

Welcome! Please do the following...

1. Sit close to the front, and in pairs
2. If you have not done the prerequisites, do them now (bottom of page at [1])
3. Use git to pull the most recent version of the repository (see instructions on bottom of page at [1])
4. Open an Anaconda Prompt and navigate to cloned repo

silent mode,
please!



[1] <https://gitlab.windenergy.dtu.dk/python-at-risoe/scientific-python-workshops/2-getting-started>

Today's objective is to...

help Matlab-experienced users begin coding in Python

- Three main packages:
 1. NumPy (arrays)
 2. Matplotlib (plotting)
 3. Pandas (REALLY cool arrays, also SQL queries and plotting)

Today's objective is to...

help Matlab-experienced users begin coding in Python

- Three main packages:
 1. NumPy (arrays)
 2. Matplotlib (plotting)
 3. Pandas (REALLY cool a

THERE ARE MANY MORE PACKAGES!

- Numba (optimized, compiled Python)
- xarray (Pandas for multiple dimensions)
- SciPy (scientific Python)
- plotly (online/interactive plotting)
- scikit-learn (machine learning)
- the list goes on...

This is going to be extremely fast, so here are more resources

- **EXCELLENT**, detailed notebooks with Jupyter/NumPy/Matplotlib/debugging from Nicolai Riis (DTU Compute)
 - [Link to repository](#)
 - Go through these later as a “refresher”, get more background
- Various “NumPy for Matlab” references
 - Jenni’s PDF on Workshop GitLab repository: [link](#)
 - Article from Numpy: [link](#)
 - Detailed article: [link](#)
- Matplotlib tutorials
 - Simple tutorial: [link](#)
 - Style gallery: [link](#)
 - Customizing with style sheets: [link](#)
- Pandas tutorials
 - Detailed notebooks by Marianne from Advanced Scientific Python Summer School: [link](#)

This is going to be extremely fast, so here are more resources

- **EXCELLENT**, detailed notebooks with Jupyter/NumPy/Matplotlib/debugging from Nicolai Riis (DTU Compute)
 - [Link to repository](#)
 - Go through these later as a “refresher”, get more background
- Various “NumPy for Matlab” references
 - Jenni’s PDF on Workshop C
 - Article from Numpy: [link](#)
 - Detailed article: [link](#)
- Matplotlib tutorials
 - Simple tutorial: [link](#)
 - Style gallery: [link](#)
 - Customizing with style sheets: [link](#)
- Pandas tutorials
 - Detailed notebooks by Marianne from Advanced Scientific Python Summer School: [link](#)

SERIOUSLY. USE THESE.

Workshop outline

10:00 – 10:05	Preliminaries
10:05 – 10:25	Importing Modules
10:25 – 10:50	NumPy Tutorial/Exercises
10:50 – 11:00	Break
11:00 – 11:25	Matplotlib Tutorial/Exercises
11:25 – 11:50	Pandas Tutorial/Exercises
11:50 – 12:00	Wrap-Up

Workshop outline

10:00 – 10:05	Preliminaries
10:05 – 10:25	Importing Modules
10:25 – 10:50	NumPy Tutorial/Exercises
10:50 – 11:00	Break
11:00 – 11:25	Matplotlib Tutorial/Exercises
11:25 – 11:50	Pandas Tutorial/Exercises
11:50 – 12:00	Wrap-Up

First question: What is a module/package/library?

- **Module:**

- A .py file with variables, functions, classes, etc. that you intend to re-use
- Must be local

- **Package:**

- A directory (folder) of modules with a file called “__init__.py”
- Can be local or remote (conda, pip, etc.)

- **Library:**

- No precise definition in Python
- “Standard library”: syntax, modules, etc. that come bundled with your Python installation

There are two basic ways to import a Python module/package*

METHOD 1

```
import $name [as $short_name]
```

Examples:

```
import numpy  
x = numpy.arange(4)
```

```
import numpy as np  
x = np.arange(4)
```

METHOD 2

```
from $name import $thing
```

Examples:

```
from numpy import arange  
x = arange(4)
```

```
from numpy import *  
x = arange(4)
```

* Assuming it is installed in your Python environment!
Remember Workshop 1 ;)

There are two basic ways to import a Python module/package*

METHOD 1

```
import $name [as $short_name]
```

Examples:

```
import numpy
x = numpy.arange(4)
```

```
import numpy as np
x = np.arange(4)
```

METHOD 2

```
from $name import $thing
```

Note that if you are importing just a module (not an installed package), you must either:

1. be in the same directory as the file or
2. have the module in your python path

```
from numpy import *
```

```
x = arange(4)
```

* Assuming it is installed in your Python environment!
Remember Workshop 1 ;)

There are two basic ways to import a Python module/package*

METHOD 1

```
import $name [as $short_name]
```

Example

```
import numpy
x = numpy.arange(4)
```

Generally preferred (shorter name, but still clear what functions are from which modules)

```
import numpy as np
x = np.arange(4)
```

METHOD 2

```
from $name import $thing
```

Example

```
from numpy import arange
x = arange(4)
```

Generally not preferred (not a hard rule).

If you do it, `$thing` should be a module (not a function or variable).

```
from numpy import arange
x = arange(4)
```

More details on imports [here](#)

* Assuming it is installed in your Python environment!
Remember Workshop 1 ;)

What's going on under the hood: magic

1. Check if module is already loaded into registry
 - If yes, reuse
 - If no, continue to step 2
2. Create empty module
3. Insert module into module registry
4. Load module core
 - May compile code if necessary (`__pycache__`/)
5. Execute code in namespace of new module

Steps shamelessly stolen from here: <http://effbot.org/zone/import-confusion.htm#many-ways>

What's going on under the hood: magic

1. Check if module is already loaded into registry
 - If yes, reuse
 - If no, continue to step 2
2. Create empty module
3. Insert module into module registry
4. Load module core
 - May compile code if necessary (`__pycache__`/)
5. Execute code in namespace of new module

`importlib.reload($mod_name)`
to force a reload, but
generally not advised

Steps shamelessly stolen from here: <http://effbot.org/zone/confusion.htm#many-ways>

just means that the things
defined in the module code
will now exist in the imported
module

*Do this with to the
person next to you*



Exercise: importing greetings

*Do this with to the
person next to you*



Exercise: importing greetings

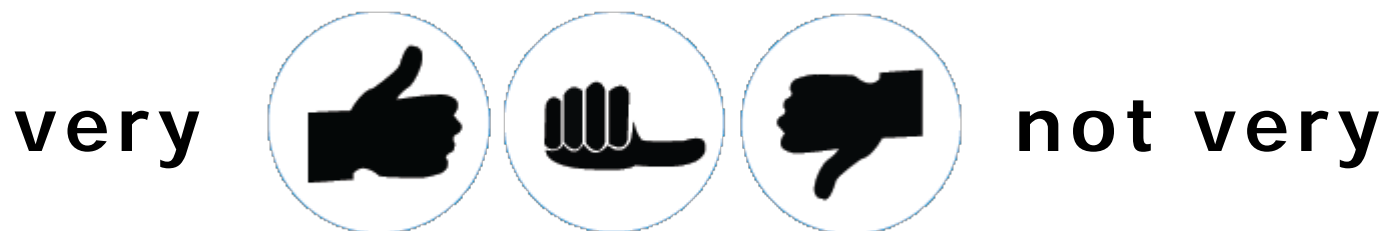
- `materials/hello_world.py` module with function `greetings()` in GitLab repository

Do the following:

1. From an iPython interpreter (either in Spyder or an Anaconda Prompt):
 - a. Import the module with its full name
 - b. Import the module with a short name (you choose)
 - c. Import `greetings()` from the module with its full name
 - d. Import `greetings()` from the module and give the function a short name (you choose)
2. Write a script in a file called `importing_hw.py` that performs the same 4 inputs as in Step 1 and prints the output from 1c and 1d.
3. Restart your kernel*, then run your script from Step 2. Come grab me, and prove to me you have done all 4 imports properly.

** Settings button in iPython console in Spyder -> Restart kernel*

How comfortable do you feel with the topics I just presented?



New: I'm using this to get a global sense of how the lecture is going.



Workshop outline

10:00 – 10:05	Preliminaries
10:05 – 10:25	Importing Modules
10:25 – 10:50	NumPy Tutorial/Exercises
10:50 – 11:00	Break
11:00 – 11:25	Matplotlib Tutorial/Exercises
11:25 – 11:50	Pandas Tutorial/Exercises
11:50 – 12:00	Wrap-Up

NumPy is an optimized Python package designed for numeric array work

- Pre-compiled C code
- Similar syntax to Matlab (some differences)
- Arrays can be ints, strings, floats, complex, even objects

Teaching time – everyone, get to Jupyter!

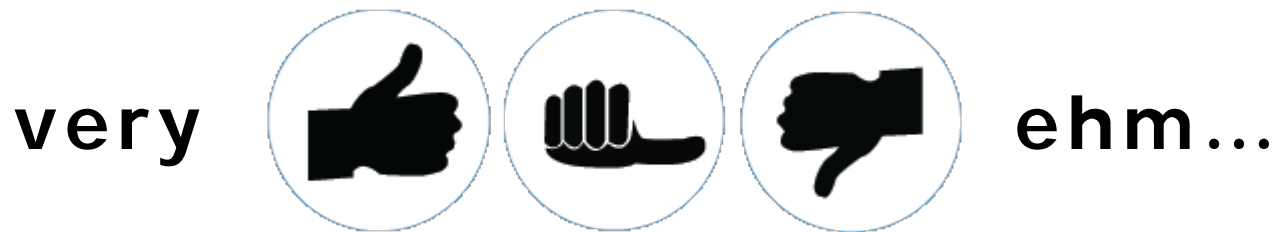
NOTE: This is gonna be superfast (more exercise time). Use notebook as a reference.

- In your Anaconda Prompt, navigate to the cloned GitLab repository
- Enter `jupyter notebook` into the prompt
- In the tab that pops up in your internet browser, click on the numpy tutorial
- `File -> Close and Halt` at the end!

Exercise: correlated random variables in NumPy

- If jupyter isn't already open:
 - In your Anaconda Prompt, navigate to the cloned GitLab repository
 - Enter `jupyter notebook` into the prompt
 - In the tab that pops up in your internet browser, click on the numpy exercise
- Otherwise, just navigate to the Exercise 1 notebook in the open tab in your internet browser

How comfortable do you feel with the topics I just presented?



Workshop outline



10:00 – 10:05	Preliminaries
10:05 – 10:25	Importing Modules
10:25 – 10:50	NumPy Tutorial/Exercises
10:50 – 11:00	Break
11:00 – 11:25	Matplotlib Tutorial/Exercises
11:25 – 11:50	Pandas Tutorial/Exercises
11:50 – 12:00	Wrap-Up



Workshop outline

10:00 – 10:05	Preliminaries
10:05 – 10:25	Importing Modules
10:25 – 10:50	NumPy Tutorial/Exercises
10:50 – 11:00	Break
11:00 – 11:25	Matplotlib Tutorial/Exercises
11:25 – 11:50	Pandas Tutorial/Exercises
11:50 – 12:00	Wrap-Up

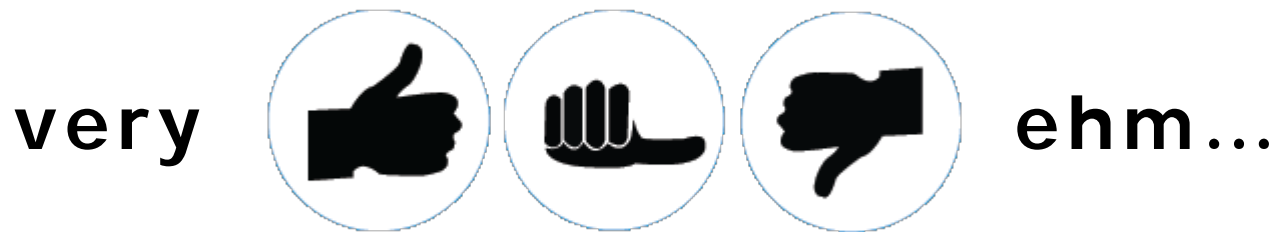
Matplotlib is a package for plotting...like in matlab

- And let's check it out in jupyter!
- Open up the jupyter notebook matplotlib tutorial

Exercise: visualize your random variables

- Open up the jupyter notebook matplotlib exercise
- (This won't be on the repo until after we've gone through the first exercise)

How comfortable do you feel with the topics I just presented?



Workshop outline

10:00 – 10:05	Preliminaries
10:05 – 10:25	Importing Modules
10:25 – 10:50	NumPy Tutorial/Exercises
10:50 – 11:00	Break
11:00 – 11:25	Matplotlib Tutorial/Exercises
11:25 – 11:50	Pandas Tutorial/Exercises
11:50 – 12:00	Wrap-Up

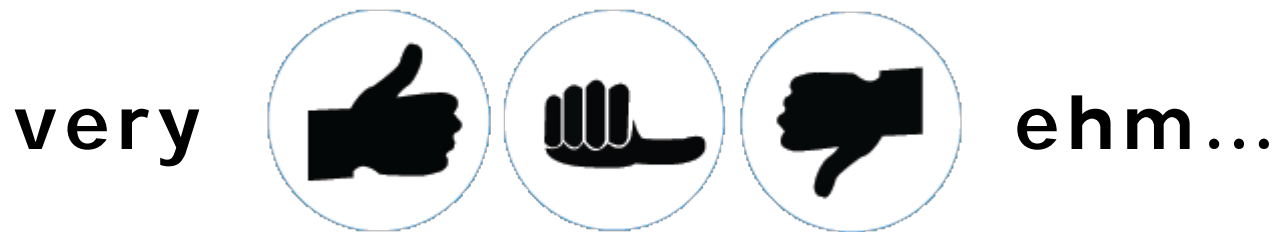
Pandas is a package designed for working with and visualizing datasets

- Particularly powerful at selecting/plotting data
- Let's check it out in jupyter!
- Open up the jupyter notebook with the pandas tutorial

Exercise: selecting and plotting in pandas

- Open up the jupyter notebook for the pandas exercise

How comfortable do you feel with the topics I just presented?



Workshop outline

10:00 – 10:05	Preliminaries
10:05 – 10:25	Importing Modules
10:25 – 10:50	NumPy Tutorial/Exercises
10:50 – 11:00	Break
11:00 – 11:25	Matplotlib Tutorial/Exercises
11:25 – 11:50	Pandas Tutorial/Exercises
11:50 – 12:00	Wrap-Up

Today we learned...

1. What it means/how to import a module/package

```
import numpy as np
np.arange(4)
```

Matrix multiplication

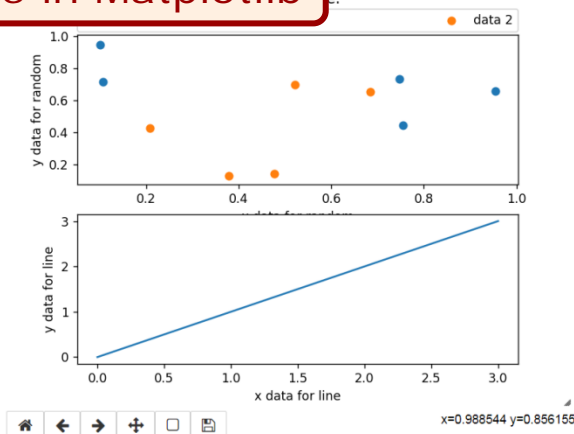
I'm not sure how that's useful to know, you are thinking. I normally do matrix multiplication. V

```
In [ ]: print(my_mat @ v1) # note this returns a 1D array
        print(my_mat @ v2) # ...while this returns a 2D array
        print(np.dot(my_mat, v2)) # you can also use the numpy function
        # print(my_mat @ v3) # this breaks with a 'dimension mismatch' error, jus
```

Well. That was easy.

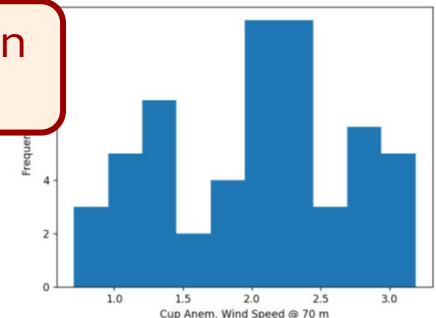
2. Basics in NumPy

3. Basics in Matplotlib



```
ax = met_df[(met_df.Wdir_41m_mean >= 250) & (met_df.Wdir_41m_mean <= 340)].plot(y=
ax.set_xlabel('Cup Anem. Wind Speed @ 70 m')
ax.legend().set_visible(False) # hide the legend since we have an x axis
```

4. Basics in pandas



Looking forward...what's next?

- Please add your name to the sign-up form if you haven't
- Next workshop: **Collaborating with Python**
 - A.k.a., how to package and share code with each other
 - 17. okt 10:00
- Google is your friend
- But so are your colleagues!
 - Python@Risø group on GitLab: [link](#)
 - rink@dtu.dk
- Expect a survey from me – please fill it out!

```
import numpy as np
np.arange(4)
```

Christian Grinderslev
Nikolay Dimitrov
Nicolai Riis

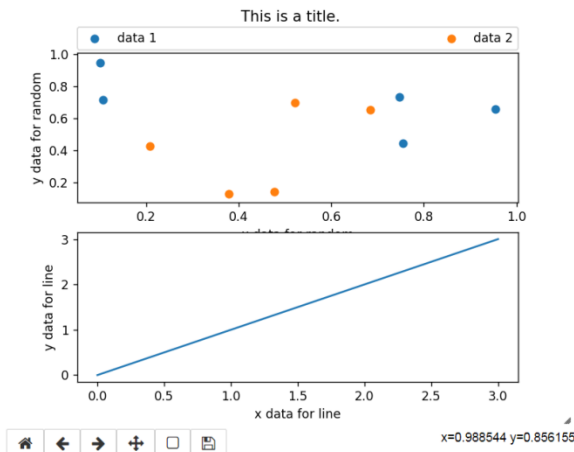
Matrix multiplication

I'm not sure how that's useful to know, you are thinking. I normally do matrix multiplication. V

```
In [ ]: print(my_mat @ v1) # note this returns a 1D array
        print(my_mat @ v2) # ...while this returns a 2D array
        print(np.dot(my_mat, v2)) # you can also use the numpy function
        # print(my_mat @ v3) # this breaks with a 'dimension mismatch' error, jus
```

Well. That was easy.

Thanks!



```
ax = met_df[(met_df.Wdir_41m_mean >= 250) & (met_df.Wdir_41m_mean <= 340)].plot(y=
ax.set_xlabel('Cup Anem. Wind Speed @ 70 m')
ax.legend().set_visible(False) # hide the legend since we have an x axis
```

