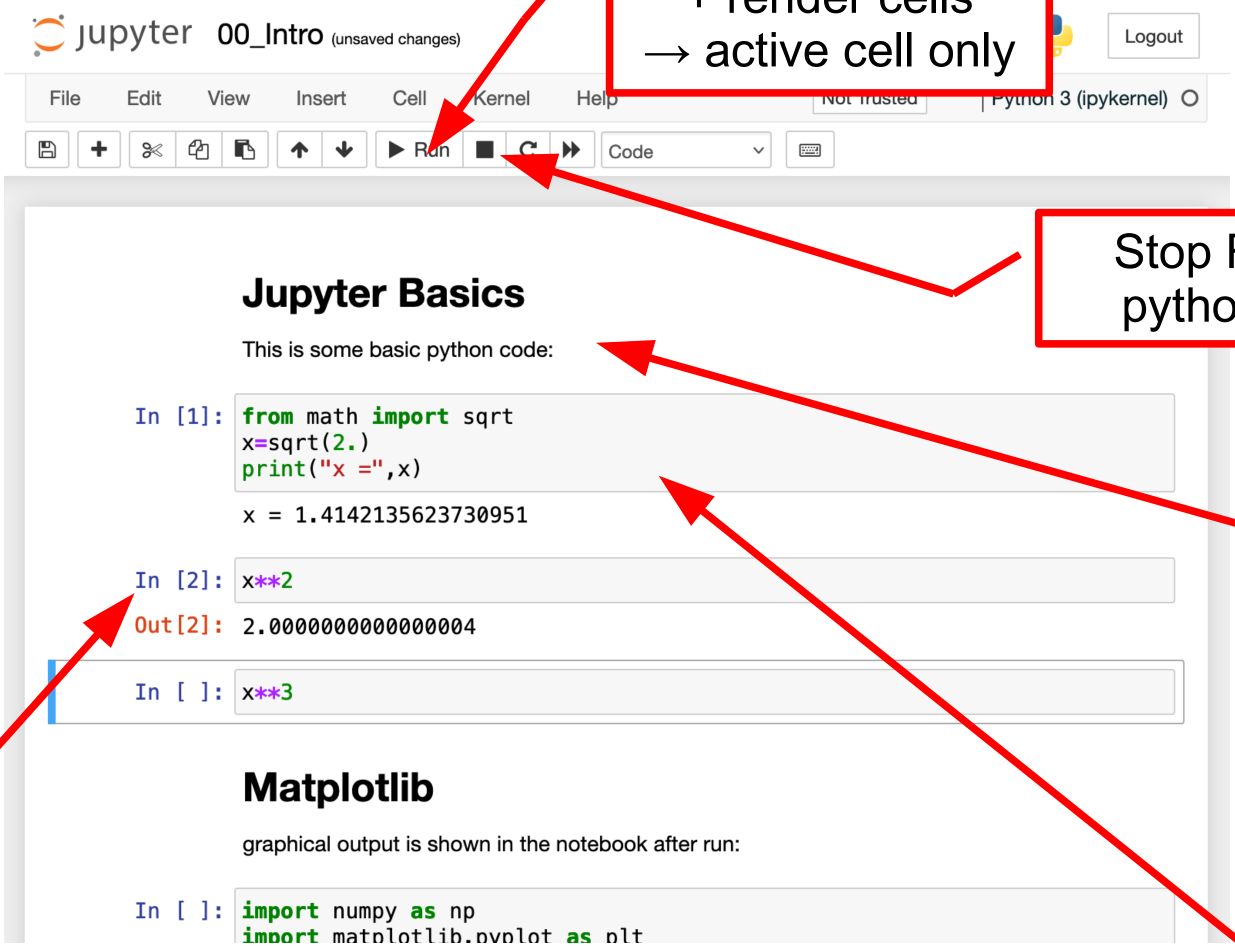
Jupyter notebook

- web application → use your favorite browser
  - contain live python code
  - includes narrative text
  - contains equations
  - includes visualizations
  - to create and share documents
- See https://jupyter.org/

A jupyter notebook

- is two things:
    - A document containing text and python code
        - Extension: ipynb
    - A session
        - to render the text
        - to run the code/sections of the code

notebook page



Jupyter Basics

This is some basic python code:

```
In [1]: from math import sqrt
        x=sqrt(2.)
        print("x =",x)

        x = 1.4142135623730951

In [2]: x**2

Out[2]: 2.0000000000000004

In [ ]: x**3
```

Matplotlib

graphical output is shown in the notebook after run:

```
In [ ]: import numpy as np
        import matplotlib.pyplot as plt
```

Run python code
+ render cells
→ active cell only

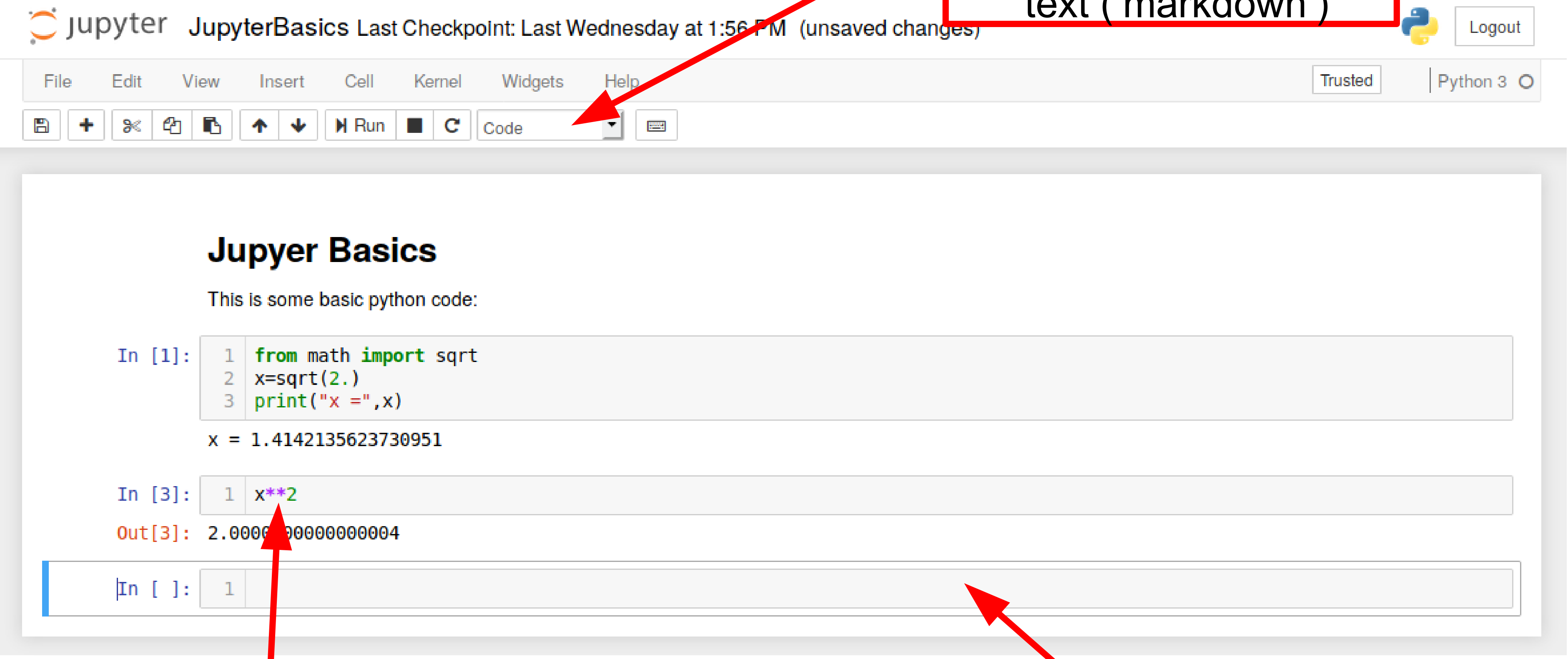
Stop Running
python code

Cell with
text

Cell with
python code

active cell with code

Cells after ‘run’


jupyter
JupyterBasics Last Checkpoint: Last Wednesday at 1:56 PM (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
Run
Code

Jupyer Basics

This is some basic python code:

In [1]:
```
from math import sqrt
x=sqrt(2.)
print("x =",x)
```

```
x = 1.4142135623730951
```

In [3]:
```
x**2
```

Out[3]:
```
2.0000000000000004
```

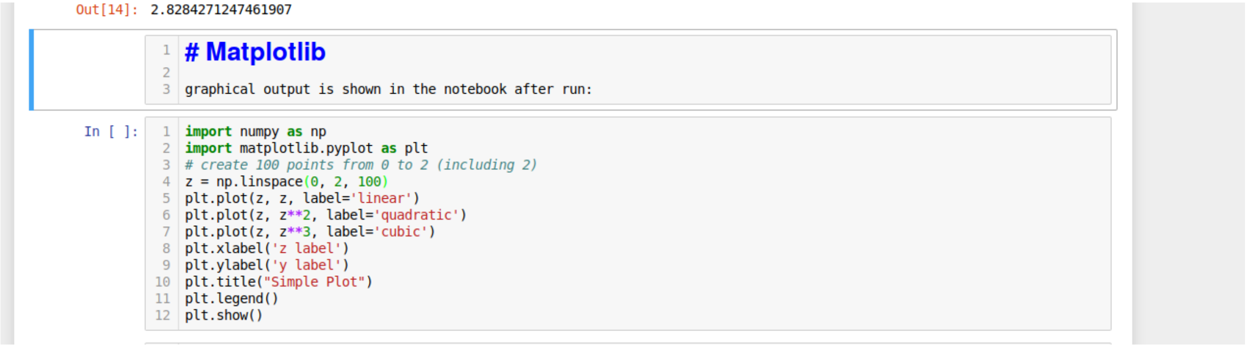
In [ ]:

Change cell type
Code or
text (‘markdown’)

All cells treated as
single python code

Cell code cell
added

Matplotlib output is integrated

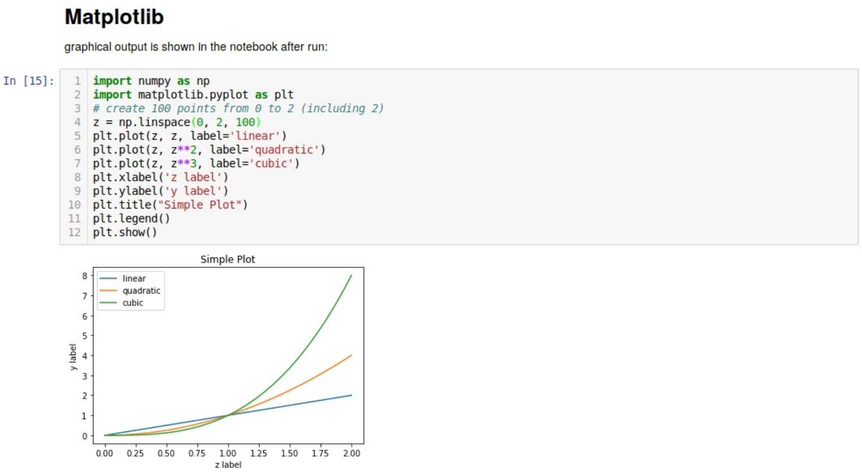


```
Out[14]: 2.8284271247461907
```

```
# Matplotlib

graphical output is shown in the notebook after run:
```

```
In [ ]:
import numpy as np
import matplotlib.pyplot as plt
# create 100 points from 0 to 2 (including 2)
z = np.linspace(0, 2, 100)
plt.plot(z, z, label='linear')
plt.plot(z, z**2, label='quadratic')
plt.plot(z, z**3, label='cubic')
plt.xlabel('z label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
plt.show()
```

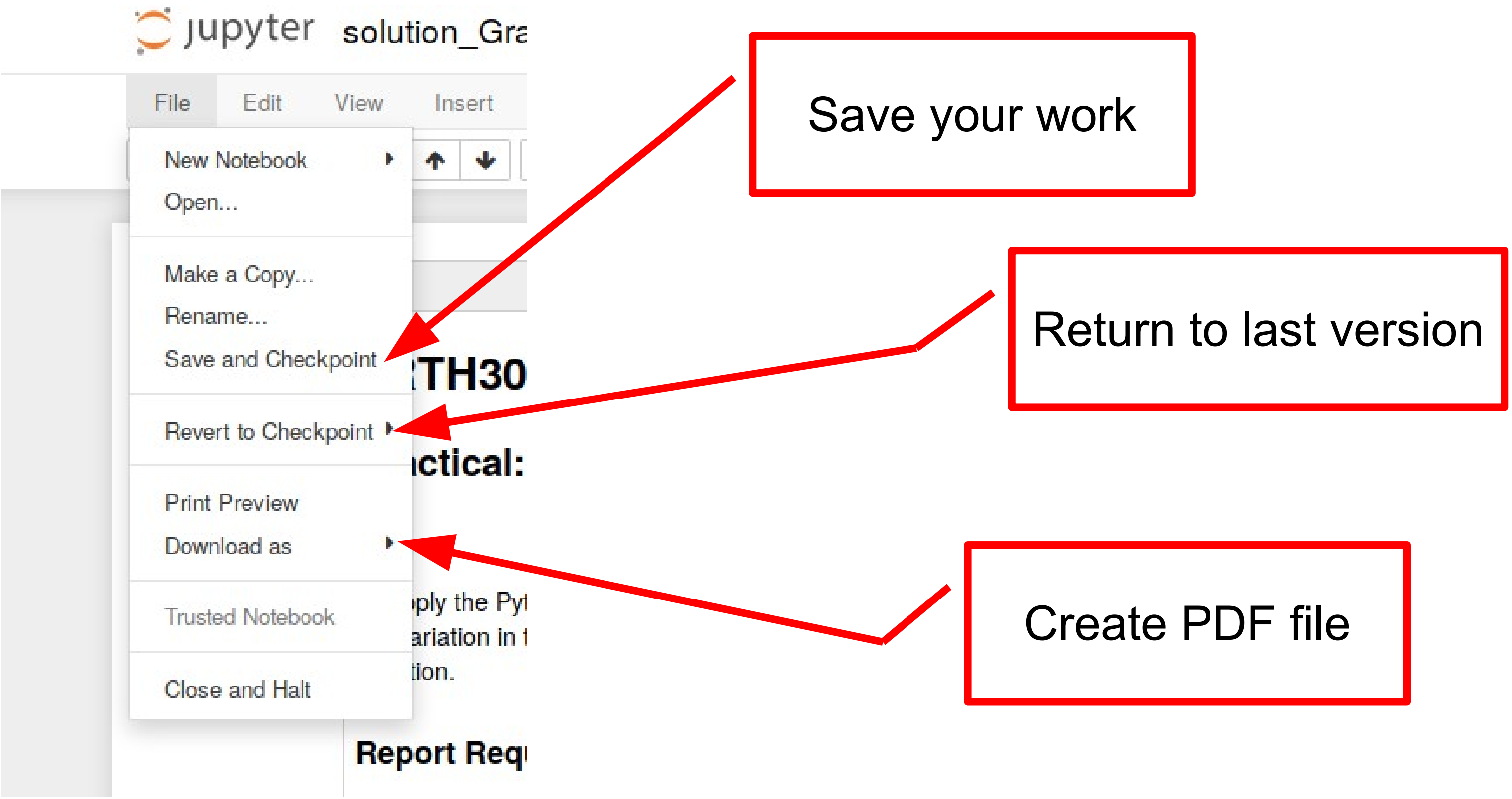


Matplotlib

graphical output is shown in the notebook after run:

```
In [15]:
import numpy as np
import matplotlib.pyplot as plt
# create 100 points from 0 to 2 (including 2)
z = np.linspace(0, 2, 100)
plt.plot(z, z, label='linear')
plt.plot(z, z**2, label='quadratic')
plt.plot(z, z**3, label='cubic')
plt.xlabel('z label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
plt.show()
```

File Menu

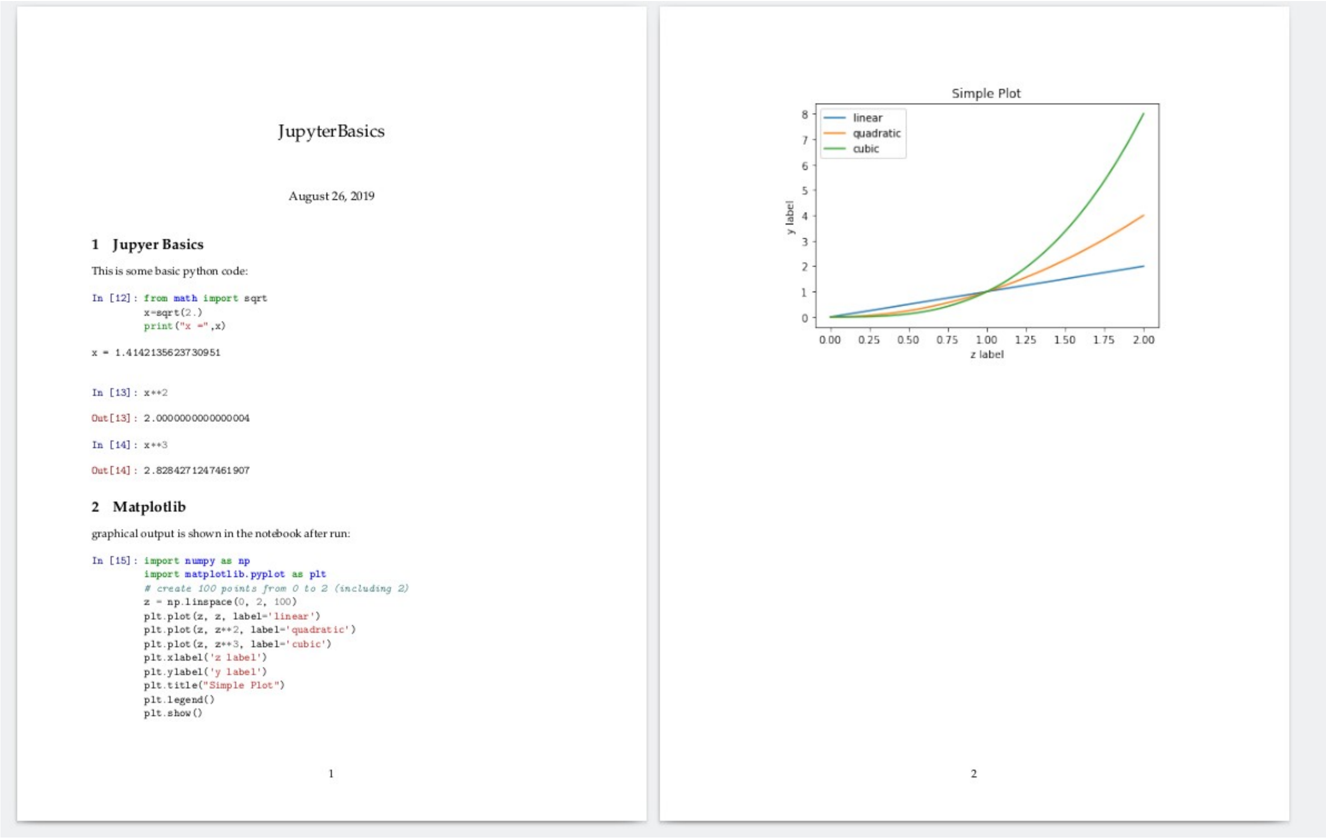


Save your work

Return to last version

Create PDF file

Example report



JupyterBasics

August 26, 2019

1 Jupyer Basics

This is some basic python code:

```
In [12]: from math import sqrt
         x=sqrt(2.)
         print("x =",x)

x = 1.4142135623730951


In [13]: x**2

Out[13]: 2.0000000000000004

In [14]: x**3

Out[14]: 2.8284271247461907
```

2 Matplotlib

graphical output is shown in the notebook after run:

```
In [15]: import numpy as np
         import matplotlib.pyplot as plt
         # create 100 points from 0 to 2 (including 2)
         z = np.linspace(0, 2, 100)
         plt.plot(z, z, label='linear')
         plt.plot(z, z**2, label='quadratic')
         plt.plot(z, z**3, label='cubic')
         plt.xlabel('z label')
         plt.ylabel('y label')
         plt.title("Simple Plot")
         plt.legend()
         plt.show()
```

1

2

And more

- ‘Edit’ → copy & past & split & merge
- ‘Insert’ → add new cell
- ‘Run’ → run all cells, run cells above/below, …
- For markdown text see for instance:
  - https://www.datacamp.com/community/tutorials/markdown-in-jupyter-notebook