AGQ – Advanced Python3, Git and Numerical Analysis

Robert Currie

Intro

Who am I?
Physicist turned Computing *Researcher/Admin/Developer/*…

Did my PhD on LHCb Particle Physics experiment at Edinburgh in 2009-2014.

During my PhD I spent a lot of time developing and optimizing C++ software dedicated to “fitting” (optimizing a PDF to a given set of data). Under the hood this uses a lot of the same technologies as training Machine Learning models.

After my PhD I spent 2 years as a Python software developer in ICL. Since I returned to Edinburgh, I have gained experience in various computing fields including distributed storage systems, full-stack web development, optimizing and debugging at scale and working with large-scale batch computing.

My AGQ Lectures

- **Lecture 1: Embracing Python3 & Git**

  Intro, Intermediate **Python3**, **Git**

  Getting Setup with **Conda**, Basics of **Numpy** & **Pandas**

- **Lecture 2: Machine Learning from the ground up**

  Building a simple **DNN** using **NumPy**, then the same again using **PyTorch**.

  Constructing a Simple **Classifier** using **PyTorch**.

- **Lecture 3: More complex Machine Learning models**

  Building more **complex graphs** using **PyTorch**.

  Building an **AutoEncoder** using **PyTorch**.

- **Lecture 4: Modern Machine Learning Concepts**

  DNN, **CNN**, **Attention** and model **Precision**.

  Building a **1-bit precision Classifier** using PyTorch

Lecture 1: Embracing Python3 & Git

Some *advanced* Python concepts.

If you’re familiar with classes and functors then this is brushing up on the basics.

Python Virtual Environments

Using them, setting them up, throwing them away.

Machine Learning software and Numerical Minimization using Python

Using Git for version control

The basics of version control management with git, fork, branch, PR.

Modern Computing

This Photo by Unknown Author is licensed under CC BY-SA

- Computers can be messy and difficult.

- Unfortunately, Moore’s Law and Denard Scaling have now slowed, and you don’t get ~2x performance at ~1.5x power efficiency every 18months.

- This means a good laptop in 2010 was ½ as good as one in 2013,

  but; a good laptop in 2024 will be ~the-same as one in 2026…

- However: **GPUs, FPGAs, NPUs** and other weird hardware is now seeing >2x performance improvements every 18months due to the use pipelines and other gains.

Python3 – Why use it?

- One of the biggest problems with accessing the hardware in your computer is:

  using complex hardware requires complex domain knowledge.

- Python3 ecosystem is more focussed on:
  "make it easy to get stuff done"

- OK, so if I simply example multiply 2 (500x500) arrays of double precision numbers.

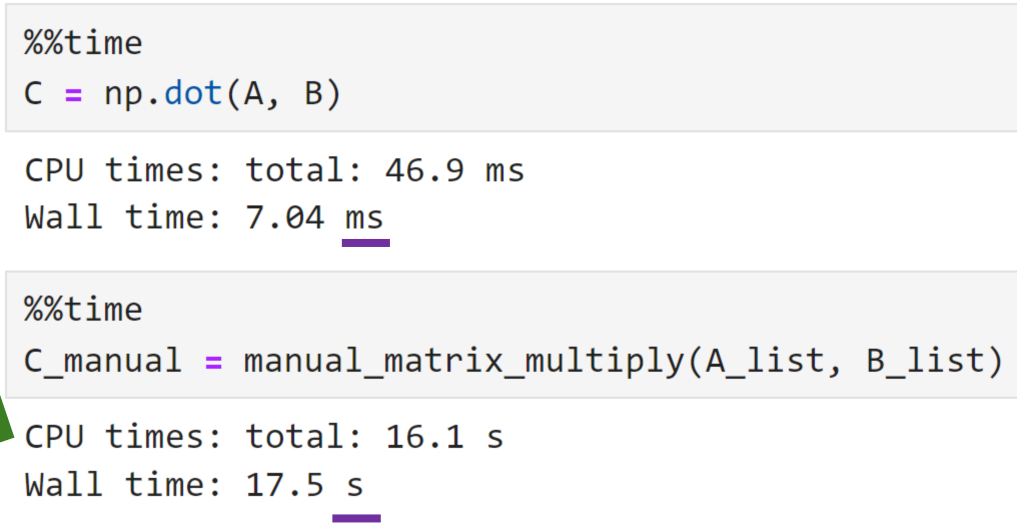
- How long do you think that will take?

Python3 – Why use it?

- Using **SIMD** on an Intel CPU requires *~100 lines of low-level C code* which is compiled to use processor specific instructions.

- This can do many multiplications, in parallel, per-clock-cycle, on a multi-threaded CPU. (*This means higher throughput*)

- Doing this **manually**,
  for 500x500:
  take 2 doubles,
  multiply
  store
  Takes a very long time…



```
%%time
C = np.dot(A, B)

CPU times: total: 46.9 ms
Wall time: 7.04 ms

%%time
C_manual = manual_matrix_multiply(A_list, B_list)

CPU times: total: 16.1 s
Wall time: 17.5 s
```

Python3 – Why use it?

- Very large growing scientific ecosystem with many complex tools and libraries already written/debugged/supported by a community.
- IMO, most good Python code tends to be very readable.
  The language has a low barrier to entry.

- Lots of tooling being developed around the idea of:
  "**performance meta-programming**".
- Simple concept:

  "*let the computing experts do the optimization*", but,
  "**keep the code readable for scientists**"

Python3 – General advice

“the rule of 7”

- I’m a strong advocate of, *and someone who regularly breaks*,

  Try to keep all methods 7 lines long,
  Never use more than 7 arguments,
  Try to stick to 7 methods in a class

  Avoid more than 7 levels of indentation,
  Try not to give 7 examples of the rule of 7,

- Python allows for the use of global variables.

  Unless you’re an experienced developer ***avoid using them.***
  They’re basically the programming equivalent of a hand-grenade with a missing pin waiting to go off!

Python3 – General advice

Python3;

is a great dynamically-typed polymorphic interperated scripting language

Also:

Python3;

handles types in a messy way, has some strange performance features and can be very difficult to debug/optimize.

Python3 – Classes

- **Classes** are a programming concept common to many languages.

- **Classes** are defined as being the blueprints or instructions for how to construct/assemble an **object**.

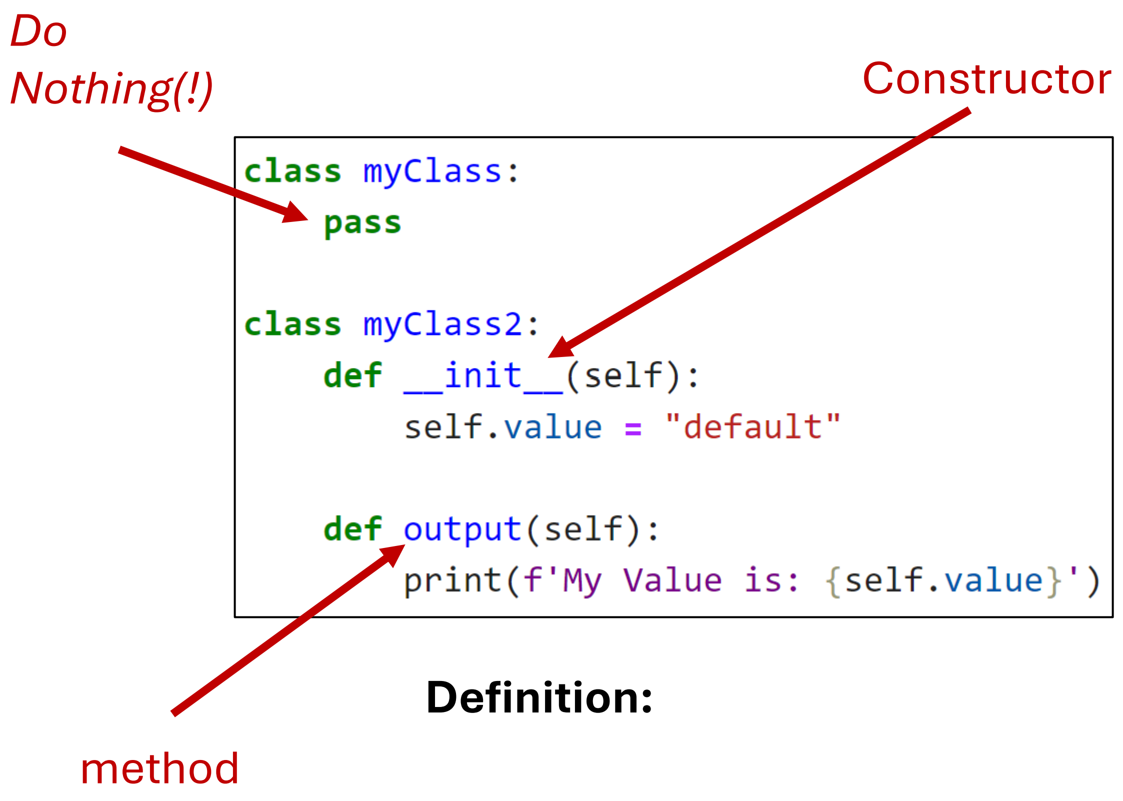
- **Classes** can have methods and attributes.

  Methods are typically callable functions with or without arguments.
  Attributes are values or objects which can be externally accessed.

- In Python there is no real concept of "*private*" or completely hidden methods/objects from external access.

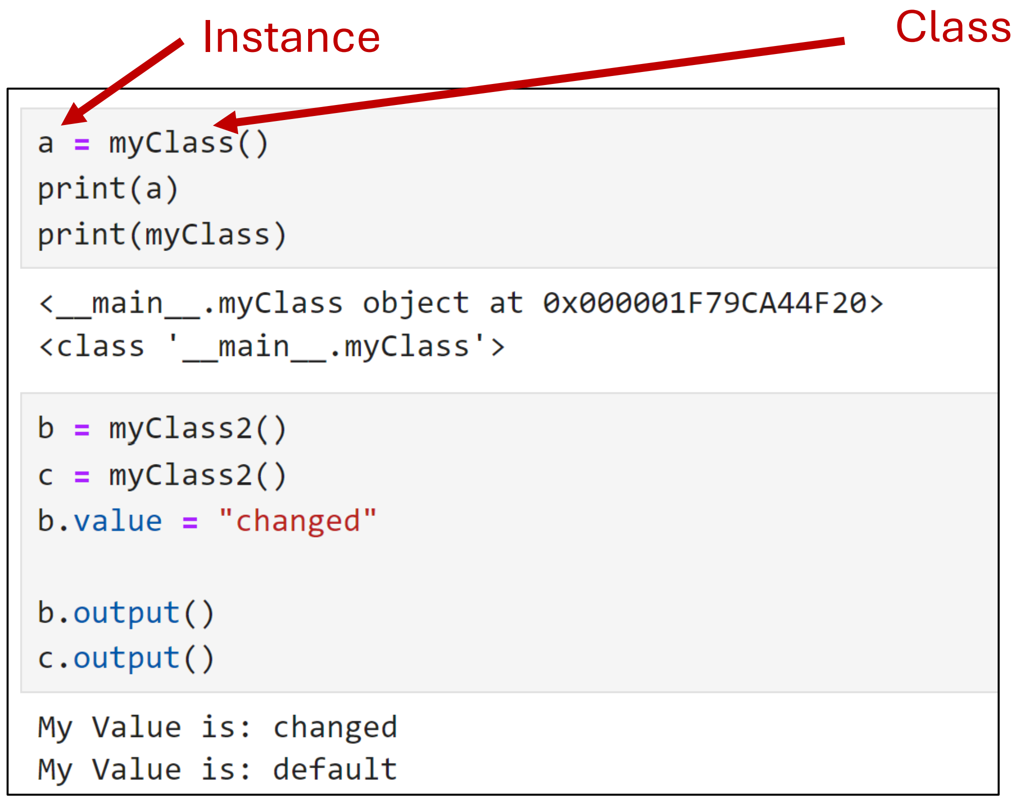
Python3 – Classes

Classes are defined as being the blueprints or instructions for how to construct/assemble an object.



```python
class myClass:
    pass

class myClass2:
    def __init__(self):
        self.value = "default"

    def output(self):
        print(f'My Value is: {self.value}')
```

Definition:



```python
a = myClass()
print(a)
print(myClass)
```

```
<__main__.myClass object at 0x000001F79CA44F20>
<class '__main__.myClass'>
```

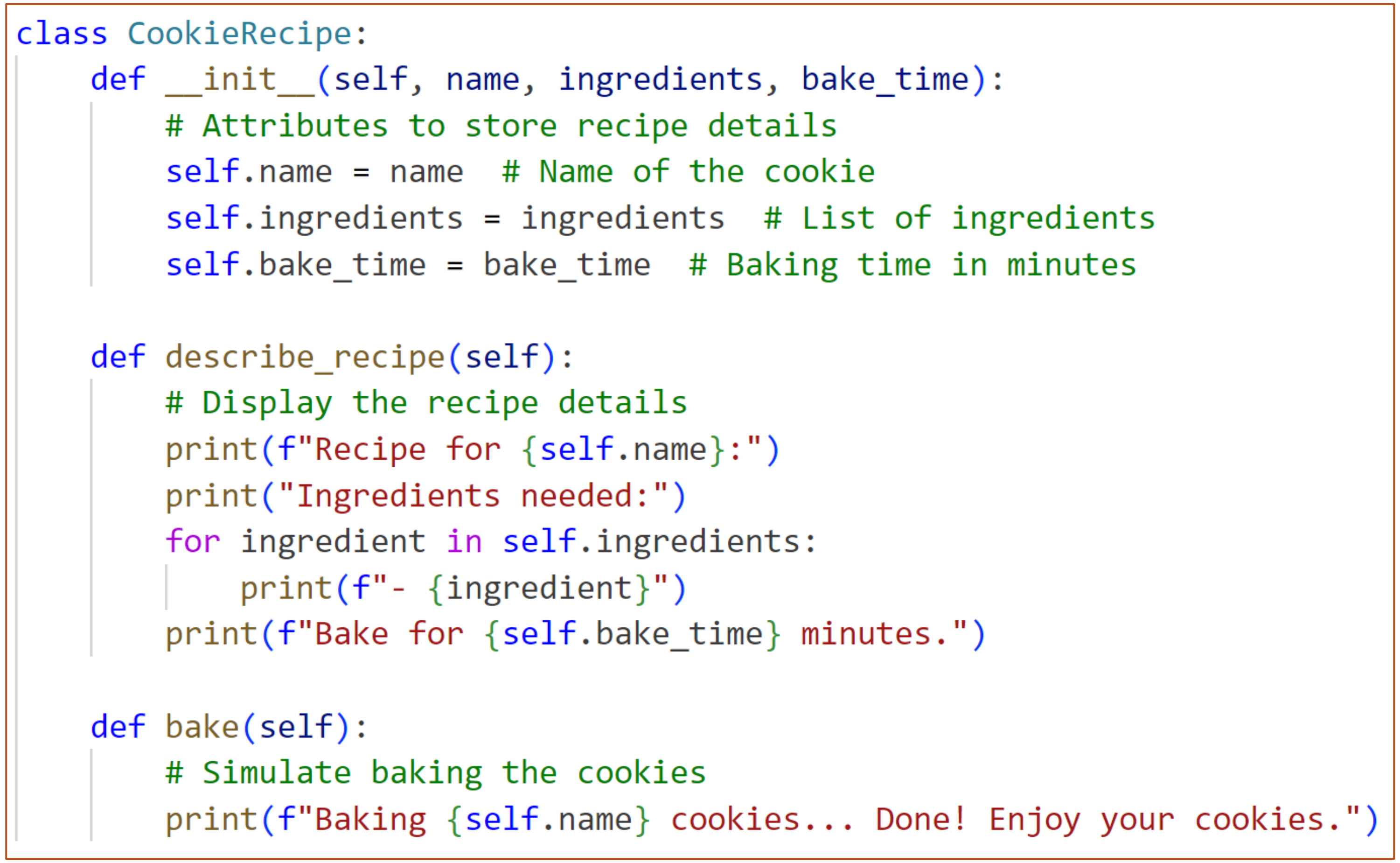
```python
b = myClass2()
c = myClass2()
b.value = "changed"

b.output()
c.output()
```

```
My Value is: changed
My Value is: default
```

Use:

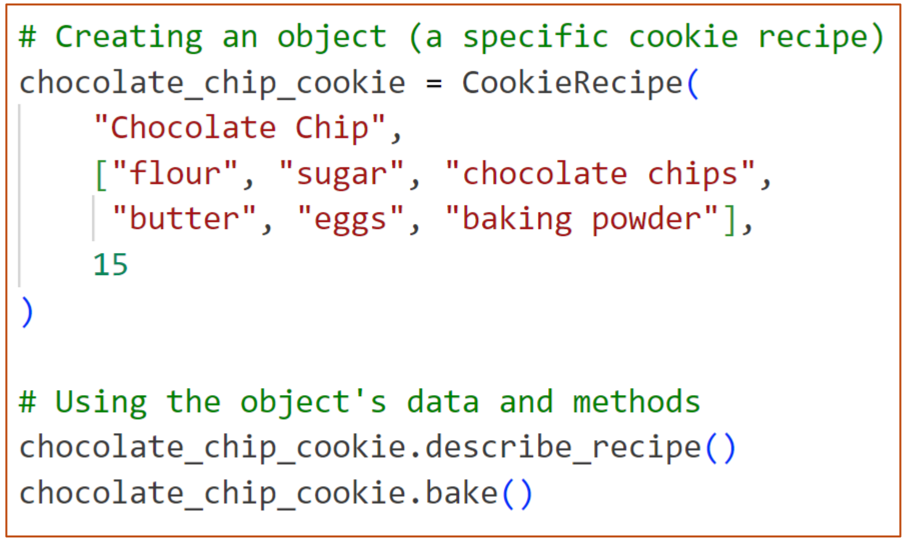
Python3 – Classes

Cookie Recipe (Class Definition)



```python
class CookieRecipe:
    def __init__(self, name, ingredients, bake_time):
        # Attributes to store recipe details
        self.name = name  # Name of the cookie
        self.ingredients = ingredients  # List of ingredients
        self.bake_time = bake_time  # Baking time in minutes

    def describe_recipe(self):
        # Display the recipe details
        print(f"Recipe for {self.name}:")
        print("Ingredients needed:")
        for ingredient in self.ingredients:
            print(f"- {ingredient}")
        print(f"Bake for {self.bake_time} minutes.")

    def bake(self):
        # Simulate baking the cookies
        print(f"Baking {self.name} cookies... Done! Enjoy your cookies.")
```

Baking a “chocolate_chip_cookie”



```python
# Creating an object (a specific cookie recipe)
chocolate_chip_cookie = CookieRecipe(
    "Chocolate Chip",
    ["flour", "sugar", "chocolate chips",
     "butter", "eggs", "baking powder"],
    15
)

# Using the object's data and methods
chocolate_chip_cookie.describe_recipe()
chocolate_chip_cookie.bake()
```

Python3 – Classes – Inheritance

You may also want to re-use an existing class to get something done.

This is useful to save re-writing the same methods over and over...

Argument (Required)

No Argument(s)

```python
class BaseAnimal:
    def __init__(self, name):
        self.name = name

    def make_noise(self):
        raise NotImplementedError("Subclasses must implement make_noise method")
```

base_animal.py

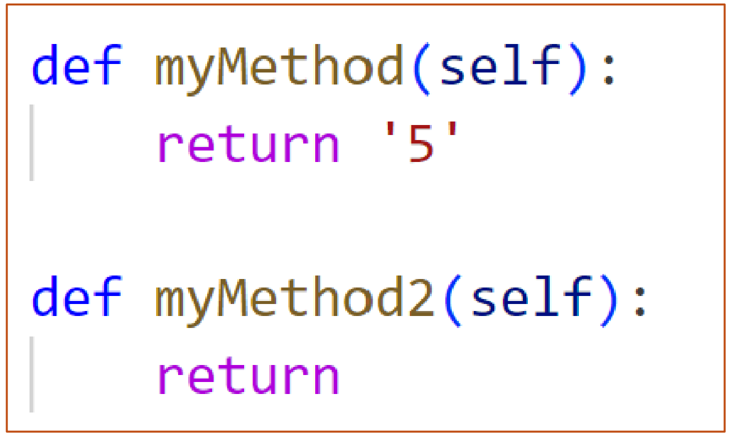
```python
from .base_animal import BaseAnimal

class Chicken(BaseAnimal):
    def make_noise(self):
        return f"{self.name} says Cluck!"
```

chicken.py

Python3 – Classes – Methods

Methods on a class can alter the state of objects within the class itself.

Methods may return a value, i.e. ` str, int, float`

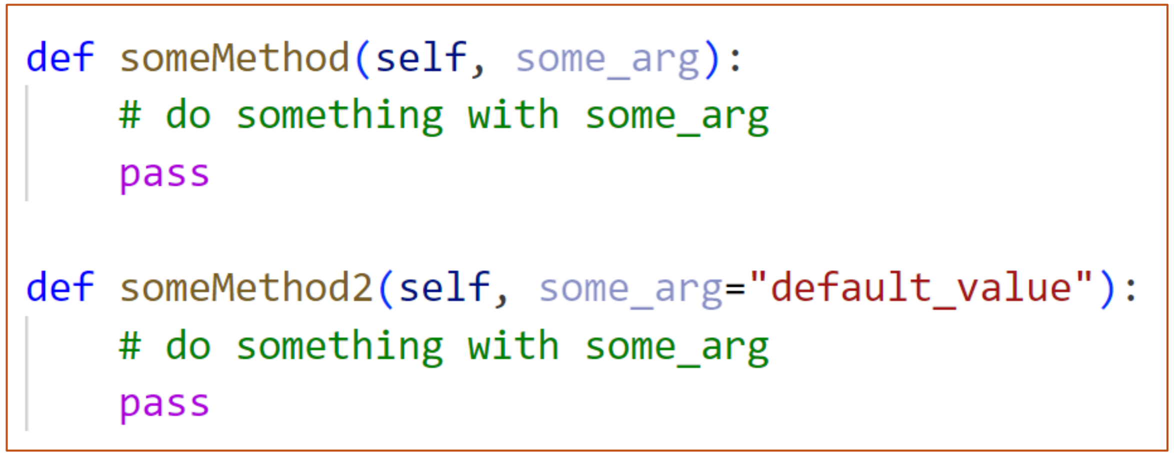


```python
def myMethod(self):
    return '5'

def myMethod2(self):
    return
```

Methods may also not return a value.

Confusingly in Python this is a ` None` object.

Some methods will take arguments:



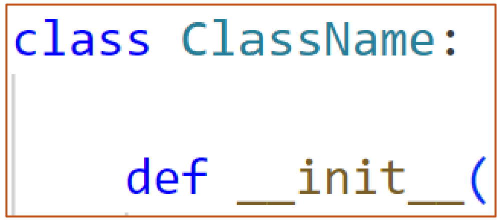
```python
def someMethod(self, some_arg):
    # do something with some_arg
    pass

def someMethod2(self, some_arg="default_value"):
    # do something with some_arg
    pass
```

Some arguments are optional:

Python3 – Classes – Methods 2

There are several semi-private methods in Python classes.

The most common you will come across is:



```
class ClassName:

    def __init__(
```

By construction these are methods which "hide" functionality from a user.

Almost all functionality in Python can be overloaded(!)

Python3 – Classes – Overloading

- Formally this approach to programming is called "*monkey-patching*":

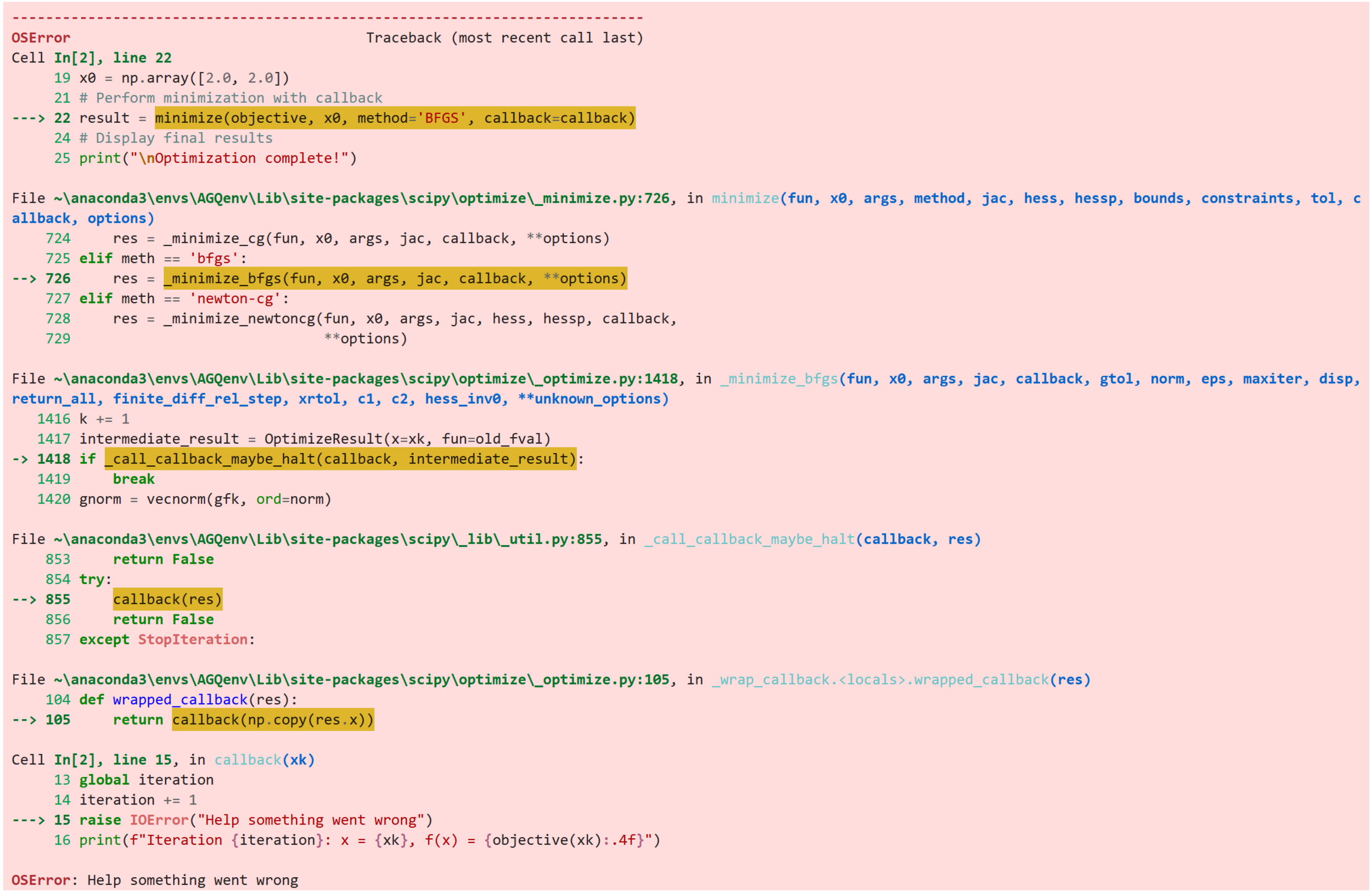
- You can re-define almost anything you want at any time.

- Just be extremely careful…

```python
original_int = int
def int(value):
    if value == 8:
        return original_int(9)
    return original_int(value)

print(int(8))  # Output: 9
```

Python3 – Exception Handling

- Exceptions in a lot of languages can seem scary.
- In Python they’re quite friendly if you know how to use them correctly.
- These are the pieces of code that give horrible long error messages e.g.:



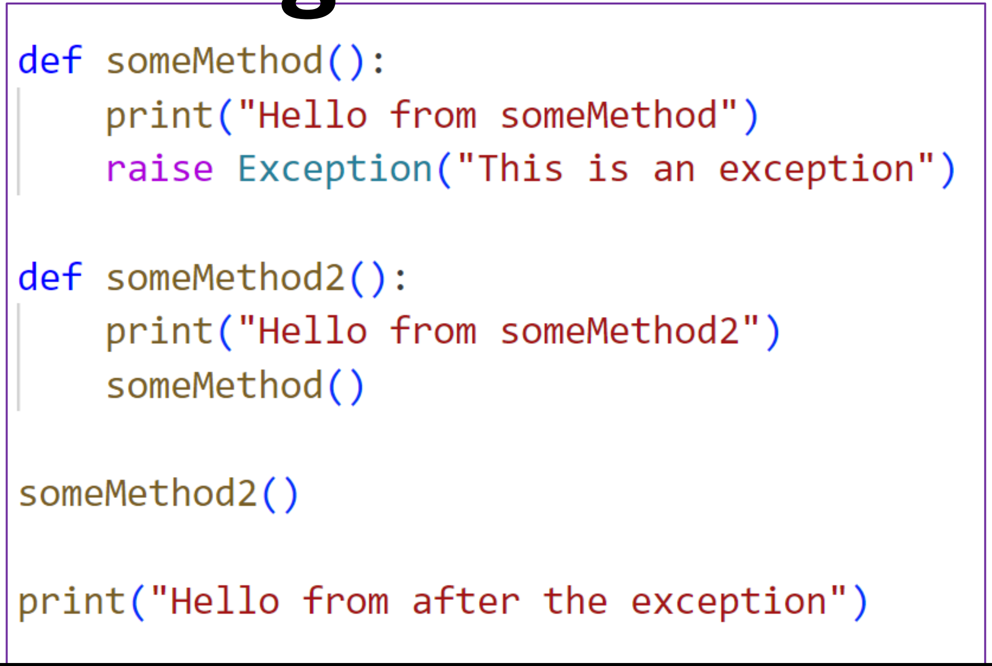
```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
Cell In[2], line 22
     19 x0 = np.array([2.0, 2.0])
     21 # Perform minimization with callback
---> 22 result = minimize(objective, x0, method='BFGS', callback=callback)
     24 # Display final results
     25 print("\nOptimization complete!")

File ~\anaconda3\envs\AGQenv\Lib\site-packages\scipy\optimize\_minimize.py:726, in minimize(fun, x0, args, method, jac, hess, hessp, bounds, constraints, tol, c
allback, options)
    724     res = _minimize_cg(fun, x0, args, jac, callback, **options)
    725 elif meth == 'bfgs':
--> 726     res = _minimize_bfgs(fun, x0, args, jac, callback, **options)
    727 elif meth == 'newton-cg':
    728     res = _minimize_newtoncg(fun, x0, args, jac, hess, hessp, callback,
    729                              **options)

File ~\anaconda3\envs\AGQenv\Lib\site-packages\scipy\optimize\_optimize.py:1418, in _minimize_bfgs(fun, x0, args, jac, callback, gtol, norm, eps, maxiter, disp,
return_all, finite_diff_rel_step, xrtol, c1, c2, hess_inv0, **unknown_options)
   1416 k += 1
   1417 intermediate_result = OptimizeResult(x=xk, fun=old_fval)
-> 1418 if _call_callback_maybe_halt(callback, intermediate_result):
   1419     break
   1420 gnorm = vecnorm(gfk, ord=norm)

File ~\anaconda3\envs\AGQenv\Lib\site-packages\scipy\_lib\_util.py:855, in _call_callback_maybe_halt(callback, res)
    853     return False
    854 try:
--> 855     callback(res)
    856     return False
    857 except StopIteration:

File ~\anaconda3\envs\AGQenv\Lib\site-packages\scipy\optimize\_optimize.py:105, in _wrap_callback.<locals>.wrapped_callback(res)
    104 def wrapped_callback(res):
--> 105     return callback(np.copy(res.x))

Cell In[2], line 15, in callback(xk)
     13 global iteration
     14 iteration += 1
---> 15 raise IOError("Help something went wrong")
     16 print(f"Iteration {iteration}: x = {xk}, f(x) = {objective(xk):.4f}")

OSError: Help something went wrong
```

Python3 – Exception Handling

- The content of most of that error is what’s called a “stack-trace”
- This is a full recall of how you got to the piece of code which is throwing an Exception.
- The Exception is telling the user something went wrong.
- Different Exceptions exist to indicate different types of errors.

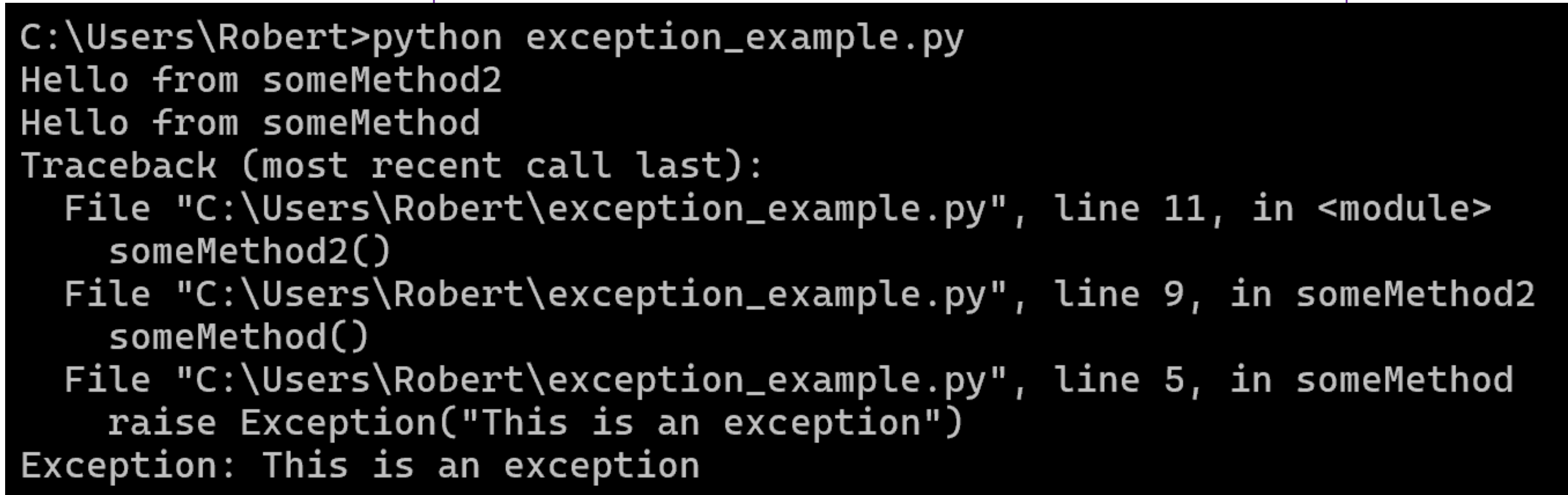
Python3 – Exception Handling

- In code this is relatively simple:



```python
def someMethod():
    print("Hello from someMethod")
    raise Exception("This is an exception")

def someMethod2():
    print("Hello from someMethod2")
    someMethod()

someMethod2()

print("Hello from after the exception")
```

- Running this gives an error:



```
C:\Users\Robert>python exception_example.py
Hello from someMethod2
Hello from someMethod
Traceback (most recent call last):
  File "C:\Users\Robert\exception_example.py", line 11, in <module>
    someMethod2()
  File "C:\Users\Robert\exception_example.py", line 9, in someMethod2
    someMethod()
  File "C:\Users\Robert\exception_example.py", line 5, in someMethod
    raise Exception("This is an exception")
Exception: This is an exception
```

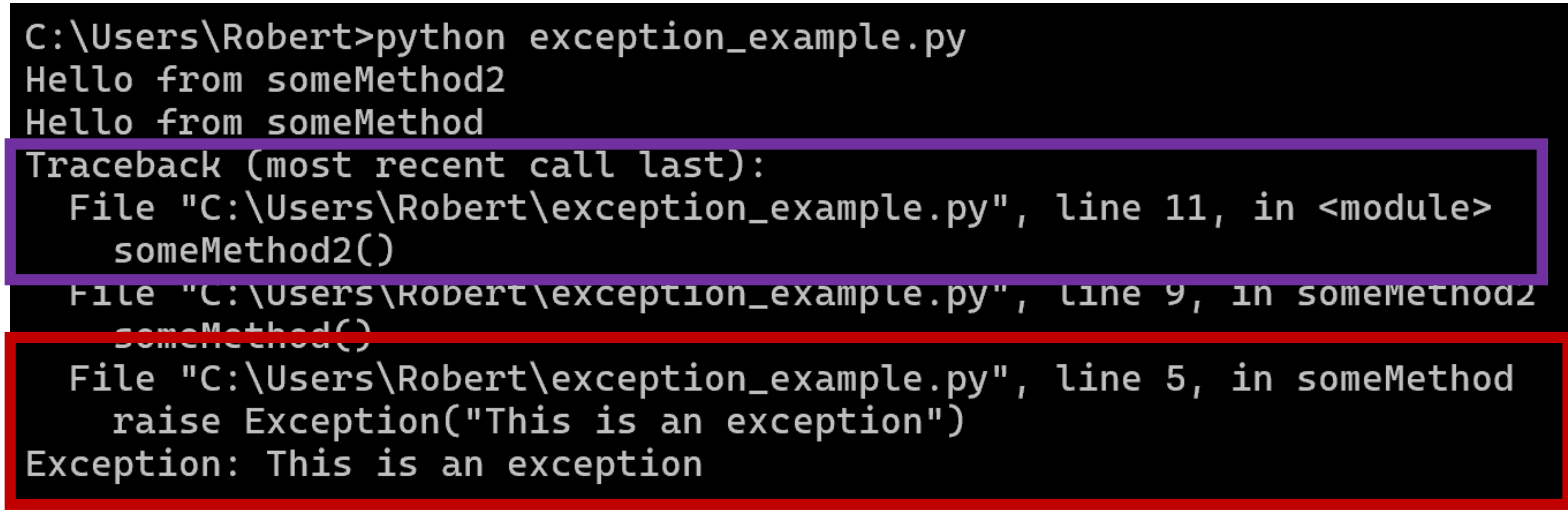
Python3 – Exception Handling

- What does the error mean?
- 2 Most important parts. The Top and the Bottom.

Top: Where the main program had reached by the time the error was raised(thrown)

Bottom:

What the exception/error was and where it reached.



```
C:\Users\Robert>python exception_example.py
Hello from someMethod2
Hello from someMethod
Traceback (most recent call last):
  File "C:\Users\Robert\exception_example.py", line 11, in <module>
    someMethod2()
  File "C:\Users\Robert\exception_example.py", line 9, in someMethod2
    someMethod()
  File "C:\Users\Robert\exception_example.py", line 5, in someMethod
    raise Exception("This is an exception")
Exception: This is an exception
```

Python3 – Exception Handling

- You should always take note of the bottom.
  Sometimes this is enough to fix an issue straight away.

  E.g. IOError – Did you use the correct filename?
  – Is the usb stick still attached?

  Sometimes not:

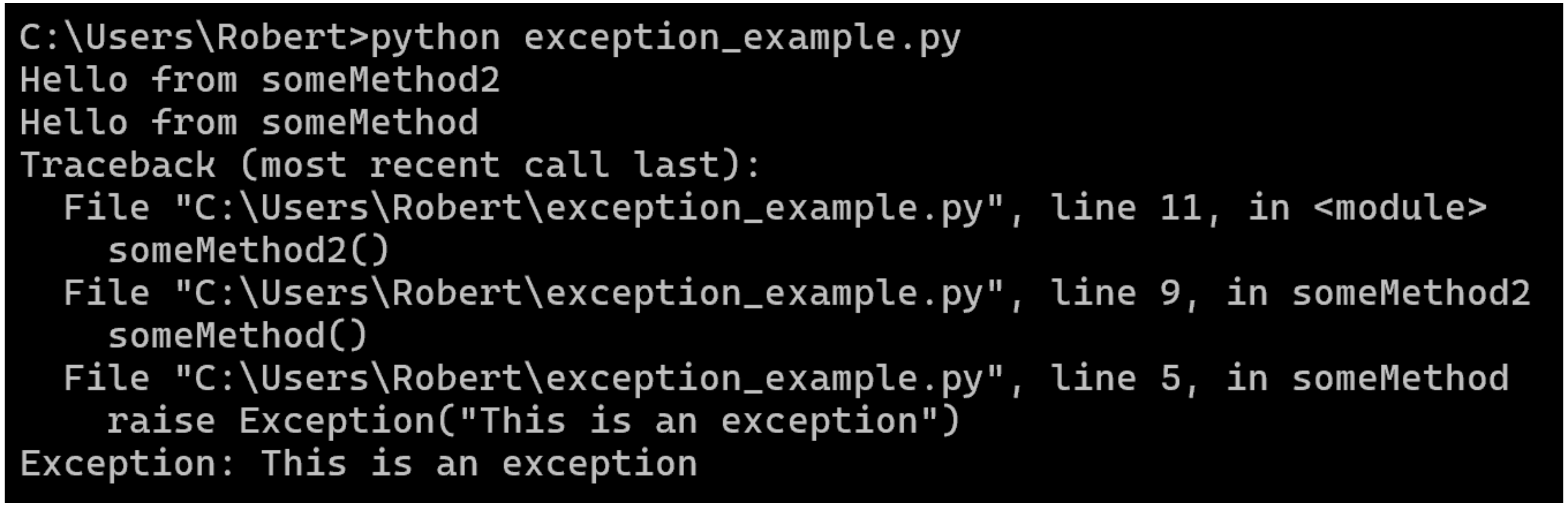
  E.g. Exception “Shouldn’t be here…”

- In the case that the error doesn’t make sense straight away or you’ve not seen it, read from the top->down.
- This details exactly what Python was doing at the time of the crash.

Python3 – Exception Handling

- In our simple case we called someMethod from within someMethod2.

- Top to bottom shows this being unravelled through the “stack”.



```
C:\Users\Robert>python exception_example.py
Hello from someMethod2
Hello from someMethod
Traceback (most recent call last):
  File "C:\Users\Robert\exception_example.py", line 11, in <module>
    someMethod2()
  File "C:\Users\Robert\exception_example.py", line 9, in someMethod2
    someMethod()
  File "C:\Users\Robert\exception_example.py", line 5, in someMethod
    raise Exception("This is an exception")
Exception: This is an exception
```

Python3 – Exception Handling

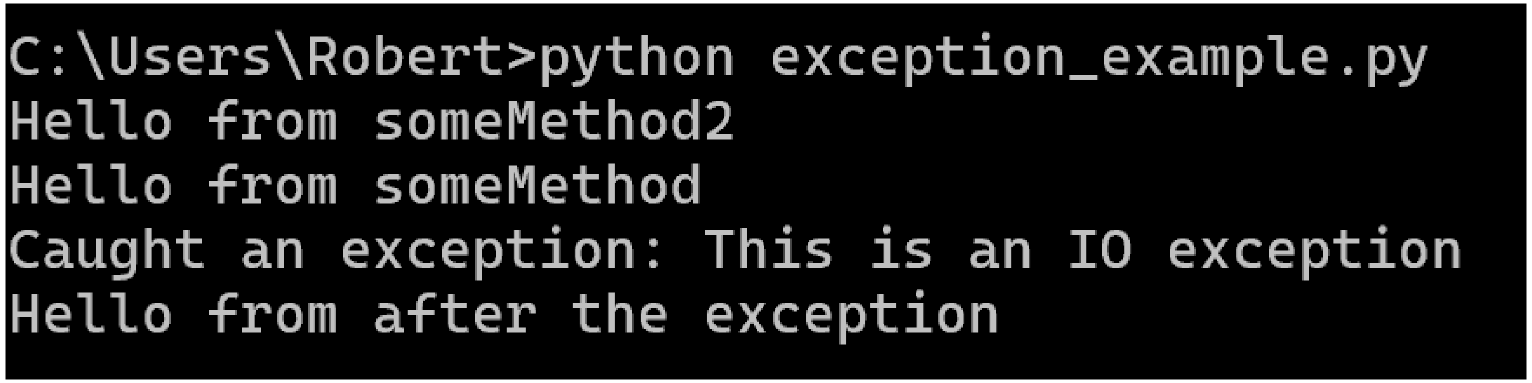
- Sometimes Exceptions are expected, or from buggy code that someone else wrote that you can’t or don’t want to fix.

- In this case you want to “**catch**” a “**thrown**” or **raise**-d exception:

```python
def someMethod():
    print("Hello from someMethod")
    raise IOError("This is an IO exception")

def someMethod2():
    print("Hello from someMethod2")
    someMethod()

try:
    someMethod2()
except IOError as e:
    print("Caught an exception: " + str(e))

print("Hello from after the exception")
```

Python3 – Exception Handling

- Running this no longer returns an error:



```
C:\Users\Robert>python exception_example.py
Hello from someMethod2
Hello from someMethod
Caught an exception: This is an IO exception
Hello from after the exception
```

- Catching the error allows the code to execute as intended ☺

- But beware, blindly catching all exceptions can hide catastrophic errors(!) ☹

Python3 – Exception Handling

- Exception handling has a bit of a “bad-rep”
- In early computing (pre-1990s) throwing an exception was heavily discouraged. It was expensive and costly in terms of lines of code or commands to execute on a processor.
- In modern systems when using Python, exceptions now have much, much, smaller performance overheads.

- It’s not good practice to use them to control your main program.

  But it’s good to know how to handle them to automatically tell if something worked as intended, such as “turn on the GPU and load X”.

Using Existing Python Packages

Python Software Ecosystem

- Software in Python comes distributed via a few different mechanisms.

- Host level (installed as part of your OS)

- User level pip (package from pypi.org you installed yourself)
- User level source (something you installed yourself from source)

- Anaconda (packages installed via anaconda)

Python3 – Packages

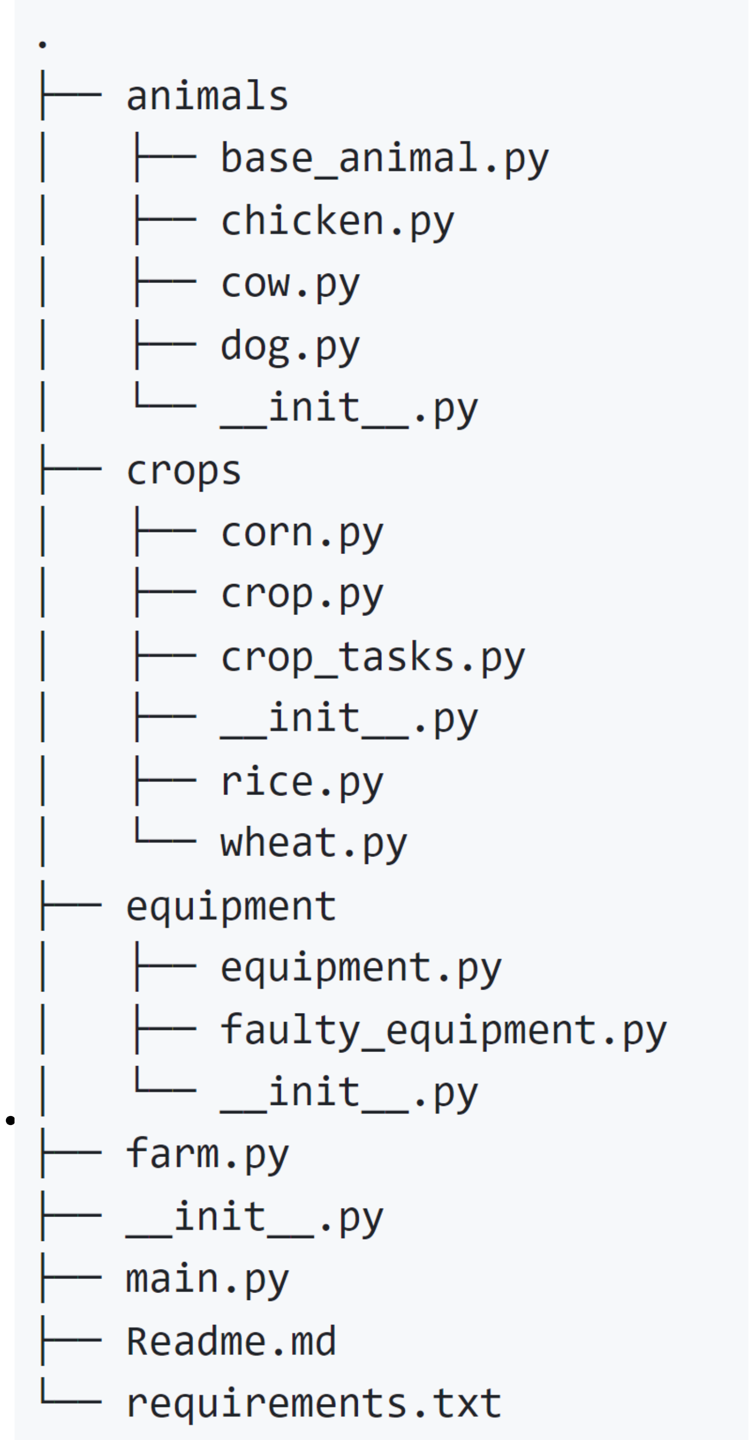
- A Python package is typically composed of multiple files.
- These can be borrowed into your existing code to allow you to use instructions from elsewhere to make your own instances.
- This allows you to import a class definition from elsewhere and use that in your own project.

Python3 – Packages

- https://www.github.com/rob-c/farm

- This “library” is composed of files across 3 different folders.

- Each folder contains a __init__.py file by convention.
  In a lot of cases this file is empty/missing.



```
.
├── animals
│   ├── base_animal.py
│   ├── chicken.py
│   ├── cow.py
│   ├── dog.py
│   └── __init__.py
├── crops
│   ├── corn.py
│   ├── crop.py
│   ├── crop_tasks.py
│   ├── __init__.py
│   ├── rice.py
│   └── wheat.py
├── equipment
│   ├── equipment.py
│   ├── faulty_equipment.py
│   └── __init__.py
├── farm.py
├── __init__.py
├── main.py
├── Readme.md
└── requirements.txt
```

Python3 – Packages

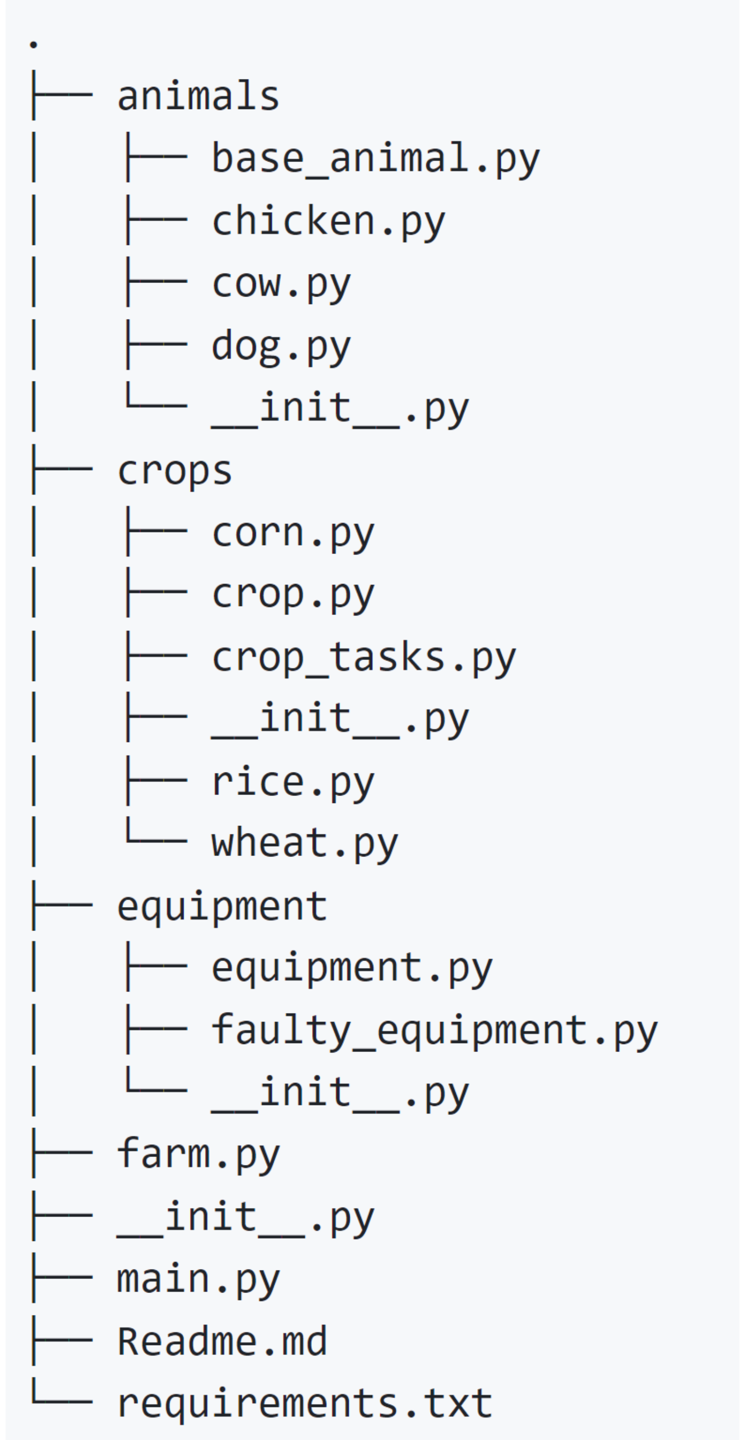
- From the root of the project "."

- I can run: `import animals.chicken`

  Behind the scenes this loads:

  `animals/__init__.py`
  then
  `animals/chicken.py`

- Now I can run `chicken.Chicken()`



```
.
├── animals
│   ├── base_animal.py
│   ├── chicken.py
│   ├── cow.py
│   ├── dog.py
│   └── __init__.py
├── crops
│   ├── corn.py
│   ├── crop.py
│   ├── crop_tasks.py
│   ├── __init__.py
│   ├── rice.py
│   └── wheat.py
├── equipment
│   ├── equipment.py
│   ├── faulty_equipment.py
│   └── __init__.py
├── farm.py
├── __init__.py
├── main.py
├── Readme.md
└── requirements.txt
```

Python3 – Packages

- It’s possible to have multiple layers of different classes within folders.

- Large frameworks such as PyTorch often have hundreds or thousands of different classes which can be imported.

- Best way to work out what you want for a given project is read the docs.

  Or,

  Ask an LLM of your choice for exanmples. (...but check its output!)

```
.
├── animals
│   ├── base_animal.py
│   ├── chicken.py
│   ├── cow.py
│   ├── dog.py
│   └── __init__.py
├── crops
│   ├── corn.py
│   ├── crop.py
│   ├── crop_tasks.py
│   ├── __init__.py
│   ├── rice.py
│   └── wheat.py
├── equipment
│   ├── equipment.py
│   ├── faulty_equipment.py
│   └── __init__.py
├── farm.py
├── __init__.py
├── main.py
├── Readme.md
└── requirements.txt
```

Python3 and Jupyter-Notebooks

Python3 and Jupyter-Notebooks

Jupyter Notebooks run Python, but they are not what most computing people thing of if you say running Python.

All code written in a notebook can be run from the command line.
Jupyter-notebooks can even be exported to be just raw Python code.

However, Jupyter notebooks keep the results of what you just ran inline with your code.
Data, numbers, graphics, the result of long-running many-CPU-hour tests or checks

Jupyter notebooks in principle support more than just Python, i.e. R, C++, RUST, BASH, …

For this course we will be sticking to Python3

ML/AI Software

- Many software libraries support ML style workflows.

  I will be exclusively using Python and Jupyter notebooks for this course.

- Advantages:

  - simpler to develop for, very dynamic
  - easy to learn
  - can express complex problems using simple code
  - wide adoption of language in industry

  - once setup, need no code changes* to access GPU, TPU, FPGA…

- Disadvantages:

  - potential performance overhead(s)
  - abstract interface is far away from problems/bugs
  - installation/setup can be… tricky…

*well, almost no code changes.

ML/AI Software – Frontend(ish) Projects

Typically speaking tend to use *just 1 or 2* of these per project/problem.

- **Jupyter** *(notebook, lab, …)*

  Biggest piece of software you will likely directly encounter doing data analysis/ML.

- **TensorFlow**

  This is the biggest library, used for building ML models for training, predicting & generating data.
  Uses Tensor objects to pass data around.

- **Keras**

  This can be thought of as the model component within TF but is developed independen
  Used for generating Neural Net models.

- **SciKit Learn**

  This project is widely used for classification, regression and clustering style problems.

- **PyTorch**

  This is used for building large AI models and is widely used in production by industry.

Icons from Wikipedia/project home-page

ML/AI Software – Backend(ish) Projects

Supporting projects, used as needed.

- **Numpy**
  This framework is a widely used Python numerical framewor
- **Pandas**
  Pandas is a library built around making an easily manipulated data frame
- **Matplotlib**
  This library is used for visualising data that is generated in Python and builds atop the Numpy framework
- **Seaborn**
  Statistical visualization framework built atop Matplotli
- **SciPy**
  This framework is built atop numpy and provides code to help with high-level mathematical functions such as fft, linear algebra, integration…

Icons from Wikipedia/project home-pag

ML/AI Software - *A word of caution*!

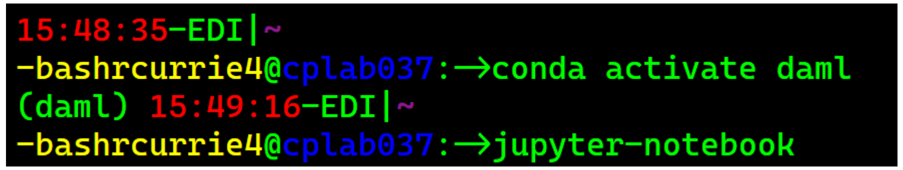
- It’s 2025, Python2 is now long dead!
  (If you’re too young to remember it’s ok)

- **Save regularly** and **<u>don’t be afraid of trying/breaking things</u>** ☺

ML/AI Software HowTo

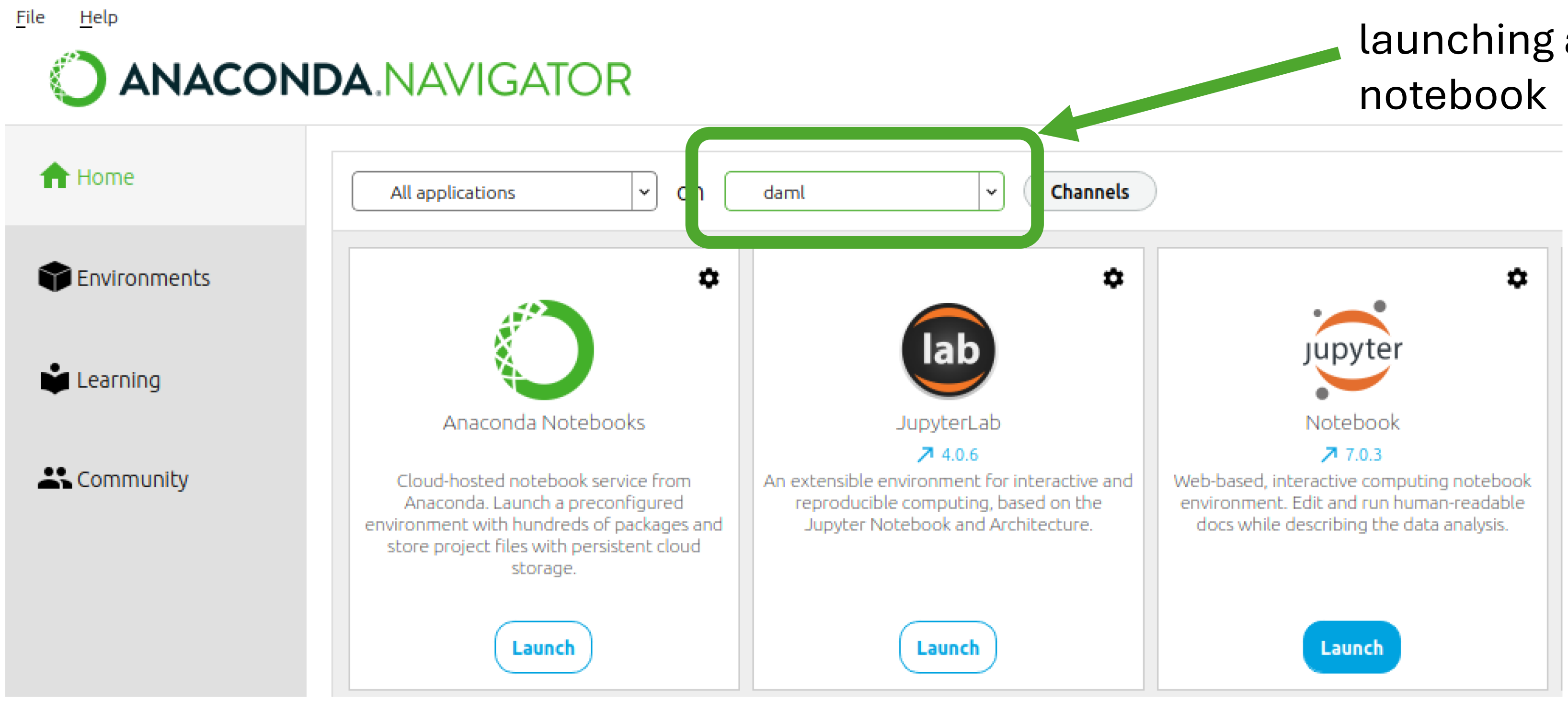
Make sure that you launch Jupyter notebooks from within the correct Anaconda environment:

Terminal:



```
15:48:35-EDI|~
-bashrcurrie4@cplab037:→conda activate daml
(daml) 15:49:16-EDI|~
-bashrcurrie4@cplab037:→jupyter-notebook
```

GUI:



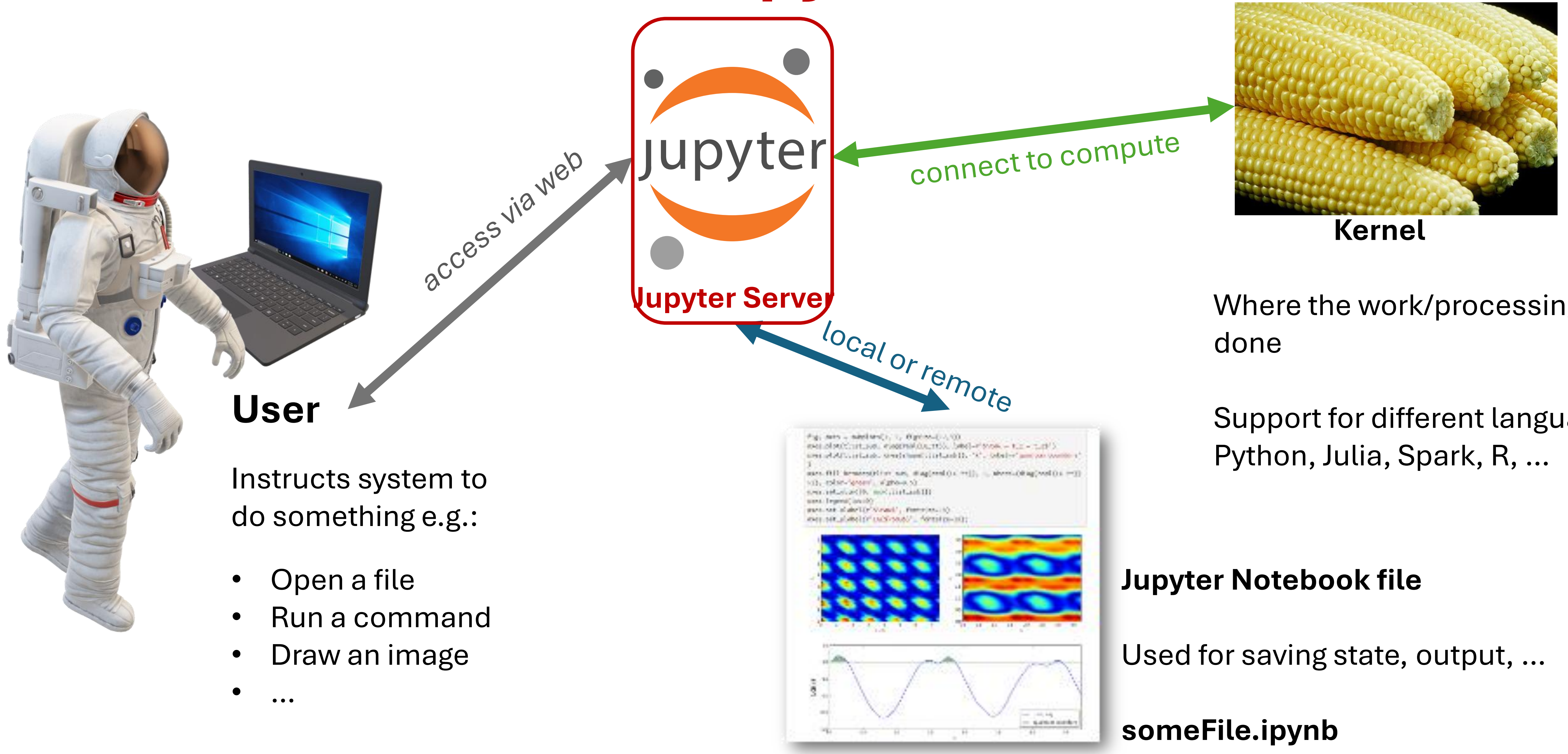
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Select the **AGQM** environment **BEFORE** launching a jupyter notebook

ML/AI Software – Jupyter notebook



User

Instructs system to do something e.g.:

- Open a file
- Run a command
- Draw an image
- …

Kernel

Where the work/processing is done

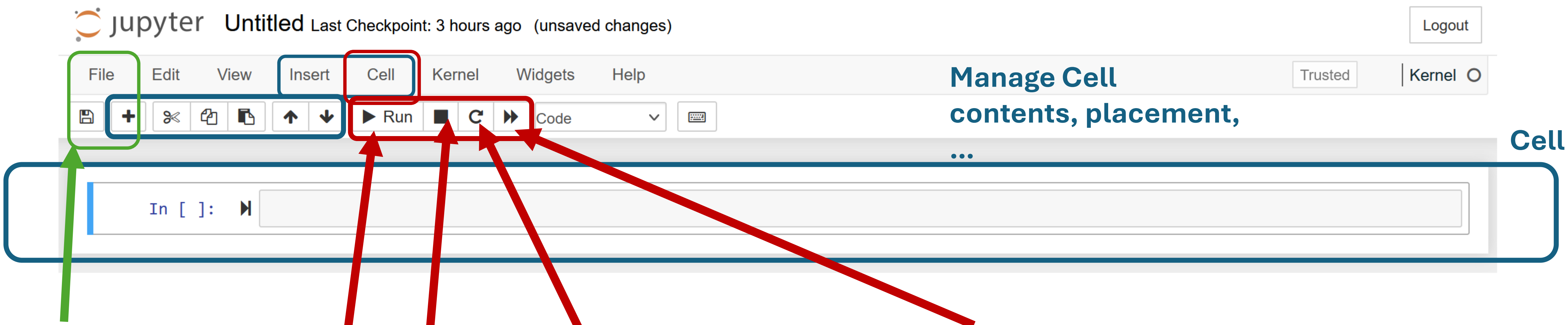
Support for different languages: Python, Julia, Spark, R, …

Jupyter Notebook file

Used for saving state, output, …

someFile.ipynb

ML/AI Software – Jupyter notebook



Manage Cell contents, placement, …

Cell

File operations, open/close/save/ …

Run the selected cell(<u>only</u>) now

Stop the current execution.

(Same as Ctrl+C or killing the process via the host-OS, technically interrupts the

Restart the Kernel

Beware this deletes all transient variables.

Reset the Kernel and re-run all the cells in the notebook

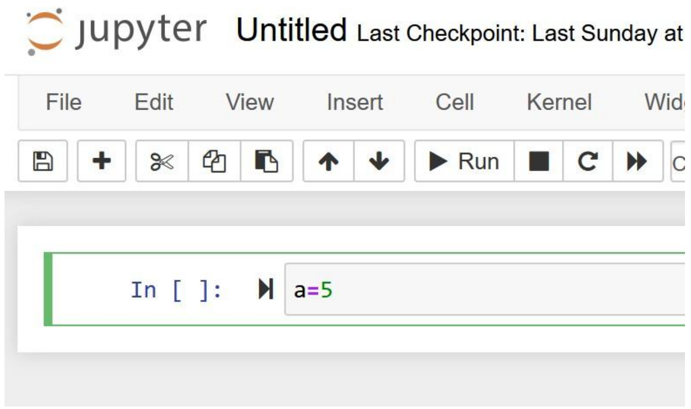
(will stop on error/exception)

My personal advice:

This is expensive/slow but it’s worth repeating <u>regularly(!)</u>

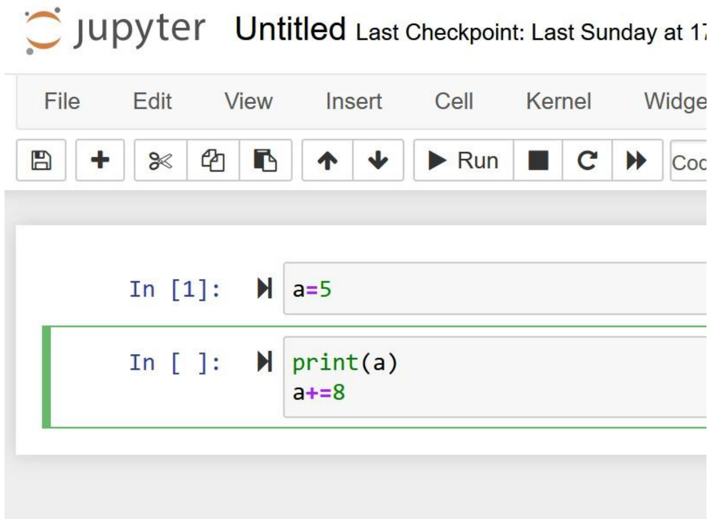
ML/AI Software – Jupyter notebook

1) First type command in the cell…



```
In [ ]:  a=5
```

2) Shit+Enter executes cell command(s)

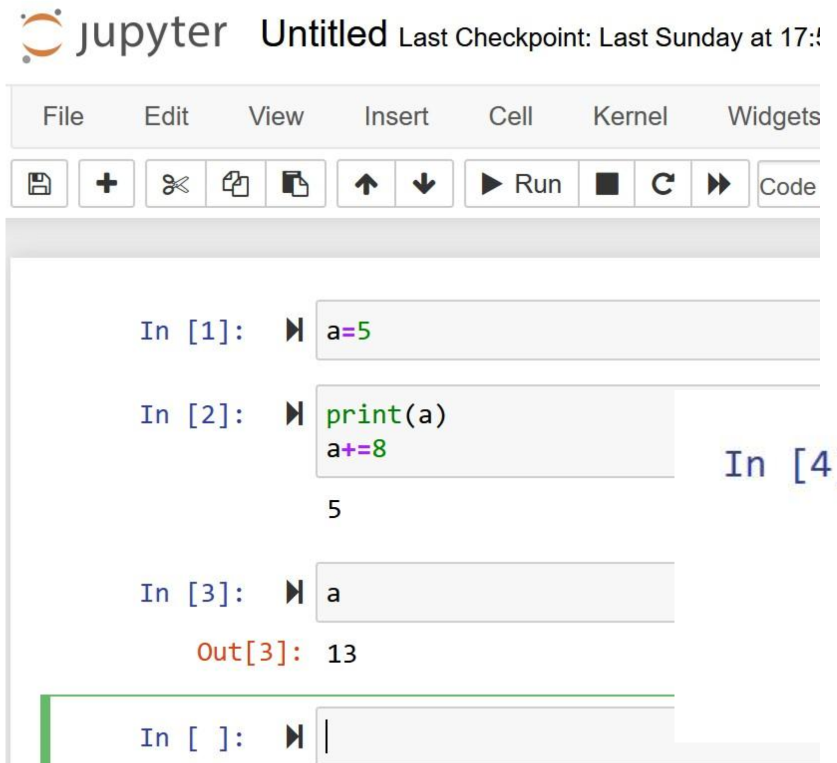


```
In [1]:  a=5

In [ ]:  print(a)
         a+=8
```

3) Using return within a cell allows for multiple commands per-cell.

4) Output from print(a) command gets saved.

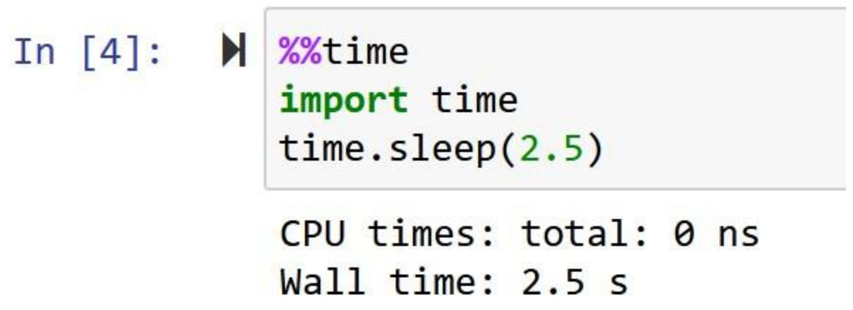
5) Executing an object prints the string value as in interactive Python.



```
In [1]:  a=5

In [2]:  print(a)
         a+=8

         5

In [3]:  a

Out[3]:  13

In [ ]:  |
```

6) **%%magic** commands at top of cells have special behaviour!

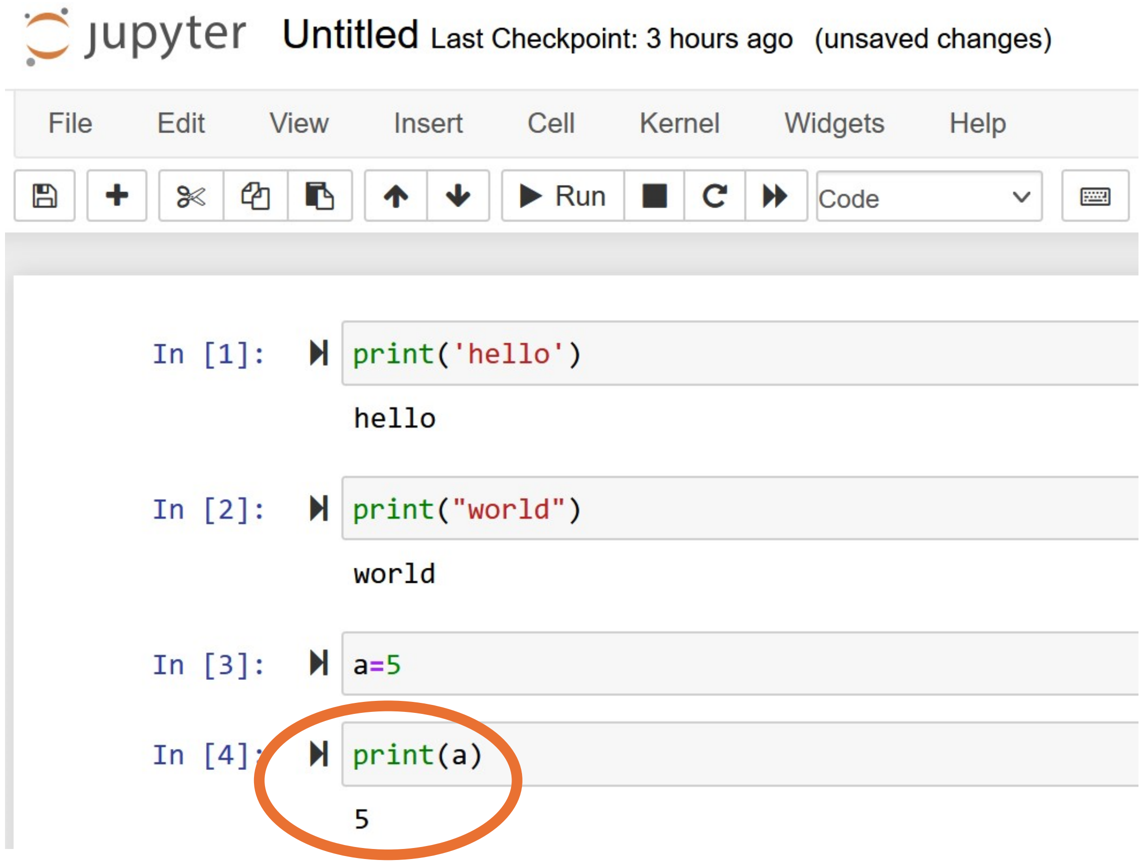
%%time allows us to time the execution of a command



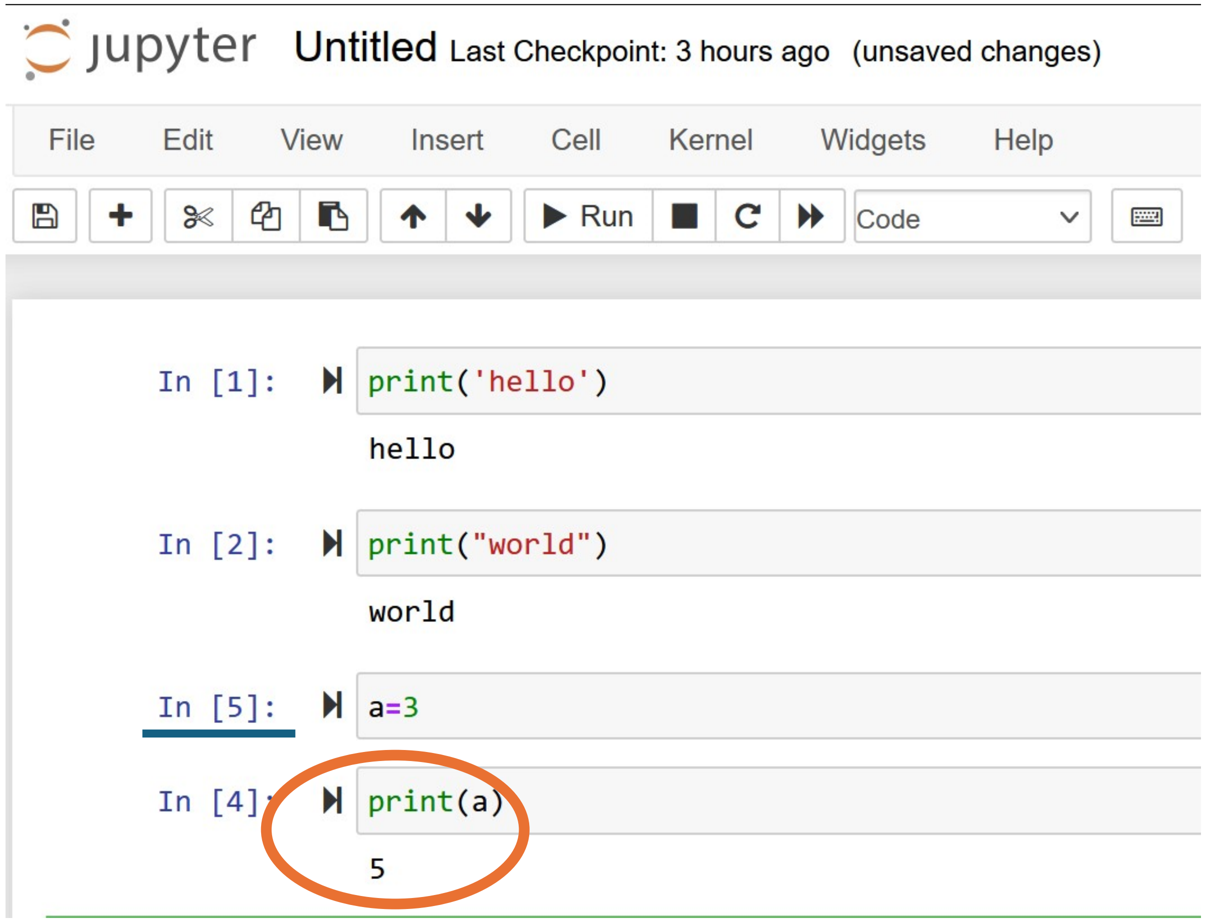
```
In [4]:  %%time
         import time
         time.sleep(2.5)

         CPU times: total: 0 ns
         Wall time: 2.5 s
```

ML/AI Software – Jupyter notebook

Good + makes sense Bad + Confusing(!)


jupyter Untitled Last Checkpoint: 3 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

```
In [1]:  print('hello')
         hello

In [2]:  print("world")
         world

In [3]:  a=5

In [4]:  print(a)
         5
```

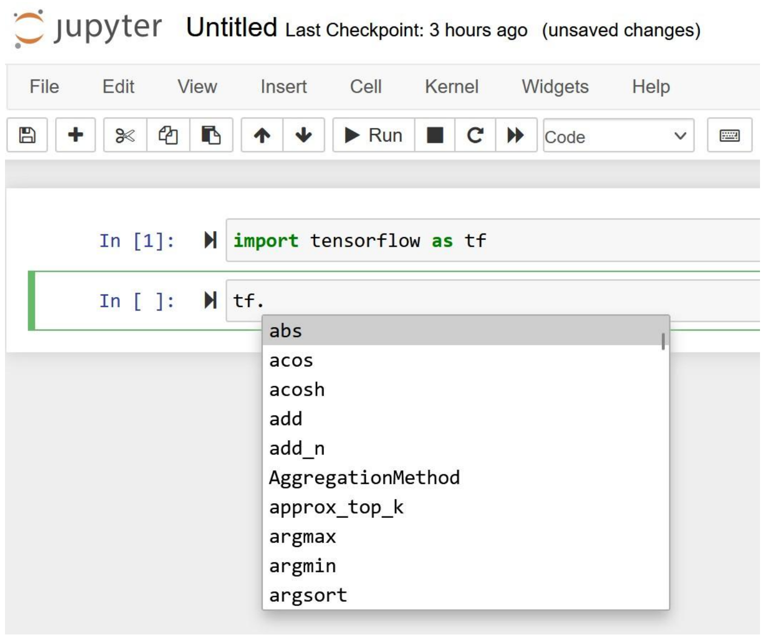

jupyter Untitled Last Checkpoint: 3 hours ago (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Run
Code

```
In [1]:  print('hello')
         hello

In [2]:  print("world")
         world

In [5]:  a=3

In [4]:  print(a)
         5
```

ML/AI Software – *using* Jupyter notebooks

TAB complete <*TAB*> i.e.

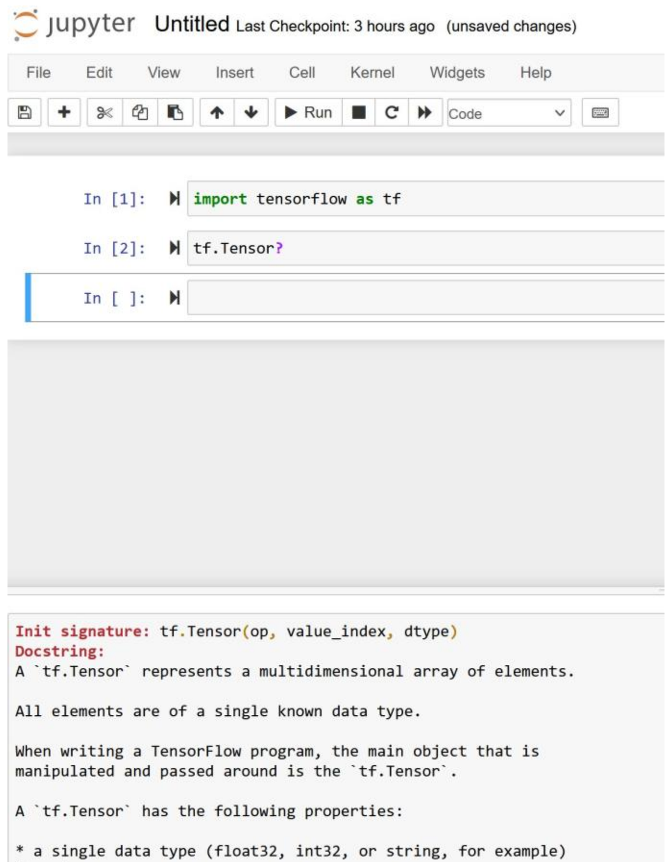
hitting tab after the `.` to see object attributes/functions in a scrollable menu



```
jupyter  Untitled Last Checkpoint: 3 hours ago  (unsaved changes)

In [1]:  import tensorflow as tf
In [ ]:  tf.
         abs
         acos
         acosh
         add
         add_n
         AggregationMethod
         approx_top_k
         argmax
         argmin
         argsort
```

Documentation quick-access: ?

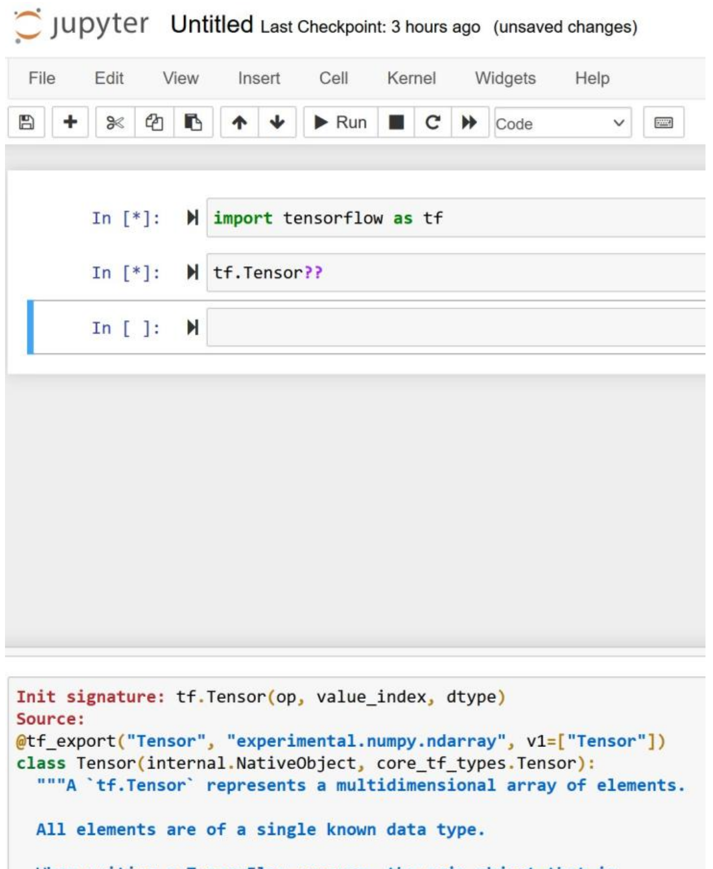
Execute `object?` to see the docs on the object



```
jupyter  Untitled Last Checkpoint: 3 hours ago  (unsaved changes)

In [1]:  import tensorflow as tf
In [2]:  tf.Tensor?
In [ ]:

Init signature: tf.Tensor(op, value_index, dtype)
Docstring:
A `tf.Tensor` represents a multidimensional array of elements.

All elements are of a single known data type.

When writing a TensorFlow program, the main object that is
manipulated and passed around is the `tf.Tensor`.

A `tf.Tensor` has the following properties:

* a single data type (float32, int32, or string, for example)
```

Source view: ??

Execute `object??` to see the source of the object



```
jupyter  Untitled Last Checkpoint: 3 hours ago  (unsaved changes)

In [*]:  import tensorflow as tf
In [*]:  tf.Tensor??
In [ ]:

Init signature: tf.Tensor(op, value_index, dtype)
Source:
@tf_export("Tensor", "experimental.numpy.ndarray", v1=["Tensor"])
class Tensor(internal.NativeObject, core_tf_types.Tensor):
  """A `tf.Tensor` represents a multidimensional array of elements.

  All elements are of a single known data type.
```

ML/AI Software – *using* Jupyter notebooks

Jupyter notebooks tie into external libraries to collect (and save) the output into the notebook.

Notebooks do NOT save the transient(running) state of the kernel.

This means you can't expect to re-open a notebook and re-run "cell 56" without first running cells 1-55.

ML/AI Software – *using* Jupyter notebooks

There's an example notebook: "**data-science-tools.ipynb**" please read through it.

This notebook goes through examples raw data manipulation using:

- Numpy Arrays
- Statistics
- Pandas DataFrames
- Matplotlib

It's useful to read through this notebook, play with values, experiment with any notation you're not familiar with.

Using Python for Numerical Minimization

- Most problems in this course are performing numerical minimization over complex datasets with complex functions.
  (aka, statistical problems)

- There are simple examples demonstrating minimization in a jupyter notebook for this workshop for those keen to explore this in more detail.

- This will be more important for next lecture/workshop but we'll introduce the concepts here and give you a chance to play with some "fitting" to see what is involved with this behind the scenes.

Using Python for Numerical Minimization

- In a perfect world we're dealing with analytical functions with well defined **Jacobian** and **Hessian** matrices.

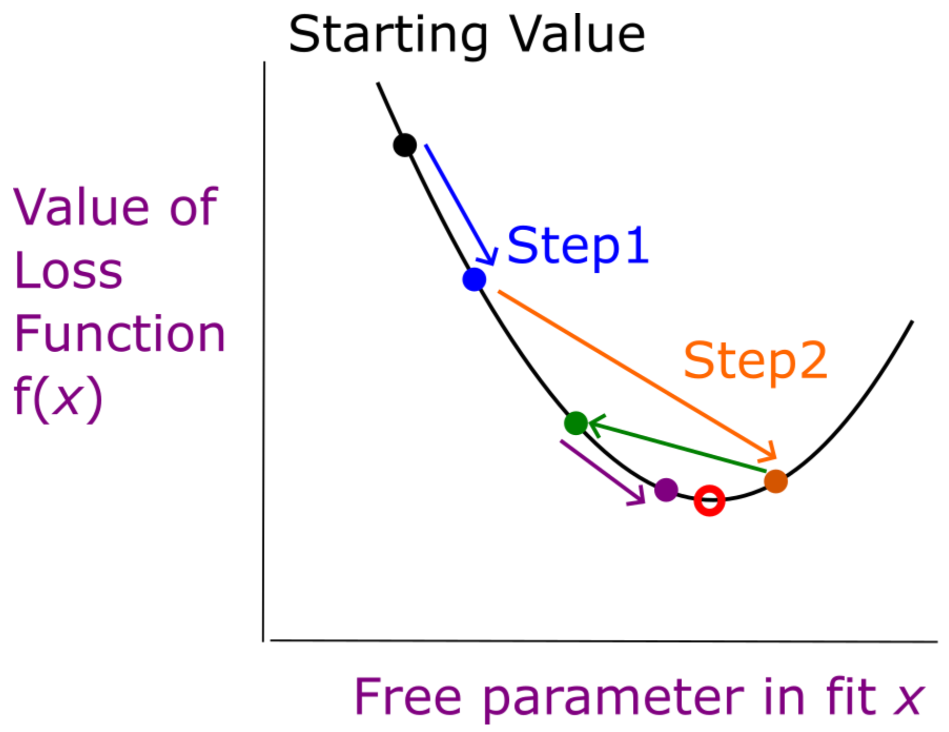
- In the real-world (and Machine Learning) problems quickly jump into parameter-spaces with millions or billions of free parameters.
- And, we don't really know the exact function, or the derivatives we're working with.

- With that in mind we must resort to **numerical techniques** to extract the gradient of our function.

Numerical Minimization – Gradient Descent

The process of "**Gradient Descent**" makes use of the derivative of free parameters and **Learning Rate** to adjust their values.

1D example case



Step1 f'(x) -ve go right

Step2 f'(x) +ve go left

Step3 f'(x) -ve go right

Step4 f'(x) -ve go right

...

Step n f'(x) zero, at minimum

**Learning Rate* x *Gradient* gives the size of the step at each point*

Numerical Minimization – Stochastic Gradient Descent

Dealing with large datasets can be a difficult prospect.
In the real world these often end up many **GB**, **TB** or a *few* ***PB*** in size.

When fitting using datasets so large we divide them into ***batches*** and iterate.

SGD is a statistical technique.

Instead of calculating the gradient of the loss function over a ***whole dataset***
(as in **GD**) the gradient is calculated using ***randomly selected batches*** at each step. *

This has the advantage of potentially being *much faster* and offers a more resilient route to the global minima.
It is also *potentially* less impacted by local minima within the whole function space.

*In practice this is usually updating *every* batch

Numerical Minimization – Other Routes to the Bottom

As well as **SGD** there are other algorithms designed to calculate how to find the global function minima.

The most common example of this is the "***Adam***" algorithm, which is an example of an "***adaptive optimizer***".

Without going into implementation details, ***Adam*** is often a superior algorithm because it also does the following:

- ***Remembers the momentum of previous optimization steps***
  i.e. Using previous steps to decide how quickly/slowly to adjust.
- ***Applies regularization to internal weights***
  This reduce the impact of outliers in the calculation of the loss function.
- **Dynamically adjusts the Learning Rate during training**

GIT

(not just an offensive elephant)

Git – An Intro

Git is the most popular version control system in use today.

What does this mean?

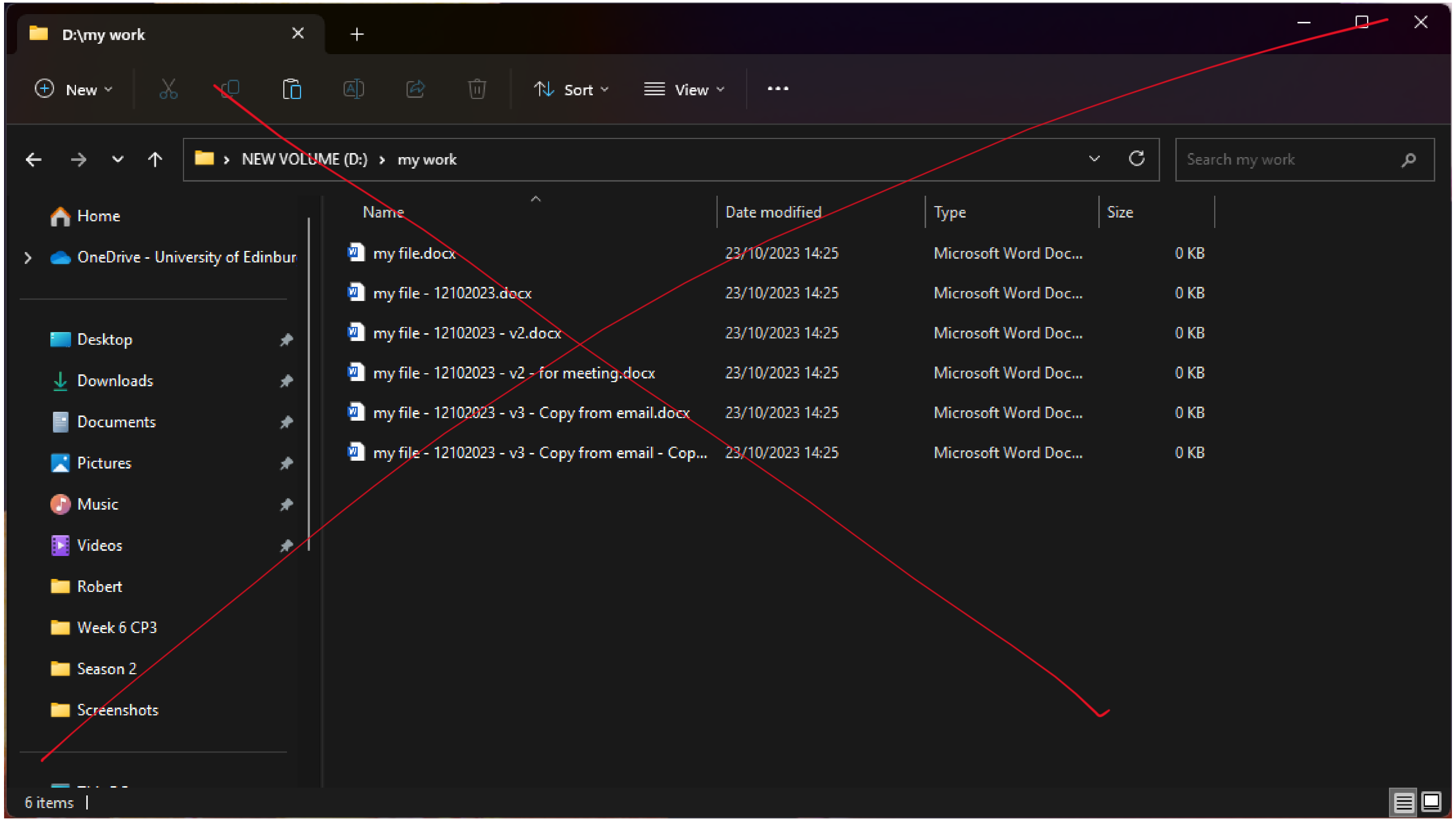
Imagine you have a file and want to make changes.

You want to also keep the old version of the file to refer to later.

The ‘simplest’ solution is to make a new file so that you have 2 files.

This sounds good, but when you want to work on a file with many authors or a thesis this approach can cause problems.

Git – An Intro



D:\my work

NEW VOLUME (D:) › my work

Name	Date modified	Type	Size
my file.docx	23/10/2023 14:25	Microsoft Word Doc...	0 KB
my file - 12102023.docx	23/10/2023 14:25	Microsoft Word Doc...	0 KB
my file - 12102023 - v2.docx	23/10/2023 14:25	Microsoft Word Doc...	0 KB
my file - 12102023 - v2 - for meeting.docx	23/10/2023 14:25	Microsoft Word Doc...	0 KB
my file - 12102023 - v3 - Copy from email.docx	23/10/2023 14:25	Microsoft Word Doc...	0 KB
my file - 12102023 - v3 - Copy from email - Cop...	23/10/2023 14:25	Microsoft Word Doc...	0 KB

6 items

Git – For a ~~Physicist~~ Beginner

I have some work and want to share it with someone.

Easiest option will likely be to use the services at github.
(Alternatives exist such as bitbucket, …)

Gitlab/UoE/<other-servie-here> have gitlab instances which are also good, but don't always have a public facing version of the repo.

So, you get an account, upload your code/document/photos and get back to work.

NB: GitHub ≈ GitLab but there are some minor changes in the names of things.

GitHub – For a ~~Physicist~~ Beginner

- **First** you need to have a github account (out of scope of this talk)
  https://education.github.com/students

- **Second** you will want to make a github **token**.
  https://github.com/settings/tokens/new

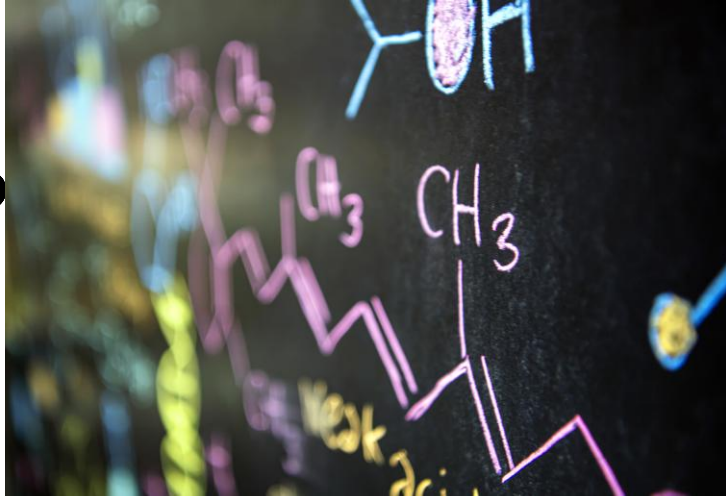
- What's a **token**?

  Ignoring a lot of subtleties, a token can be thought of as a secure password that optionally has an expiry associated with it.

  The idea behind this is to offer much better security than passwords

- Somewhere to put stuff; a **new** or **existing** repo.

Git – What do I want to do with it?



The requirements for git are typically something like the following:

1. I have some code/files either I wrote or copied
2. Keep track of the changes I make with this code/files
3. Have copy of this work on my device and somewhere remotely
4. I want other people to be able to see and potentially change it

Git – Why is it different?

The revolutionary idea that makes git different is that **all the information** stored in the repo is **always available** once it's been "checked out".

This means that if you go and grab the git repo to a major project you can go backwards in time and see what it looked like 3 months or 3 years ago without having to be connected to a remote service.

"My backups are everyone else's machines" – L. Torvalds creator of git

Git – The essentials

Command	What it’s used for
git init	Making new repos
git clone	Downloading a local copy of a repo from somewhere
git status	Have there been any local changes
git diff	Examining what a change(s) look like before committing them
git add	Add a new file to a git repo
git rm	Remove a file from a git repo
git commit	Putting local changes into your copy of the repo
git push	Copying your local changes remotely
git pull	Copying remote changes back into your local repo
git log	View the history of changes within a repo
git checkout	Checkout to a specific revision of the repository

GitHub – Beginners – making a new repo

- Need to create a repo.
- You can use the command line:

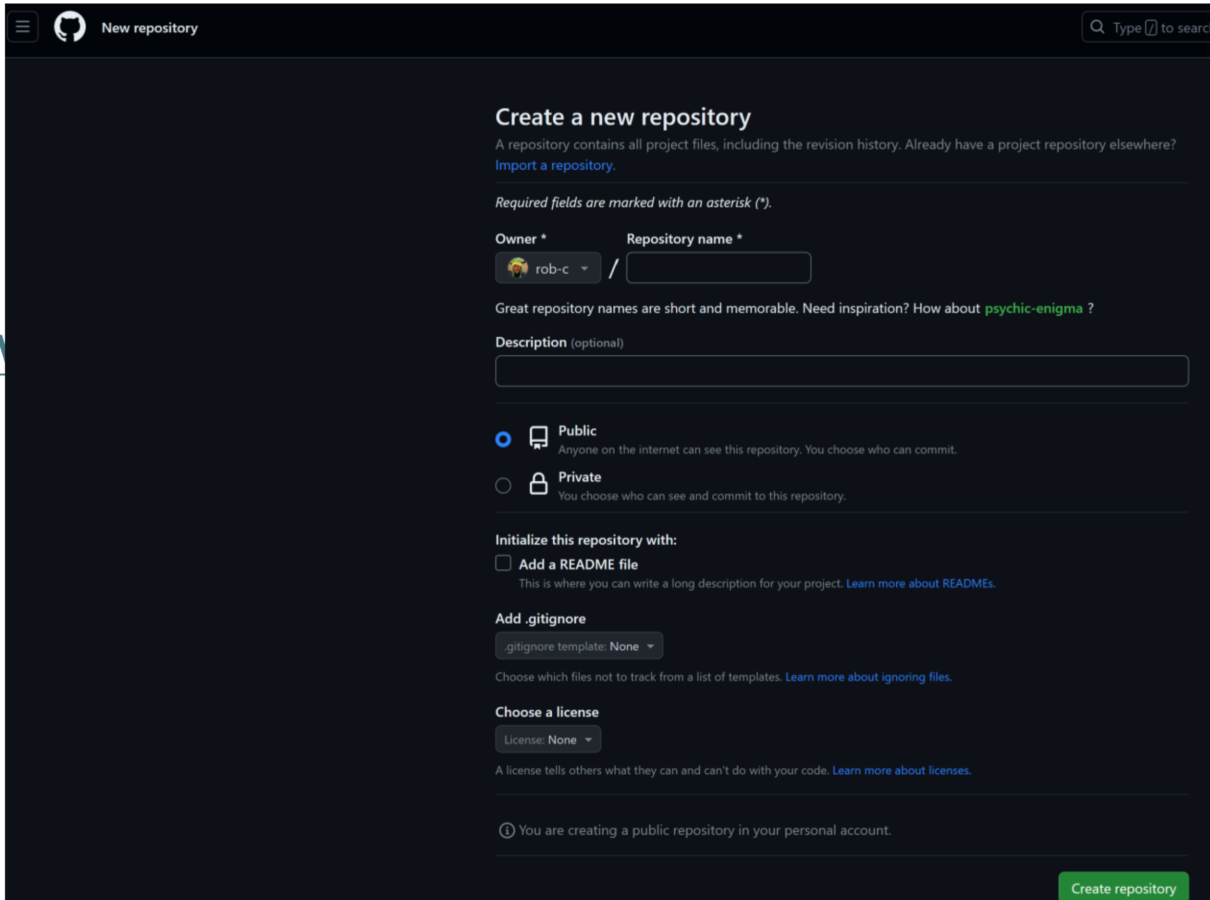
  ` git init` .
- I rarely need to do this; I suggest you avoid this to make life easier.
- For simplicity make a repo via the web-interface and then make changes unless specific instructions tell you otherwise.

GitHub – Beginners – making a new repo

- Making a new repo varies per system. I’ll talk about github. If you struggle with another site, I recommend asking the service admin or google.

- For Github: https://github.com/new

- Just need a repo name and you’re good to go. ☺



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Owner * rob-c / Repository name *

Great repository names are short and memorable. Need inspiration? How about psychic-enigma ?

Description (optional)

[✓ 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:

☐ Add a README file
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore
.gitignore template: None
Choose which files not to track from a list of templates. Learn more about ignoring files.

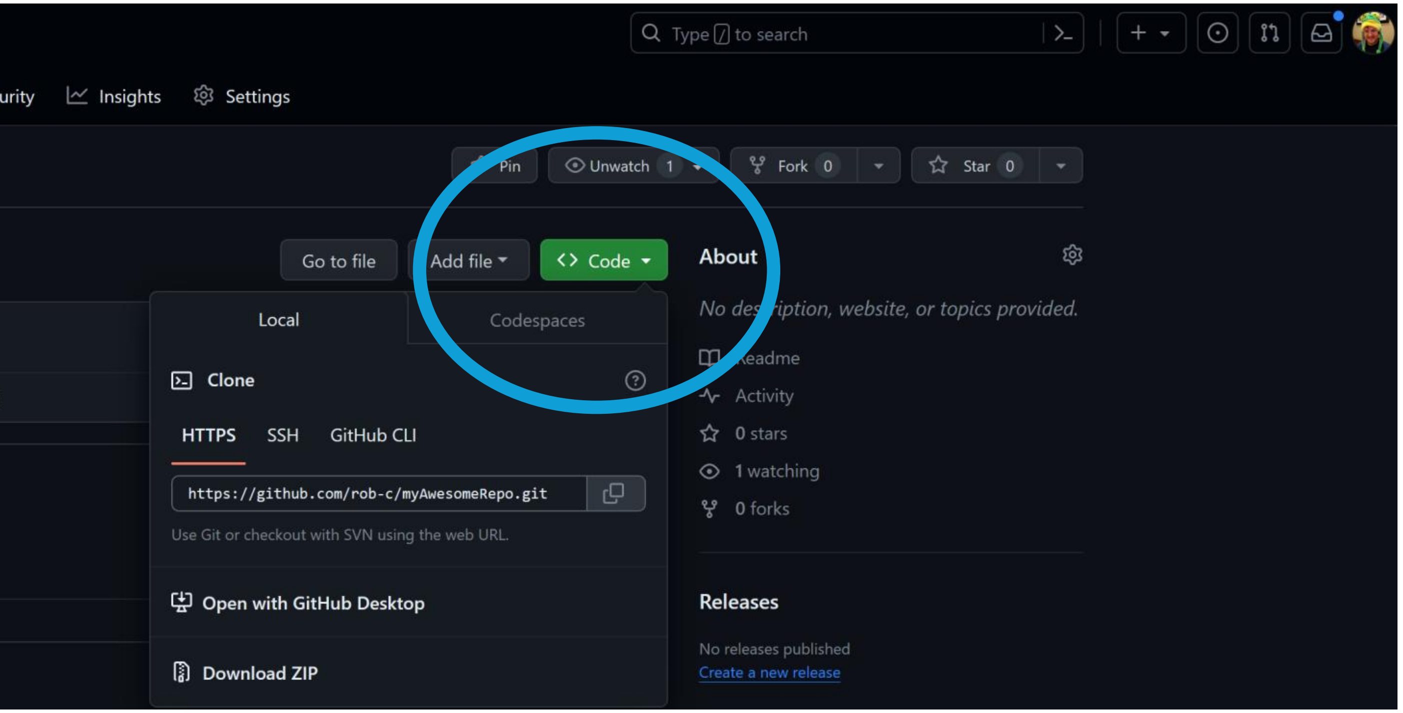
Choose a license
License: None
A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a public repository in your personal account.

Create repository

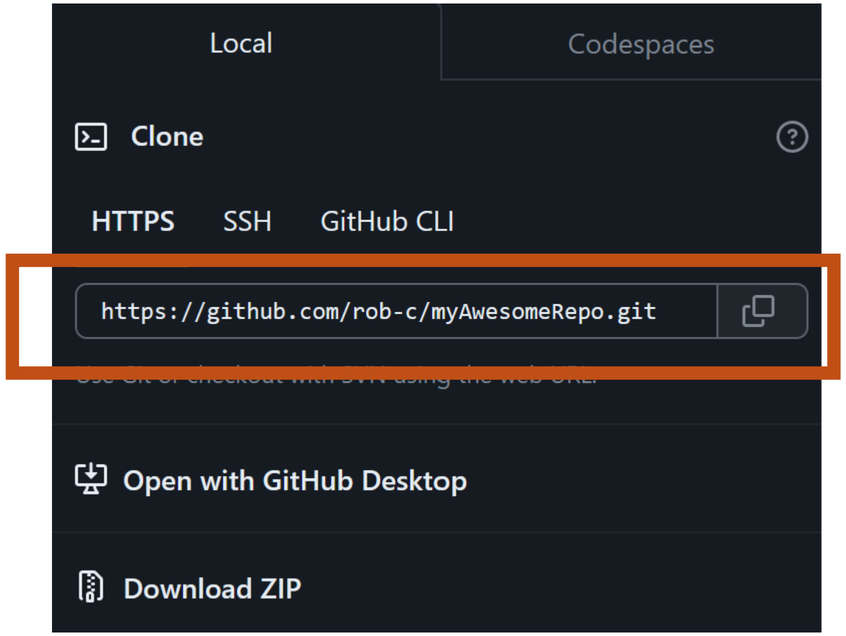
GitHub/GitLab – An existing repo

- Typically, there is a clone button on the web interface, e.g:



```
https://github.com/rob-c/myAwesomeRepo.git
```

Use Git or checkout with SVN using the web URL.



```
https://github.com/rob-c/myAwesomeRepo.git
```

GitHub/GitLab – An existing repo

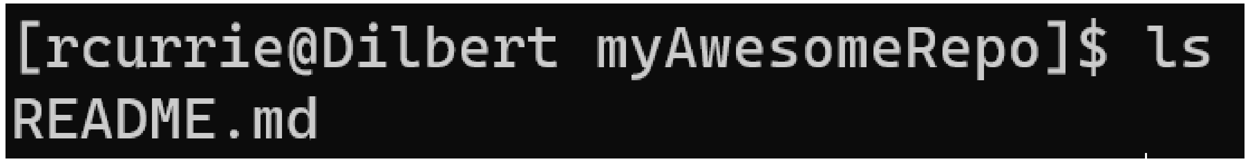
- Now we have an address of an existing repo we can **clone** it:

- > **git clone** https://github.com/rob-c/myAwesomeRepo.git

  (Tip: I always prefer the https protocol for this, because the git protocol isn’t supported everywhere. There’s complicated networking reasons behind this I don’t want to get into)

GitHub – Beginners

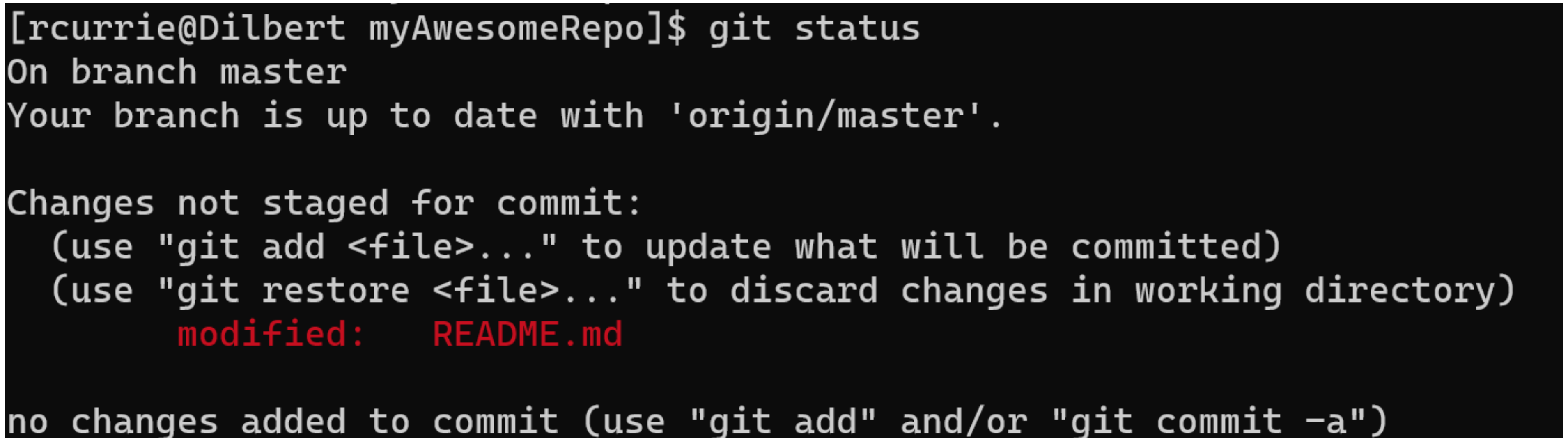
- Now we have a folder ‘myAwesomeRepo’ let's **cd** into it.
- Now we have a **local** copy of a repo:



```
[rcurrie@Dilbert myAwesomeRepo]$ ls
README.md
```

- We use a local copy for making changes.
- Edit a file and run

  git status



```
[rcurrie@Dilbert myAwesomeRepo]$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

GitHub – Beginners

- Now we have a local copy of the repo. And have made changes.

- If we want to make the changes available to others, we need to:

  - Commit our changes to the repo
  - Check our repo is in sync with the remote repo
  - Upload our changes to the remote repo

- Once that is done someone else with a remote copy call ‘pull’ our changes.

- So how do we do this?

GitHub – Beginners

1. First, we need to commit our changes:

   ```
   git commit –m “my first change” ./theChangedFile
   ```

2. Then we check we are in sync with the code on github:

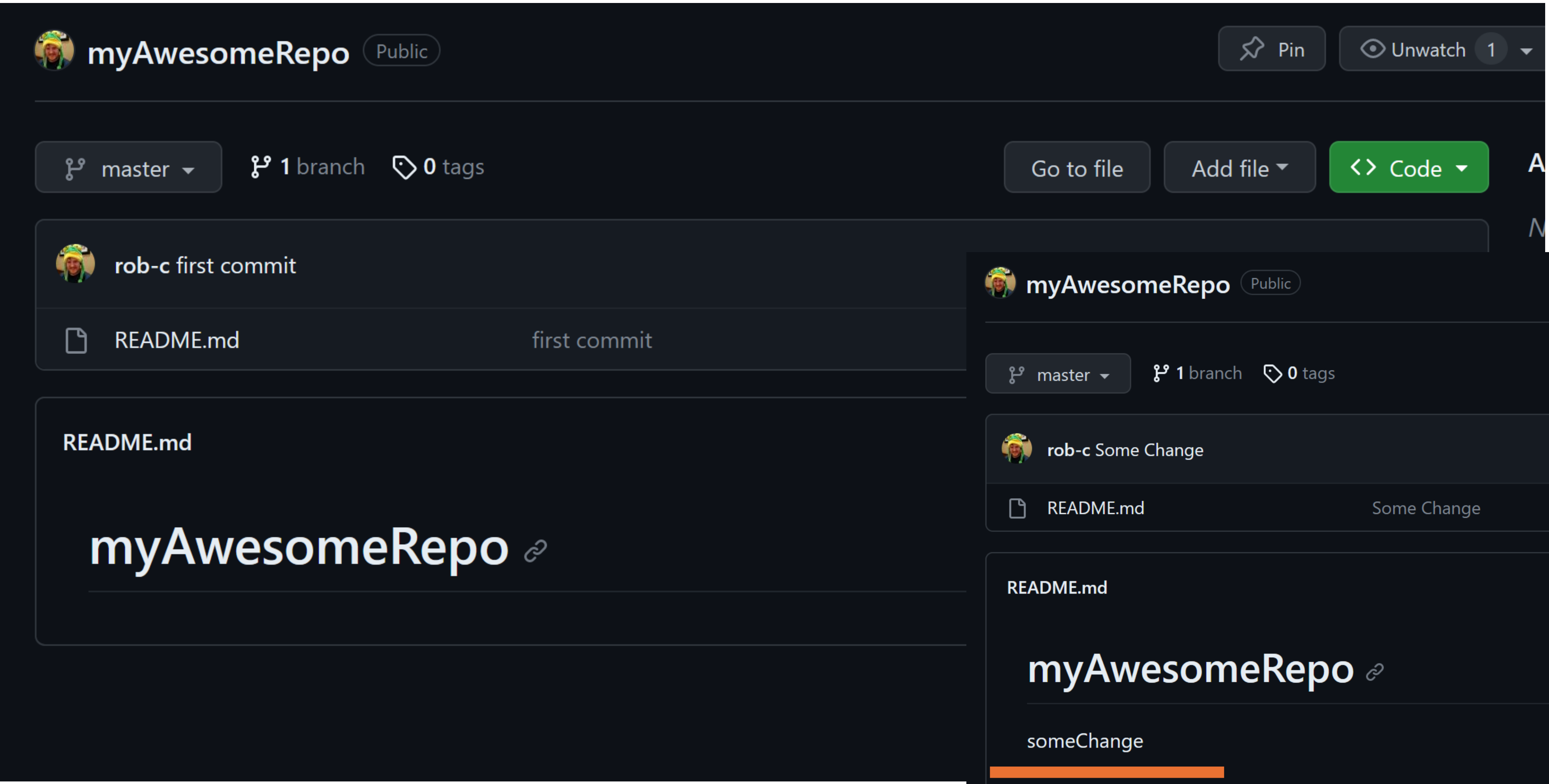
   ```
   git pull
   ```

3. Then if everything is good, we push our change to github:

   ```
   git push
   ```

   (This step will ask for your username and your **token**!)

GitHub – Before and After

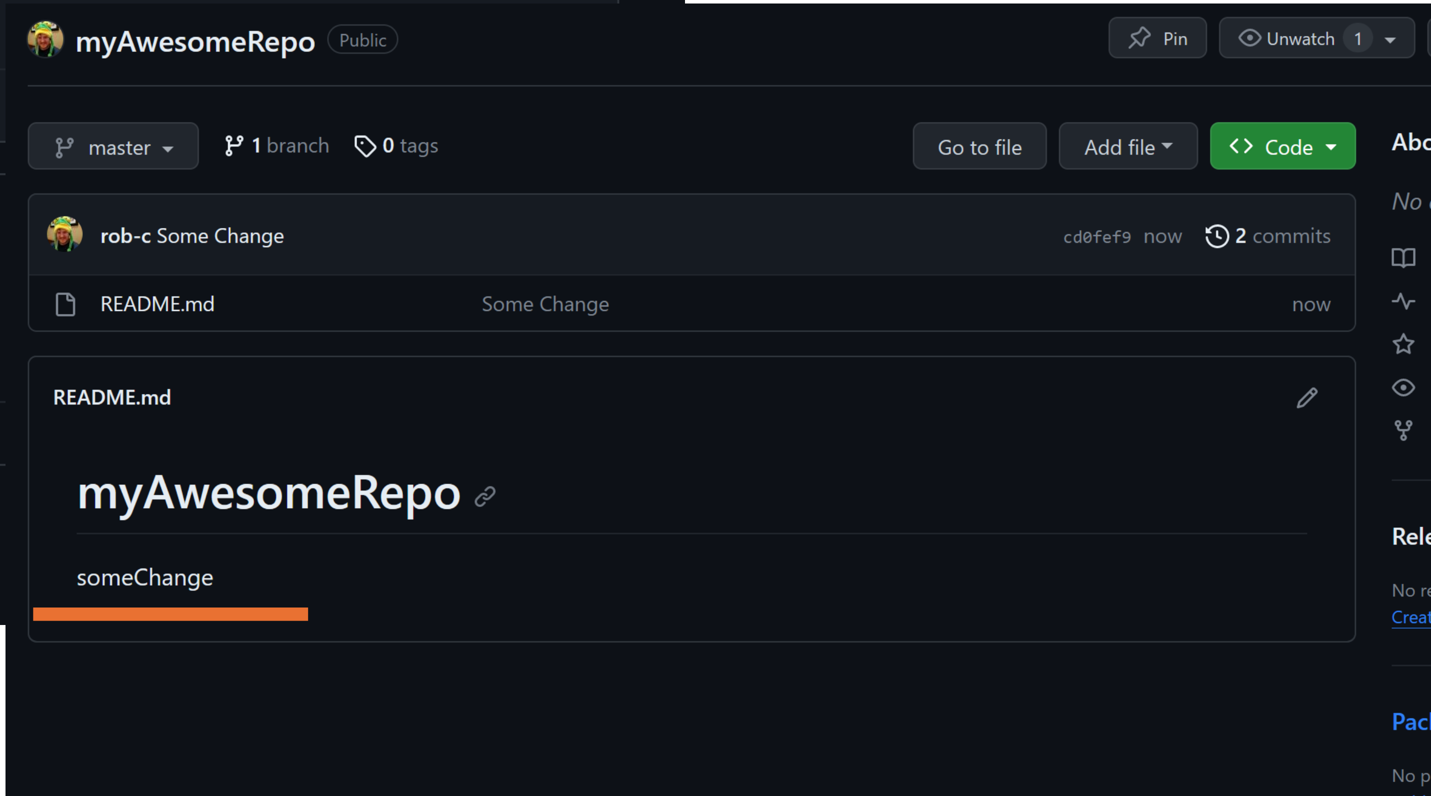


rob-c first commit

README.md first commit

README.md

myAwesomeRepo



rob-c Some Change cd0fef9 now 2 commits

README.md Some Change now

README.md

myAwesomeRepo

someChange

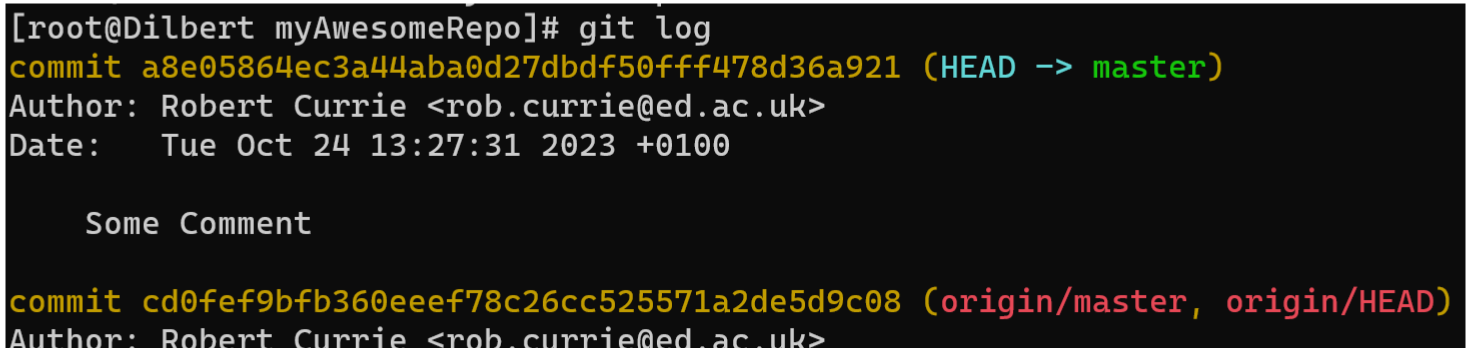
GitHub – Version Control

The scenario: You've been working on something complex, and you have been making changes over the last few weeks. Now, you have a question about how the code looked a few days ago before some changes. How do you look at the code from back then?

git log gives the history of changes with unique **commitIDs** and timestamps for your repo.

Using a given **commitID** you can revert your repository to this previous state temporarily so, you can examine the file

git checkout cd0fef9bfb360eeef78c2



```
[root@Dilbert myAwesomeRepo]# git log
commit a8e05864ec3a44aba0d27dbdf50fff478d36a921 (HEAD -> master)
Author: Robert Currie <rob.currie@ed.ac.uk>
Date:   Tue Oct 24 13:27:31 2023 +0100

    Some Comment

commit cd0fef9bfb360eeef78c26cc525571a2de5d9c08 (origin/master, origin/HEAD)
Author: Robert Currie <rob.currie@ed.ac.uk>
```

GitHub – Easy Right?

- Since git is easy everyone loves it. But collaborating with larger groups on bigger projects gets tricky so let's talk about some advanced topics.

- **Forking a repo**

  This is the same as taking a copy of a project so you can work on changes in isolation.

- **Branching**

  This is when you make a private workspace in a project to work on changes and be able to share them without breaking the stable part if the project for everyone.

GitHub – Using branching effectively

The full process for working on a local branch with a remote repo is:

1. **git clone** https://...../myRepo
2. **cd** myRepo

3. **git branch** newFeature

4. **git checkout** newFeature
5. ***... work ... work ... work ...***
6. **git commit** –m “my new feature is finished” ./file1 ./file2 ...

7. **git push –u origin** newFeature

After here you make a MR/PR for peer-review, or merge your branch back into your mainline branch.

GitHub – Advanced Workflows

- Without going into details 2 common advanced workflows are:

Branch	Fork
1. Create a new branch	1. Fork a project
2. Checkout the branch	2. Make changes
3. Make changes	3. Commit changes
4. Commit	4. Push to project directly
5. Push for review	5. Invite others to contribute
6. Make a PR (Pull Request) or MR (Merge Request)	6. When happy make a PR/MR against original project
7. Make changes as discussed with collaborators	7. Collaborate and fix
8. Merge changes into project for everyone	8. Changes accepted by original project

- If you want to get an advanced experience with github branching I recommend: https://learngitbranching.js.org/ ← learn in the browser

Git – What can go wrong?

- If 2 people have edited the same file their changes may **conflict**.

- If you're using git properly you will notice this typically during **git pull**

- The correct approach is to follow the instructions that appear.

  - The pull has been paused mid-flow due to a conflict
  - ***Fix*** the ***commit*** such that there is just 1 change present
  - Commit the ***fixed*** code to the repo
  - Tell git to continue with the pull request it was performing

Git – When things go really wrong ☹

- Although it’s always possible to fix almost all problems with fancy clever git commands a lot of experts will sometimes just admit defeat as it can be quicker to start again.

- If you hit a lot of conflicts which aren’t caused by changes you’ve made a lot of people resort to the following:

1. Backup your changes
2. Git clone again into a new folder
3. Make your finished changes again
4. Push to the remote repo.

NB: This should be a **LAST RESORT**. You lose change history, what the code looked like, why you ended up making the changes the way you did.

Git – Review

- Using git should be easy once you are familiar with the basics.
- It can require a lot of experience when things go wrong, don’t be afraid of breaking things for others, all commands apart from **git push** operate on your local copy of everything.
- Some more useful online resources are:

https://gitimmersion.com/
https://githowto.com/
https://swcarpentry.github.io/git-novice/

WORKSHOP

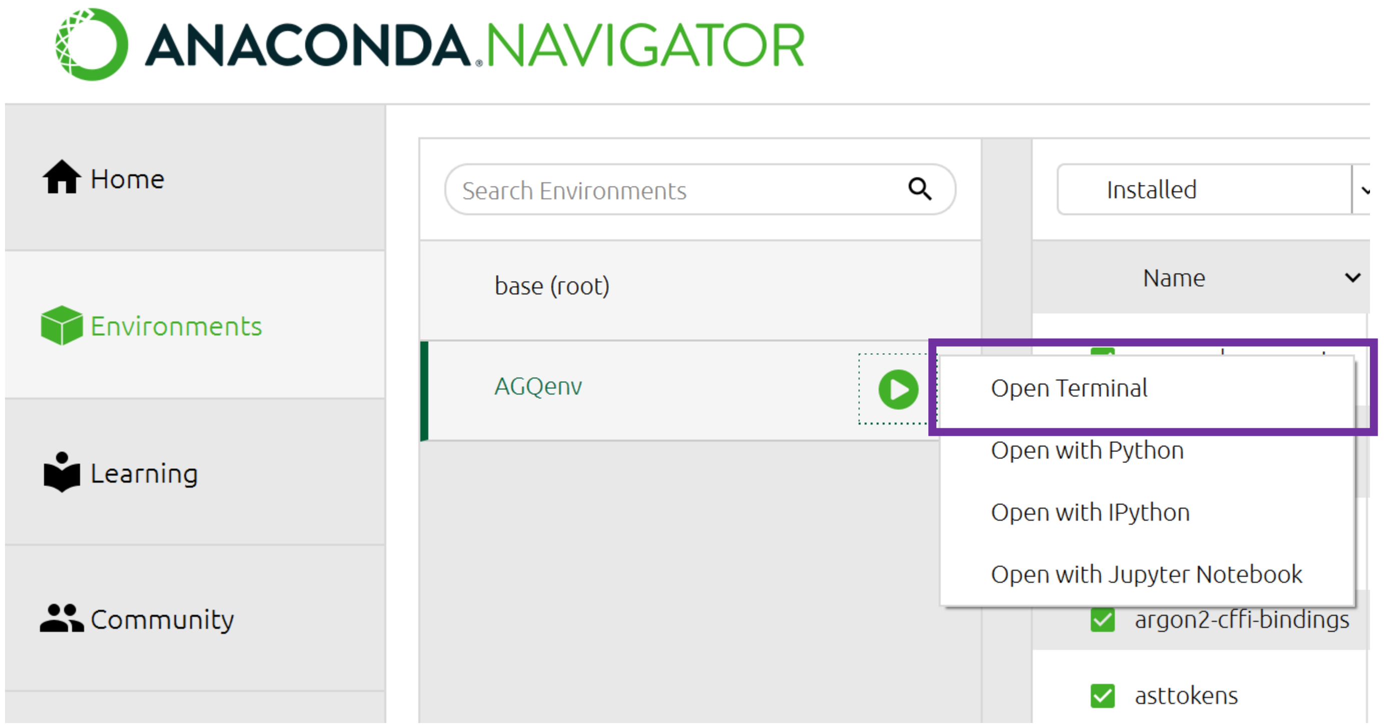
Lecture 1: Workshop

- **Zero;** Getting Setup with Anaconda
- **First;** Getting Started with VirtualEnvs
- **Second;** Using git and some advanced programming concepts
- **Third;** Working with numpy and pandas in jupyter-notebooks
- **Fourth;** More fun with jupyter-notebooks

Setting up Anaconda env

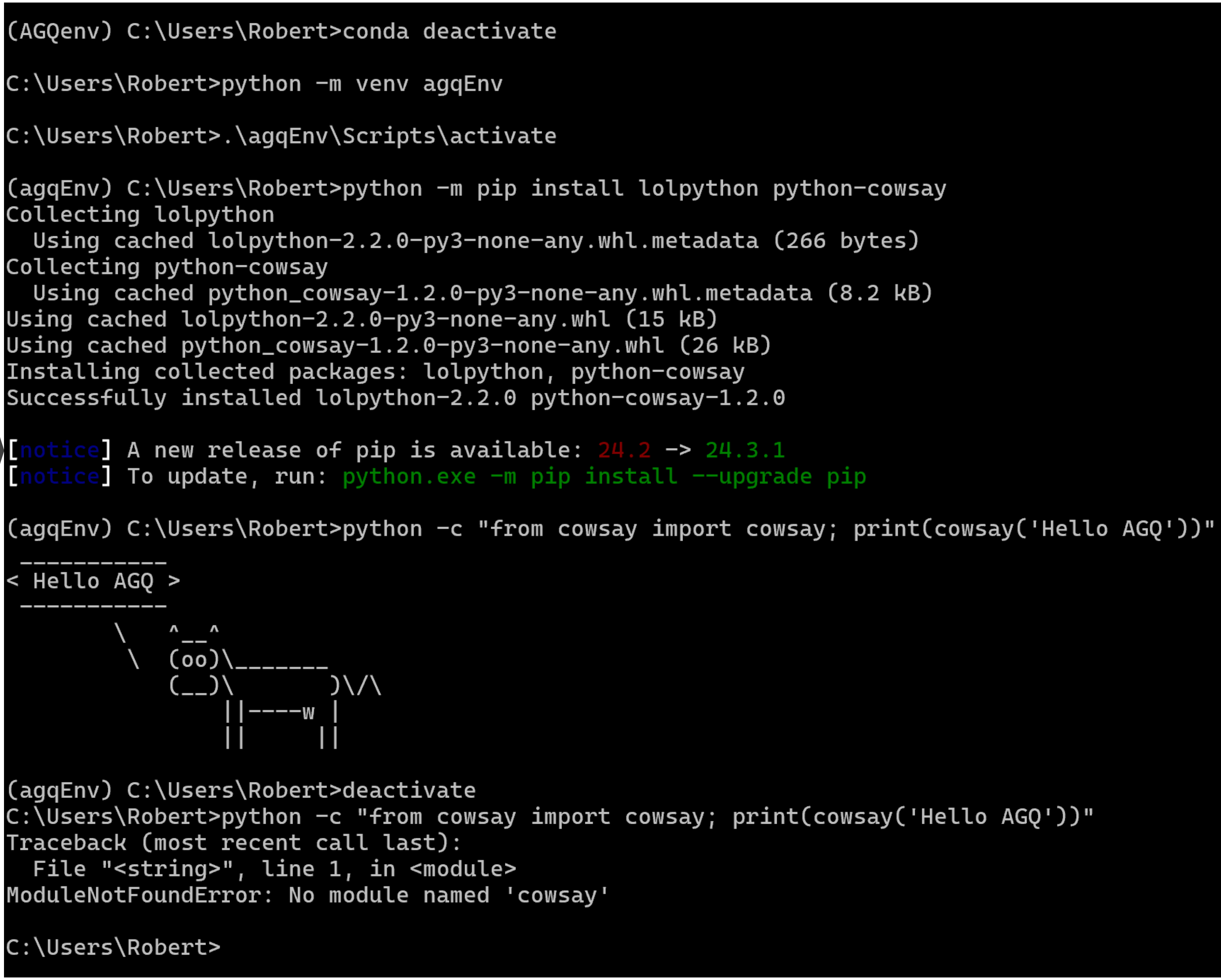
- If anyone has had problems setting up the anaconda env please speak up now and we can help.

- First part won't be using anything too advanced in anaconda/jupyter



VirtualEnvs example

(Windows here!)



```
(AGQenv) C:\Users\Robert>conda deactivate

C:\Users\Robert>python -m venv agqEnv

C:\Users\Robert>.\agqEnv\Scripts\activate

(agqEnv) C:\Users\Robert>python -m pip install lolpython python-cowsay
Collecting lolpython
  Using cached lolpython-2.2.0-py3-none-any.whl.metadata (266 bytes)
Collecting python-cowsay
  Using cached python_cowsay-1.2.0-py3-none-any.whl.metadata (8.2 kB)
Using cached lolpython-2.2.0-py3-none-any.whl (15 kB)
Using cached python_cowsay-1.2.0-py3-none-any.whl (26 kB)
Installing collected packages: lolpython, python-cowsay
Successfully installed lolpython-2.2.0 python-cowsay-1.2.0

[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(agqEnv) C:\Users\Robert>python -c "from cowsay import cowsay; print(cowsay('Hello AGQ'))"
 ___________
< Hello AGQ >
 -----------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

(agqEnv) C:\Users\Robert>deactivate
C:\Users\Robert>python -c "from cowsay import cowsay; print(cowsay('Hello AGQ'))"
Traceback (most recent call last):
  File "<string>", line 1, in <module>
ModuleNotFoundError: No module named 'cowsay'

C:\Users\Robert>
```

Python3 VirtualEnvs

- I would recommend becoming familiar with how virtualenvs work.

- By convention a good directory name for a virtualenv is something like:
  “env” “.env” “environment”

- This indicates what the folder likely contains.

- ***In the case of Python projects in git repos, git is configured to ignore these folders.***

Python3 + Git(hub) exercise

- If people want to brush up on Python3, learn to write some new classes, learn how to handle exceptions etc. I recommend the following:

1. Login to Github
2. Fork this repo https://github.com/rob-c/farm
   (I’ll demo this in a minute)
3. Clone your fork and make changes,
   (add a new animal, add/fix equipment, …)
4. Push your changes to your github repo
5. Make a Pull Request against the original repo from your repo

Ideally, commit your changes in a local branch and push that branch back to github(!)

Python3 examples

- https://github.com/rob-c/farm contains instructions on how to get setup with this example project.

- I would recommend playing around and if you're interested in learning how to interact with github better submit your code and make a pull-request ☺

- The GitHub web-ui is similar, but not the same to bitbucket or gitlab for instance, but they all tend to have similar functionality.

Jupyter-Notebook examples

- If you want to spend some time familiarizing yourself with Jupyter notebooks there is a jupyter-notebook "problem" set.
- This is intended to give you a chance to play with different approaches to using numerical minimization to find a minima.
- There is also a jupyter-notebook exploring how to use pandas and numpy for data manipulation.
  I would recommend reading through this and make sure the examples make sense.

  If they don't please as I'm happy to help ☺