Introduction to coding with python™

Workshop 1 – 08-09-2023

Swinda Falkena – s.k.j.falkena@uu.nl – BBG 604
Institute of Marine and Atmospheric Research

Today

- What is Python and when do you use it?
- Intro to coding in Python: Notebooks, celltypes, calculations, printing, loops, functions and how to deal with errors

python™

- A programming language
- Created in the early 90s by Guido van Rossum (UvA), his goals were:
  - Open source, everyone can contribute
  - Easy to read and intuitive

- In 2018, third most popular language (after Java and JavaScript)
- Now used world wide for: plotting, data analysis, modelling and other complex computations

Who uses Python?

Google

Spotify

WIKIPEDIA
The Free Encyclopedia

reddit

amazon

Who uses Python?

Google

Spotify

NASA

CERN

WIKIPEDIA
The Free Encyclopedia

Institute for
Marine and Atmospheric
research Utrecht

reddit

amazon

And many more universities, companies, etc …

Now we get started!

- No experience at all?

datacamp

"Duolingo" for Python

Now we get started!

- No experience at all?

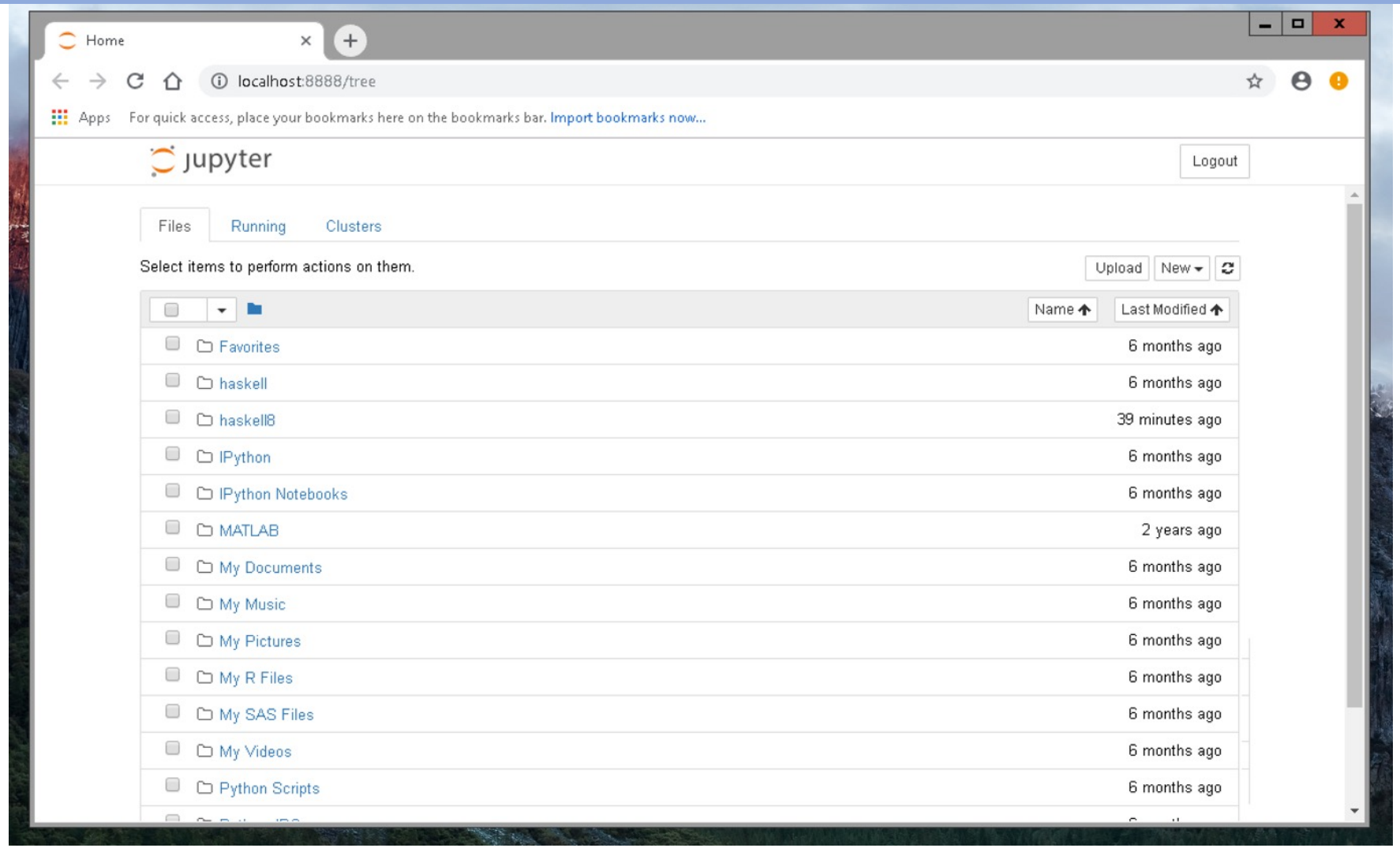
datacamp

"Duolingo" for Python

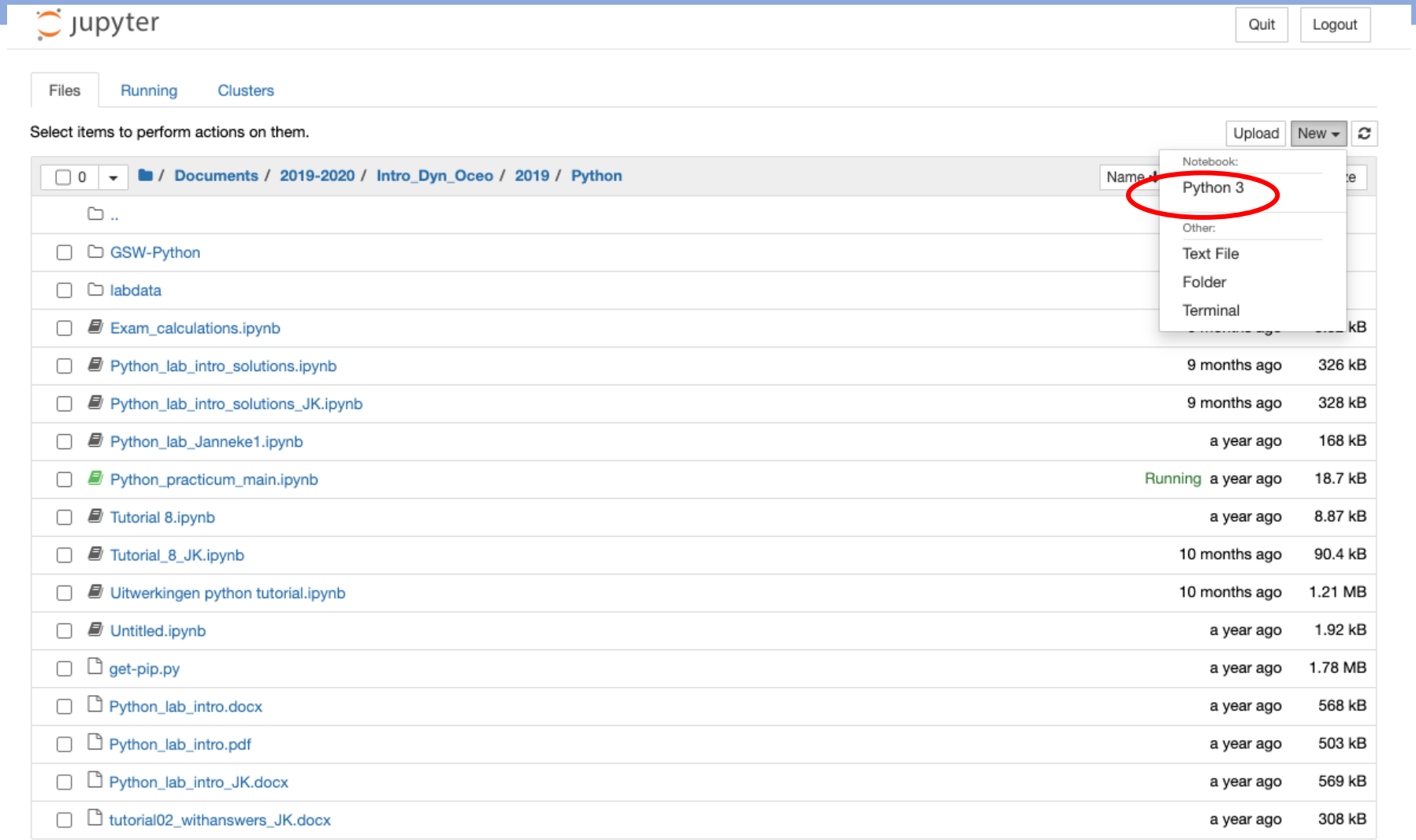
- Has everyone installed Anaconda?

- If yes ☺ then launch Jupyter Notebook

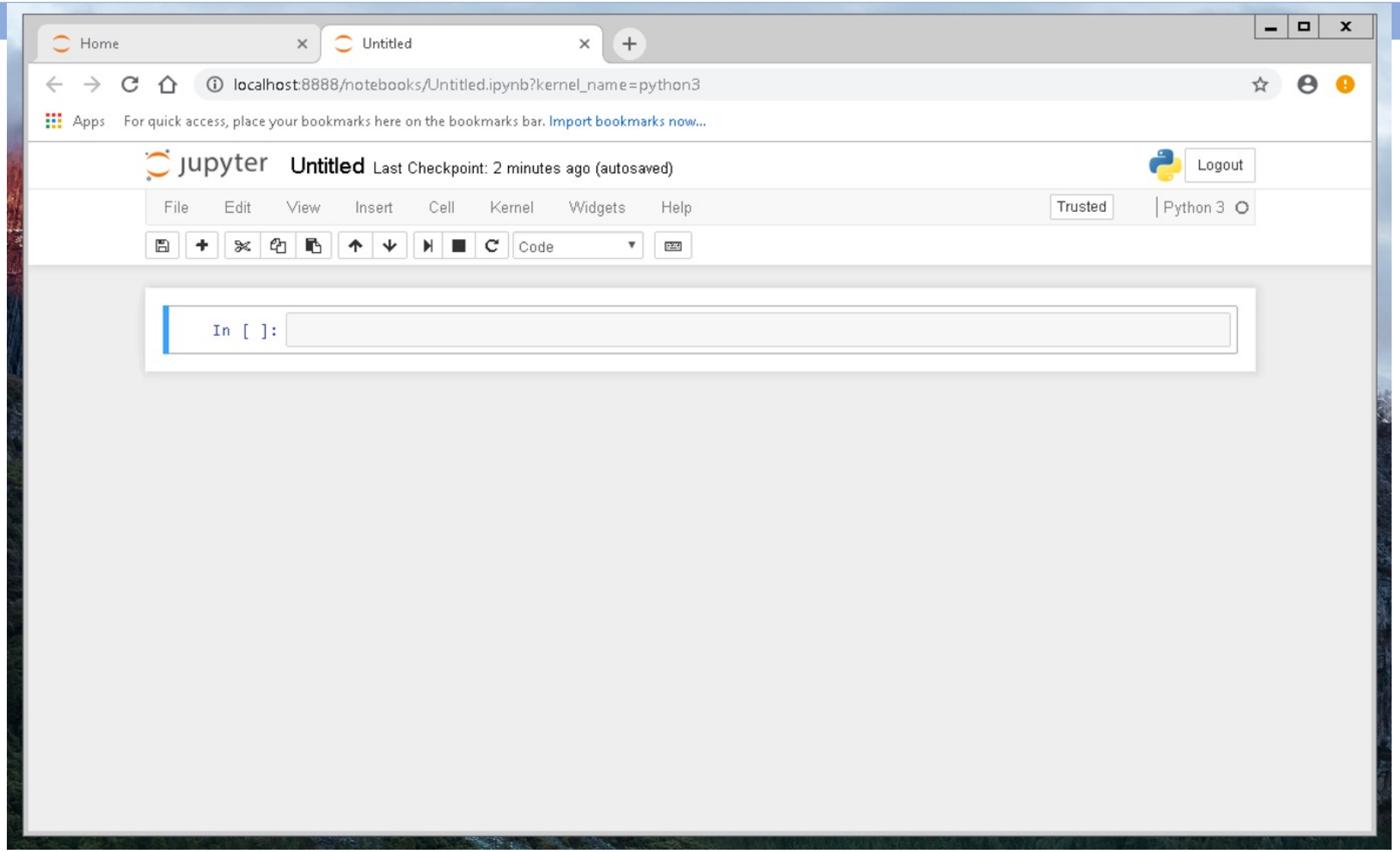
1. Launch Jupyter Notebook



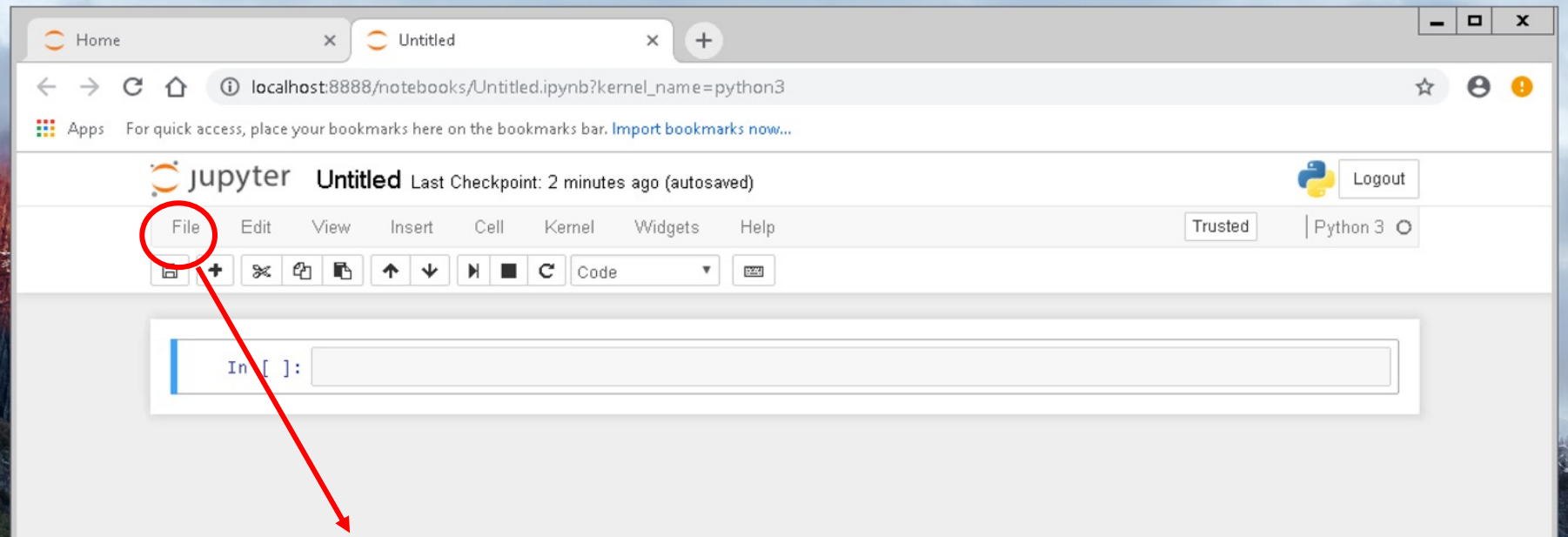
2. Create a folder for this course where you save all your notebooks



3. Open a new notebook



4. Saving a Notebook



Always save your Notebooks before closing!

Go to with '*File -> Save and Checkpoint*'.

The documents are saved as files with *.ipynb* extension.

5. Modes

- Keyboard does different things depending on which mode the Notebook is in

5. Modes

- Keyboard does different things depending on which mode the Notebook is in
- **Edit mode**: type into the cells like a normal text editor and you see a blinking cursor (cells are green).

5. Modes

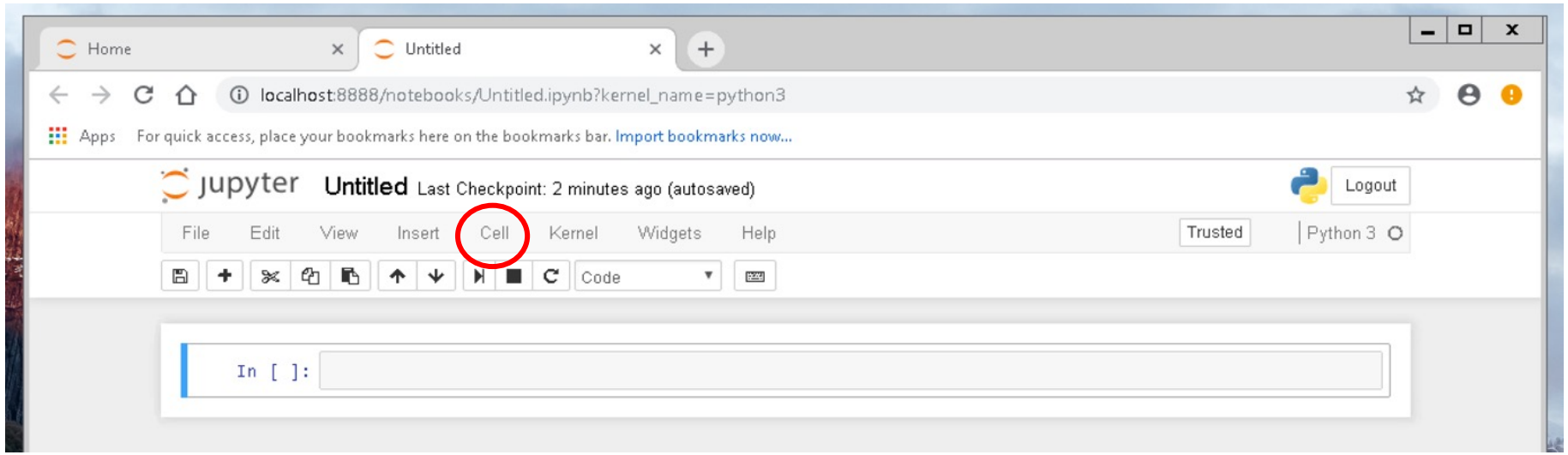
- Keyboard does different things depending on which mode the Notebook is in
- **Edit mode**: type into the cells like a normal text editor and you see a blinking cursor (cells are green).
- **Command mode:** edit the notebook as a whole, but not type into individual cells (cells are blue).

5. Modes

- Keyboard does different things depending on which mode the Notebook is in
- **Edit mode**: type into the cells like a normal text editor and you see a blinking cursor (cells are green).
- **Command mode:** edit the notebook as a whole, but not type into individual cells (cells are blue).
- `Esc`: from edit to command
- `Enter` or double click: command to edit

6. Cell type

- Three types of cells:
  1. **Markdown cells**: cells in which you write text (Latex included)
  2. **Code cells:** cells in which your interactive code is written
  3. **Raw code**: if you want to transfer to another language later



7. Cell shortcuts

- `Shift + Enter:` evaluate a cell and move to the next
- `Ctrl + Enter:` evaluate a cell and stay in the current one
- `a/b:` create a new cell above/below the current
- `d+d:` (press d twice) delete a cell
- `z:` undo
- `m/r/y:` changes cell to markdown/raw/code respectively

8. Coding!

- Using Python as a calculator
- Printing of results
- Lists and indexing
- If-, while- and for loops
- Creating functions
- Comments in your code
- Dealing with errors

Python as a calculator

- You can directly “ask” Python to do calculations:

  `3*4+6/8` `shift+return` to run cell

- By default *only the result of the last line in your cell* is shown after evaluation

Printing of results

- You can print strings or variables with the command `print`:

```
print('Hello, world!’)
var = 3
print(var)
```

- To print strings and numerical variables together: concatenate (“glue”) them with a comma,

```
print('The new variable is equal to', var)
```

- or with +, after having transformed the variables into string with `str`

```
print('The new variable is equal to ' + str(var))
```

Lists

```
l = ['dog', 'cat', 'mouse’, 'fish’]
```

Lists

```
l = ['dog', 'cat', 'mouse’, 'fish’]
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Lists

```
l = ['dog', 'cat', 'mouse’, 'fish’]
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing = selecting a subset of the values in this vector.

Note that vector indexing in Python starts at 0!

Lists

```
l = ['dog', 'cat', 'mouse’, 'fish’]
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing = selecting a subset of the values in this vector.

Note that vector indexing in Python starts at 0!

```
print(l[0])
     ‘dog’
```

Lists

```
l = ['dog', 'cat', 'mouse', 'fish']
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing = selecting a subset of the values in this vector.

Note that vector indexing in Python starts at 0!

```
print(l[0])
    'dog'
```

```
print(r[0])
    0
```

Lists

```
l = ['dog', 'cat', 'mouse', 'fish']
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing = selecting a subset of the values in this vector.

Note that vector indexing in Python starts at 0!

```
print(l[0])
    'dog'
print(l[2])
    'mouse'
```

```
print(r[0])
    0
```

Lists

```
l = ['dog', 'cat', 'mouse', 'fish']
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing = selecting a subset of the values in this vector.

Note that vector indexing in Python starts at 0!

```
print(l[0])
    'dog'
print(l[2])
    'mouse'
```

```
print(r[0])
    0
print(r[2])
    2
```

Lists

```
l = ['dog', 'cat', 'mouse’, 'fish’]
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing = selecting a subset of the values in this vector.

Note that vector indexing in Python starts at 0!

```
print(l[-1])
     ‘fish’
```

Lists

```
l = ['dog', 'cat', 'mouse', 'fish']
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing = selecting a subset of the values in this vector.

Note that vector indexing in Python starts at 0!

```
print(l[-1])
     'fish'
```

```
print(r[-1])
     9
```

Lists

```
l = ['dog', 'cat', 'mouse', 'fish']
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing = selecting a subset of the values in this vector.

Note that vector indexing in Python starts at 0!

```
print(l[-1])
     'fish'
print(l[-2])
     'mouse'
```

```
print(r[-1])
     9
print(r[-2])
     8
```

Lists

```
l = ['dog', 'cat', 'mouse’, 'fish’]
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing = selecting a subset of the values in this vector.

Note that vector indexing in Python starts at 0!

```
print(l[1:-1])
     [‘cat’, ’mouse’]
```

Lists

```
l = ['dog', 'cat', 'mouse’, 'fish’]
r = list(range(10))
r = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Indexing = selecting a subset of the values in this vector.

Note that vector indexing in Python starts at 0!

```
print(l[1:-1])
     [‘cat’, ’mouse’]
print(r[2:-2])
     [2,3,4,5,6,7]
```

If-, while- and for-loops (1)

if, while and for-loops → work on indentation level
indentation → at least four spaces (or a tab)

`if`**–loop** = do some operations, only when a certain condition is met

If-, while- and for-loops (2)

if, while and for-loops → work on indentation level
indentation → at least four spaces (or a tab)

`if`–loop = do some operations, only when a certain condition is met

```
a = 10
if a*2 == 20:
    print('a is equal to 10')
else:
    print('a is not equal to 10')
```

If-, while- and for-loops (3)

`For-` and `while-`loops = repeat a chunk of code either a predetermined number of times, or as long as a certain condition is satisfied.

If-, while- and for-loops (4)

`For-` and `while-`loops = repeat a chunk of code either a predetermined number of times, or as long as a certain condition is satisfied.

```
print('Example for-loop’)
for x in range(5):
    print(x, x**2)

print('Example while-loop’)
x = 0
while x < 2:
    print(x, x**2)
    x += 2/5
```

x=x+2/5

If-, while- and for-loops (5)

`For-` and `while-loops` = repeat a chunk of code either a predetermined number of times, or as long as a certain condition is satisfied.

```
print('Example for-loop')
for x in range(5):
    print(x, x**2)

print('Example while-loop')
x = 0
while x < 2:
    print(x, x**2)
    x += 2/5
```

x=x+2/5

Tips:

- Try writing down what your code should do in regular English
- With the loops: start small to test your code. If you need to loop through a lot of data, first test your code on a small number of data points, and preferably points you can calculate by hand what results your code should give.

Creating functions

- Python works with functions and variables.
  - **Variables:** numbers used as input
  - **Functions:** easily repeat the same calculation over different sets of variables in the same Notebook.

- You can define new functions using the word `def`.

Creating functions

- Python works with functions and variables.
  - **Variables:** numbers used as input
  - **Functions:** easily repeat the same calculation over different sets of variables in the same Notebook.

- You can define new functions using the word `def`.

- As an example, the function on the next slide calculates the Richardson number *Ri,* based on:
  - the zonal velocity at two points
  - with a certain distance in the vertical direction, small depth increment *dz*
  - the buoyancy frequency *N*

Determines the stability of a stratified shear flow

Example of a function

Name of the function

```
def richardson_num(u1, u2, dz, N):
    '''
    Function to calculate the Richardson number

    u1: zonal velocity at the lower location in m/s
    u2: zonal velocity at the upper location in m/s
    dz: depth difference between the two locations in m
    N: buoyancy frequency in 1/s
    '''

    Ri = N**2/((u2-u1)/dz)**2


    return Ri


Ri = richardson_num(u1=0.6, u2=0.2, dz=30, N=0.004)
print(Ri)
```

Example of a function

Variables used as input

```
def richardson_num(u1, u2, dz, N):
    '''
    Function to calculate the Richardson number

    u1: zonal velocity at the lower location in m/s
    u2: zonal velocity at the upper location in m/s
    dz: depth difference between the two locations in m
    N: buoyancy frequency in 1/s
    '''

    Ri = N**2/((u2-u1)/dz)**2


    return Ri


Ri = richardson_num(u1=0.6, u2=0.2, dz=30, N=0.004)
print(Ri)
```

Example of a function

```
def richardson_num(u1, u2, dz, N):
    '''
    Function to calculate the Richardson number

    u1: zonal velocity at the lower location in m/s
    u2: zonal velocity at the upper location in m/s
    dz: depth difference between the two locations in m
    N: buoyancy frequency in 1/s
    '''
    Ri = N**2/((u2-u1)/dz)**2


    return Ri


Ri = richardson_num(u1=0.6, u2=0.2, dz=30, N=0.004)
print(Ri)
```

The function is indented (tab, or 4 spaces)! This is very important, because this tells Python where the function begins and ends!

Example of a function

```
def richardson_num(u1, u2, dz, N):
    '''
    Function to calculate the Richardson number

    u1: zonal velocity at the lower location in m/s
    u2: zonal velocity at the upper location in m/s
    dz: depth difference between the two locations in m
    N: buoyancy frequency in 1/s
    '''
    Ri = N**2/((u2-u1)/dz)**2


    return Ri


Ri = richardson_num(u1=0.6, u2=0.2, dz=30, N=0.004)
print(Ri)
```

Description of what the function does, must be written between:
‘’’
…
‘’’

Example of a function

```
def richardson_num(u1, u2, dz, N):
    '''
    Function to calculate the Richardson number

    u1: zonal velocity at the lower location in m/s
    u2: zonal velocity at the upper location in m/s
    dz: depth difference between the two locations in m
    N: buoyancy frequency in 1/s
    '''
    Ri = N**2/((u2-u1)/dz)**2


    return Ri
```

If you type `help(Richardson_num)`, this text will appear

```
Ri = richardson_num(u1=0.6, u2=0.2, dz=30, N=0.004)
print(Ri)
```

Example of a function

```
def richardson_num(u1, u2, dz, N):
    '''
    Function to calculate the Richardson number

    u1: zonal velocity at the lower location in m/s
    u2: zonal velocity at the upper location in m/s
    dz: depth difference between the two locations in m
    N: buoyancy frequency in 1/s
    '''
    Ri = N**2/((u2-u1)/dz)**2

    return Ri

Ri = richardson_num(u1=0.6, u2=0.2, dz=30, N=0.004)
print(Ri)
```

The actual calculation that happens within in this function

Example of a function

```
def richardson_num(u1, u2, dz, N):
    '''
    Function to calculate the Richardson number

    u1: zonal velocity at the lower location in m/s
    u2: zonal velocity at the upper location in m/s
    dz: depth difference between the two locations in m
    N: buoyancy frequency in 1/s
    '''
    Ri = N**2/((u2-u1)/dz)**2

    return Ri

Ri = richardson_num(u1=0.6, u2=0.2, dz=30, N=0.004)
print(Ri)
```

** in Python means ^ (to the power)

The actual calculation that happens within in this function

Example of a function

```
def richardson_num(u1, u2, dz, N):
    '''
    Function to calculate the Richardson number

    u1: zonal velocity at the lower location in m/s
    u2: zonal velocity at the upper location in m/s
    dz: depth difference between the two locations in m
    N: buoyancy frequency in 1/s
    '''

    Ri = N**2/((u2-u1)/dz)**2

    return Ri
```

The output of the function

```
Ri = richardson_num(u1=0.6, u2=0.2, dz=30, N=0.004)
print(Ri)
```

Example of a function

```
def richardson_num(u1, u2, dz, N):
    '''
    Function to calculate the Richardson number

    u1: zonal velocity at the lower location in m/s
    u2: zonal velocity at the upper location in m/s
    dz: depth difference between the two locations in m
    N: buoyancy frequency in 1/s
    '''

    Ri = N**2/((u2-u1)/dz)**2


    return Ri


Ri = richardson_num(u1=0.6, u2=0.2, dz=30, N=0.004)
print(Ri)
```

Call the function to perform this calculation over a set of variables

Comments in your code

- Put comments in your code to explain what the code does:
  - helps others to understand your code
  - helps you to understand your own code, for example:
    - when you haven't looked at it for a while
    - if you are trying to track down errors

- Comments: type a # before the text
- Python will skip this text when running a cell

Dealing with errors

At some point, you will get error messages.. Don’t panic, here is what you should do:

Dealing with errors

At some point, you will get error messages.. Don’t panic, here is what you should do:

- Try to understand the error written below your cell and in which line it occurs. Often the most relevant information of an error message is at the bottom of the message.

Dealing with errors

At some point, you will get error messages.. Don’t panic, here is what you should do:

- Try to understand the error written below your cell and in which line it occurs. Often the most relevant information of an error message is at the bottom of the message.
- If it’s not exactly clear where the error occurs: simplify your code and add the other parts piece by piece. This building up process is generally considered good practice while coding.

Dealing with errors

At some point, you will get error messages.. Don’t panic, here is what you should do:

- Try to understand the error written below your cell and in which line it occurs. Often the most relevant information of an error message is at the bottom of the message.
- If it’s not exactly clear where the error occurs: simplify your code and add the other parts piece by piece. This building up process is generally considered good practice while coding.
- Do a search online: if you run into a problem, it is very likely that someone else experienced the same before you.

Dealing with errors

At some point, you will get error messages.. Don’t panic, here is what you should do:

- Try to understand the error written below your cell and in which line it occurs. Often the most relevant information of an error message is at the bottom of the message.
- If it’s not exactly clear where the error occurs: simplify your code and add the other parts piece by piece. This building up process is generally considered good practice while coding.
- Do a search online: if you run into a problem, it is very likely that someone else experienced the same before you.

Don’t ask for my help if you haven’t looked the error up online!

Online resources

- You don’t always need to reinvent the wheel. There is a lot of code online, code in these workshops, code in assignments, etc. you can (re)use.

But (!):

- Always try to understand what the code does and how it works, and if it’s correct.
- Give credit if required (copyright).

Let's code!

- The material is on the ACCP Blackboard

- Workshop 1a.ipynb
- Workshop 1b.ipynb

- Also fine if you want to work on ACCP/ice and climate Python exercises!