Introduction to Python (3.7)

Focus on satellite image processing

Environment: spyder

Jean-Luc Attié, 2020 - 2021

This Web site is better than what I can teach

https://www.w3schools.com/python/

For the moment you don't need python in your computer, but you need internet

http://physique-fac.eklablog.com/

To take some materials, files, doc, etc..
password: ups

This Web site is better than I what can teach

Python Indentation

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
  print("Five is greater than two!")
```

Try it Yourself »

Python will give you an error if you skip the indentation:

Example

Syntax Error:

```
if 5 > 2:
print("Five is greater than two!")
```

Try it Yourself »

This Web site is better than I what can teach

Python Comments

‹ Previous Next ›

Comments can be used to explain Python code.

Comments can be used to make the code more readable.

Comments can be used to prevent execution when testing code.

Creating a Comment

Comments starts with a `#`, and Python will ignore them:

Example

```
#This is a comment
print("Hello, World!")
```

Try it Yourself »

Comments can be placed at the end of a line, and Python will ignore the rest of the line:

This Web site is better than what I can teach

Python Variables

‹ Previous Next ›

Creating Variables

Variables are containers for storing data values.

Unlike other programming languages, Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

Example

```
x = 5
y = "John"
print(x)
print(y)
```

Try it Yourself »

Variables do not need to be declared with any particular *type*, and can even change type after they have been set.

Example

```
x = 4 # x is of type int
x = "Sally" # x is now of type str
print(x)
```

This Web site is better than what I can teach

Python Data Types

‹ Previous Next ›

Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`

This Web site is better than what I can teach

Getting the Data Type

You can get the data type of any object by using the `type()` function:

Example

Print the data type of the variable x:

```
x = 5
print(type(x))
```

Try it Yourself »

This Web site is better than what I can teach

Setting the Data Type

In Python, the data type is set when you assign a value to a variable:

Example	Data Type	Try it
`x = "Hello World"`	str	Try it »
`x = 20`	int	Try it »
`x = 20.5`	float	Try it »
`x = 1j`	complex	Try it »
`x = ["apple", "banana", "cherry"]`	list	Try it »
`x = ("apple", "banana", "cherry")`	tuple	Try it »
`x = range(6)`	range	Try it »
`x = {"name" : "John", "age" : 36}`	dict	Try it »
`x = {"apple", "banana", "cherry"}`	set	Try it »
`x = frozenset({"apple", "banana", "cherry"})`	frozenset	Try it »
`x = True`	bool	Try it »
`x = b"Hello"`	bytes	Try it »
`x = bytearray(5)`	bytearray	Try it »
`x = memoryview(bytes(5))`	memoryview	Try it »

This Web site is better than what I can teach

Setting the Specific Data Type

If you want to specify the data type, you can use the following constructor functions:

Example	Data Type	Try it
`x = str("Hello World")`	str	Try it »
`x = int(20)`	int	Try it »
`x = float(20.5)`	float	Try it »
`x = complex(1j)`	complex	Try it »
`x = list(("apple", "banana", "cherry"))`	list	Try it »
`x = tuple(("apple", "banana", "cherry"))`	tuple	Try it »
`x = range(6)`	range	Try it »
`x = dict(name="John", age=36)`	dict	Try it »
`x = set(("apple", "banana", "cherry"))`	set	Try it »
`x = frozenset(("apple", "banana", "cherry"))`	frozenset	Try it »
`x = bool(5)`	bool	Try it »
`x = bytes(5)`	bytes	Try it »
`x = bytearray(5)`	bytearray	Try it »
`x = memoryview(bytes(5))`	memoryview	Try it »

This Web site is better than what I can teach

Python Operators

Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Python Arithmetic Operators

Arithmetic operators are used with numeric values to perform common mathematical operations:

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

This Web site is better than what I can teach

Python Assignment Operators

Assignment operators are used to assign values to variables:

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

This Web site is better than what I can teach

Python Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

This Web site is better than what I can teach

Python Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same obj
memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

This Web site is better than what I can teach

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered and unindexed. No duplicate members.
- **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

This Web site is better than what I can teach

Python Conditions and If statements

Python supports the usual logical conditions from mathematics:

- Equals: `a == b`
- Not Equals: `a != b`
- Less than: `a < b`
- Less than or equal to: `a <= b`
- Greater than: `a > b`
- Greater than or equal to: `a >= b`

These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the `if` keyword.

Example

If statement:

```
a = 33
b = 200
if b > a:
  print("b is greater than a")
```

Try it Yourself »

This Web site is better than what I can teach

Elif

The `elif` keyword is pythons way of saying "if the previous conditions were not true, then try this condition".

Example

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

Try it Yourself »

In this example `a` is equal to `b`, so the first condition is not true, but the `elif` condition is true, so we print to screen that "a and b are equal".

This Web site is better than what I can teach

Else

The `else` keyword catches anything which isn't caught by the preceding conditions.

Example

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

Try it Yourself »

In this example `a` is greater than `b`, so the first condition is not true, also the `elif` condition is not true, so we go to the `else` condition and print to screen that "a is greater than b".

This Web site is better than what I can teach

Python Loops

Python has two primitive loop commands:

- `while` loops
- `for` loops

This Web site is better than what I can teach

The while Loop

With the `while` loop we can execute a set of statements as long as a condition is true.

Example

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Try it Yourself »

This Web site is better than what I can teach

Python For Loops

A `for` loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the `for` keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

With the `for` loop we can execute a set of statements, once for each item in a list, tuple, set etc.

Example

Print each fruit in a fruit list:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

Try it Yourself »

The `for` loop does not require an indexing variable to set beforehand.

This Web site is better than what I can teach

The range() Function

To loop through a set of code a specified number of times, we can use the `range()` function,

The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Example

Using the range() function:

```
for x in range(6):
  print(x)
```

Try it Yourself »

Note that `range(6)` is not the values of 0 to 6, but the values 0 to 5.

The `range()` function defaults to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: `range(2, 6)`, which means values from 2 to 6 (but not including 6):

Example

Using the start parameter:

```
for x in range(2, 6):
  print(x)
```

This Web site is better than what I can teach

The while Loop

With the `while` loop we can execute a set of statements as long as a condition is true.

Example

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Try it Yourself »

This Web site is better than what I can teach

The while Loop

With the `while` loop we can execute a set of statements as long as a condition is true.

Example

Print i as long as i is less than 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Try it Yourself »

Create a python ‘subroutine’

Very often we declare the subroutines in the beginning of the main code,
but they could be apart, in another file we have to import like a standard library

```
def f(x):
    return sqrt(x)
```

Could be more complicated

```
x = lambda a : a + 10
print(x(5))
```

Simple function

Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (library, ascii, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Content of the lecture

1. **Structure of a Python script**
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (library, ascii, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

1. Structure of a program

1. Description of your program
2. Call of python libraries
3. Definition of variables/constants
4. Eventual reading of external data
5. Definition of eventual subroutines
6. Corpse of your Program
7. Graphic part
8. Saving of eventual figures
9. Saving of eventual data from your program

1. Structure of a script

Example of a code start

```
# -*- coding: utf-8 -*-
"""
Created on Mon Jan 27 17:36:28 2020

@author: attjl
This program is made to plot maps of carbon monoxide
From MOPITT instrument superimposed with fires from
AVHRR
The Unit is ppbv
Input : hdf file for CO from MOPITT web site
        Netcdf files for fire counts
"""
```

DESCRIPTION OF THE PROGRAM
NO EFFECT ON RESULTS

```
import pandas as pd
from functions import land
from functions import plots_fire
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap
import time
import sys
```

EXAMPLE OF USE OF DIVERSE PYTHON LIBRARIES

1. Structure of a script

```
#;-------------------------------------------------------------
print("-----------------------------------------")
print("PROGRAM    MAPPING_IASI JLA  V1.1 APR, 2020")
print("-----------------------------------------")
#;-------------------------------------------------------------
# ------------------------------------------------------------------------
# Data
# ------------------------------------------------------------------------
year, month='2010', '12'  # See path below
version='V1.7'
# Data directory
path_savefig = "/Users/attjl/Desktop/EQUIPE/N2O/FIGURES/"
path='/Users/attjl/Desktop/EQUIPE/N2O/DATA/VERSION1.7/'+year+month+'/'
path_save='/Users/attjl/Desktop/EQUIPE/N2O/DATA/VERSION1.7/Monthly2011/'


# Wind and omega
io=7    #-> Wind and omega field 300 hPa # commence en jan 1948
ko=755+int(month)


|---------------------
# Average the N2O data on lat_grid and lon_grid (can be changed)
# ————————————————————————————————————————
```

Write the instructions of the code

4)

```
g1 = plt.subplot(211)
plt.title(name)
plt.plot(z, psi0, 'red')
plt.plot(z, psi_plus, 'blue')
plt.xlabel("z")
plt.ylabel("psi")


g2 = plt.subplot(212)
plt.plot(z, v_ph, 'blue')
plt.xlabel("z")
plt.ylabel("v_ph")

plt.show()
```

Example of graphics

Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (library, ascii, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Content of the lecture

1. Structure of a Python script
2. **Some examples/exercises**
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (library, ascii, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Calculation of factorial (n)

Simple and very easy script

- Write a script using python that calculates n! (n>=0) in 2 ways:

Ex : 5!=1*2*3*4*5 and 0! = 1

n!=1 x 2 x 3 x 4 x ... x n non recursive method

n!=n x (n-1) x (n-2) x ... x 1 recursive method

Calculation of factorial (n)

Recursive method

$$n!=n \times (n-1) \times (n-2) \times \ldots \times 1$$

```
def factorial(n):
    if n == 1:
        return n
    else:
        return n*factorial(n-1)
# take input from the user
num = int(input("Enter a number: "))
# check is the number is negative
if num < 0:
    print("does not exist for negative numbers")
elif num == 0:
    print(1)
else:
    print("The factorial of",num,"is",recur_factorial(num))
```

Calculation of factorial (n)

Non recursive method

$n! = 1 \times 2 \times 3 \times \ldots \times n$

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

def fact(n):
    """fact(n): calculation of fact. for n
(integer >= 0)"""
    x=1
    for i in range(2,n+1):
        x*=i
    return x
```

Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (library, ascii, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. **Mathematics: Numpy library**
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (library, ascii, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Numpy

Numpy is a python C library for array oriented computing

- Numpy is suited for many applications
- Image processing
- Signal processing
- Linear algebra

Numpy array

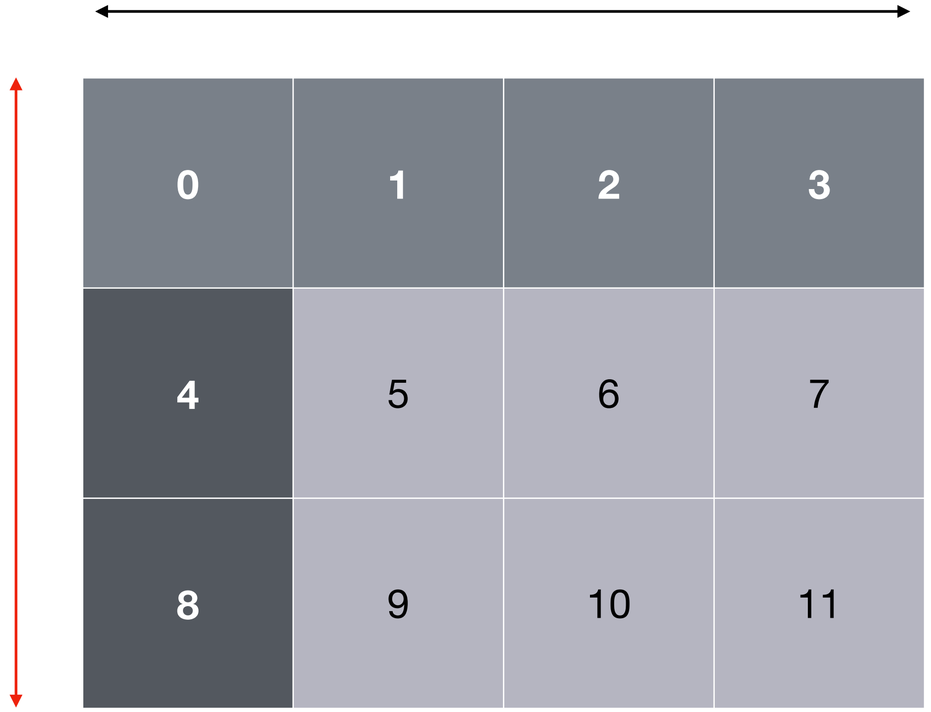
```
import numpy as np

a=np.array([1,2,3,4,5,6,7,8,9])

b=a.reshape((3,3))

b

out[9]:

array([[1, 2, 3],

       [4, 5, 6],

       [7, 8, 9]])

b*10+4

out[10]:

array([[14, 24, 34],

       [44, 54, 64],

       [74, 84, 94]])
```

One dimensional array

Shape (7,)

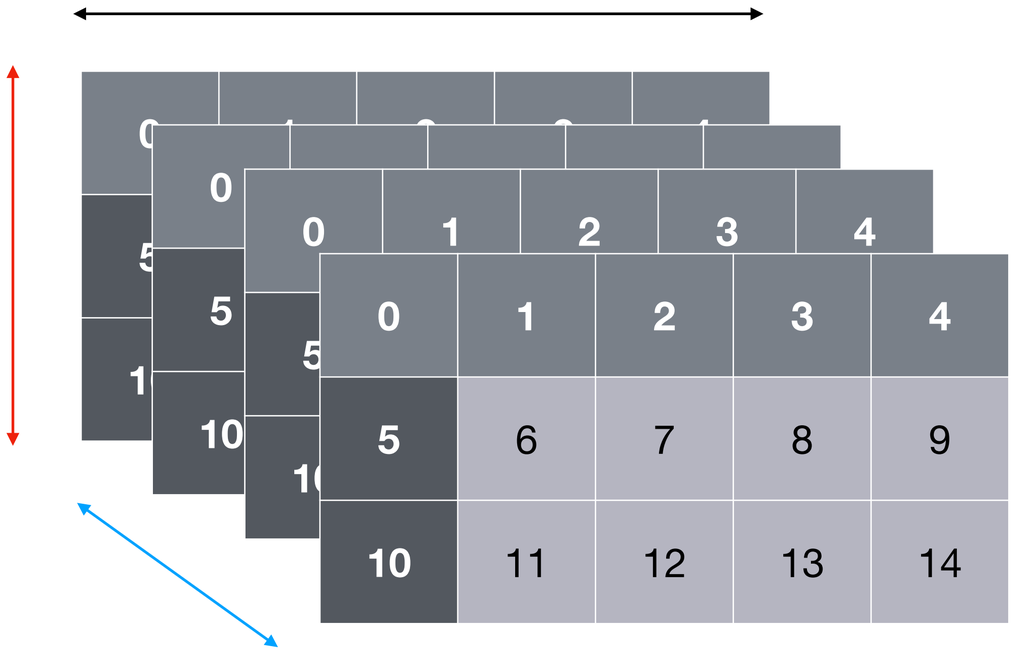
Two dimensional array

Shape (3,4)



Any dimension

Shape : (3,5,4)



Type of the array element

The type object is accessible through dtype

```
b.dtype

Out[12]: dtype('int64')

a.dtype

Out[13]: dtype('int64')

b

Out[14]:

array([[1, 2, 3],

       [4, 5, 6],

       [7, 8, 9]])

b=b/1.0

b.dtype

Out[16]: dtype('float64')
```

Type

You can specify the type

```
a=np.array([1,2,3,4,5,6,7,8,9], dtype=np.float32)

a.dtype

Out[18]: dtype('float32')
```

Use of array

•As a list of values

```
np.array([1,2,3,4,5,6,7,8,9], dtype=np.float32)

Out[21]: array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9.], dtype=float32)
```

•As a range of values

```
np.arange(10)

Out[22]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

•Control the number of values

```
np.linspace(0,1,10)

Out[24]:

array([ 0.        ,  0.11111111,  0.22222222,  0.33333333,  0.44444444,

        0.55555556,  0.66666667,  0.77777778,  0.88888889,  1.        ])
```

Initialization of array

- **Only zero values**

```
np.zeros([3,2])
Out[26]:
array([[ 0.,  0.],
       [ 0.,  0.],
       [ 0.,  0.]])
```

- **Only one values**

```
np.ones([3,2])
Out[27]:
array([[ 1.,  1.],
       [ 1.,  1.],
       [ 1.,  1.]])
```

- **Empty array**

```
np.empty([3,2])
Out[28]:
array([[ 0.,  0.],
       [ 0.,  0.],
   [0., 0.]])
```

Initialization of Array

- **Constant diagonal**

```
np.eye(4)
Out[29]:
array([[ 1.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  1.]])
```

- **Non constant diagonal**

```
np.diag([1,2,3,4])
Out[30]:
array([[1, 0, 0, 0],
       [0, 2, 0, 0],
       [0, 0, 3, 0],
       [0, 0, 0, 4]])
```

Indexing or slicing an array

Arr[0:2,:]

0	1	2	3
4	5	6	7
8	9	10	11

Arr[2:1,:]

- Example

```
print(b)
[[ 1.  2.  3.]
 [ 4.  5.  6.]
 [ 7.  8.  9.]]

b[0:2,1::]
Out[35]:
array([[ 2.,  3.],
       [ 5.,  6.]])
```

Numpy functions

- Comparison: <, <=, ==, !=, >=, >
- arithmetic: +, -, *, /, **
- exponential: exp, expm1, exp2, log, log10, log1p, log2, power, sqrt
- Trigonometric: sin, cos, tan, arcsin, arccos, arctan
- Hyperbolic: sin, tanh, arcsinh, arccosh, arctanh
- Bitwise operations: and, logical_xor, not, or
- Predicate: isinf, isfinite, isnan, signbit
- Other: abs, ceil, floor, mod, mode, round, sinc, sign, trunk

Numpy functions

```
Import numpy as np



print(np.exp(2)).       # exponentielle

print(np.sin(2*np.pi)) # sinus

print(np.cos(2*np.pi)) # cosinus
```

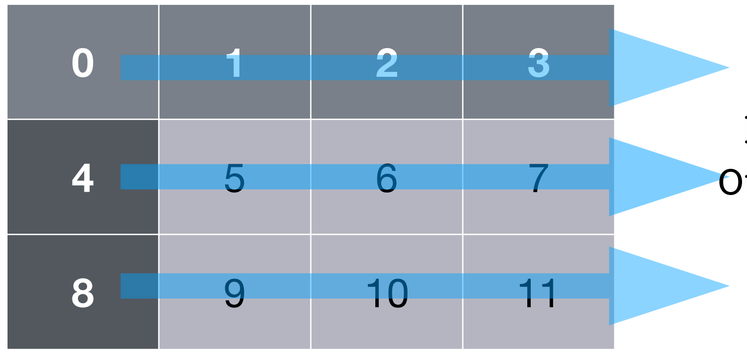
Axis of arrays

0	1	2	3
4	5	6	7
8	9	10	11

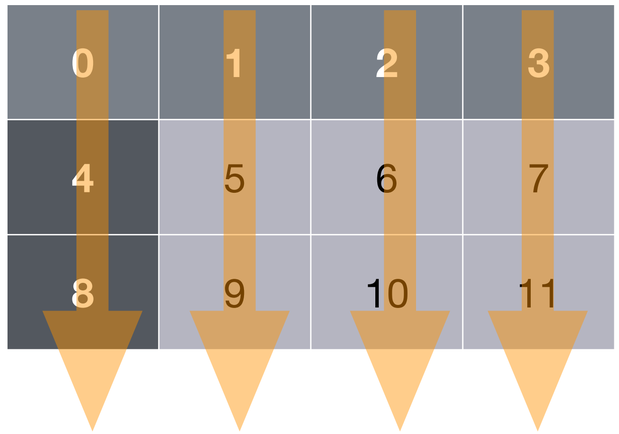
a.sum

Sum of the elements

Axis of arrays




```
b.sum(axis=1)
Out[66]: array([ 6, 22, 38])
```




```
b.sum(axis=0)
Out[65]: array([12, 15, 18, 21])
```

Array methods

Reductions

```
a.mean(), a.argmin(), a.argmax(), a.trace(),
a.cumsum(), a.cumprod()
```

Manipulations

```
a.argsort(), a.transpose(), a.reshape(…),
a.ravel(), a.fill(..), a.clip(…)
```

Complex numbers

```
a.real, a.imag, a.conj
```

Numpy functions

- Data i/o: fromfile, genfromtxt, load, loadtxt, save, savetxt
- Mesh creation: grid, meshgrid, ohrid
- Manipulation: einsum, hstack, take, stack

Other sub packages

- **numpy.fft** : fast Fourier transforms
- **numpy.polynomial** : Efficient polynomials
- **numpy.linalg** : linear algebra

Cholesky, det, eig, eigvals, inn, lstsq, norm, qr, sve

- **numpy.maths** : C standard math functions
- **numpy.random** : random number generation

Beta, gamma, geometric, hypergeometric, lognormal, normal, poison, uniform, Weibull

Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (library, ascii, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. **Graphics: Matplotlib**
5. Some examples/exercises
6. How to read a file (library, ascii, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Graphics: matplotlib

https://matplotlib.org

- `Conda install matplotlib`
- `Pip install matplotlib`
- `Import matplotlib.pyplot as plt`

matplotlib Version 3.3.2

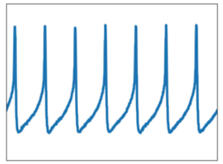
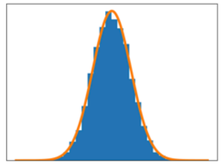
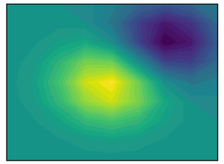
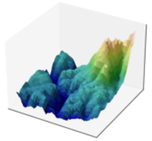
https://matplotlib.org

Installation Documentation Examples Tutorials Contributing

home | contents » Matplotlib: Python plotting

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Matplotlib makes easy things easy and hard things possible.

Create

- Develop publication quality plots with just a few lines of code
- Use interactive figures that can zoom, pan, update...

Customize

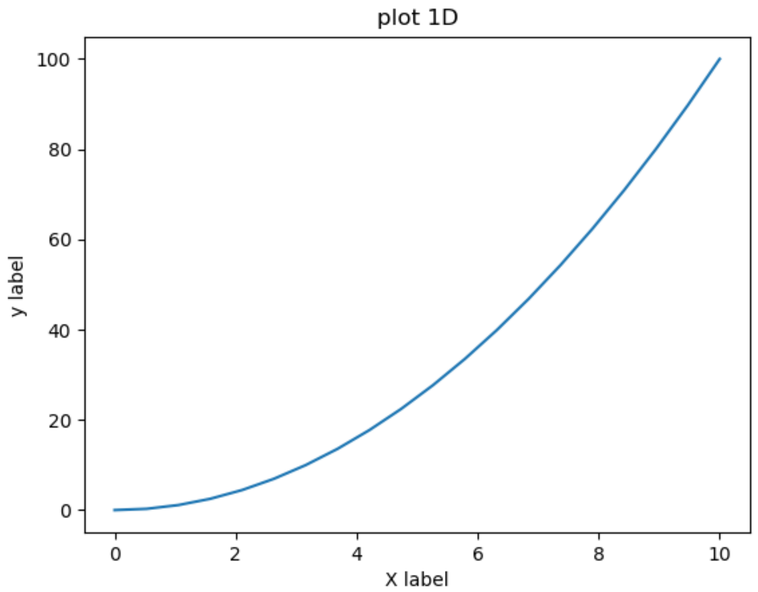
- Take full control of line styles, font properties, axes properties...
- Export and embed to a number of file formats and interactive environments

Extend

- Explore tailored functionality provided by third party packages
- Learn more about Matplotlib through the many external learning resources

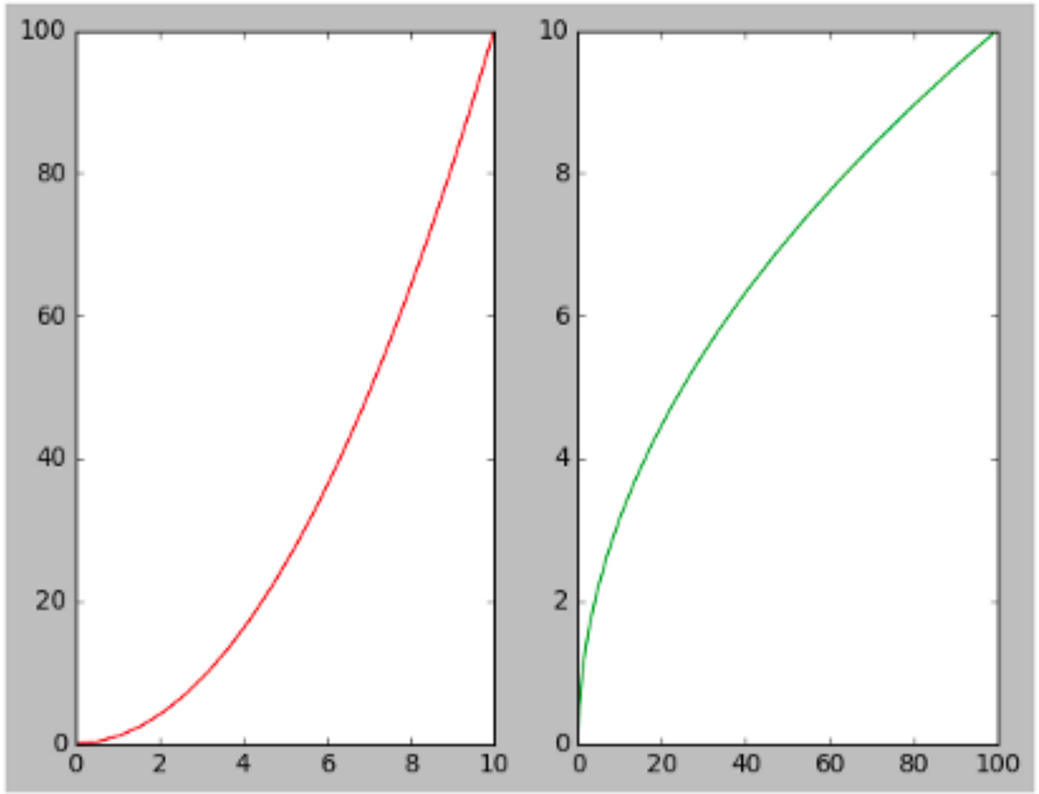
1D plot

```
import matplotlib.pyplot as plt
import numpy as np

x=np.linspace(0,10,20)
y=x**2
plt.plot(x,y)
plt.title("plot 1D")
plt.xlabel("X label")
plt.ylabel("y label")
plt.show()
```



1D plot

In [13]:
```
# plt.subplot(nrows, ncols, plot_number)
plt.subplot(1, 2, 1)
plt.plot(x, y, 'red') # More on color options later

plt.subplot(1,2,2)
plt.plot(y, x, 'green');
```



1D plot

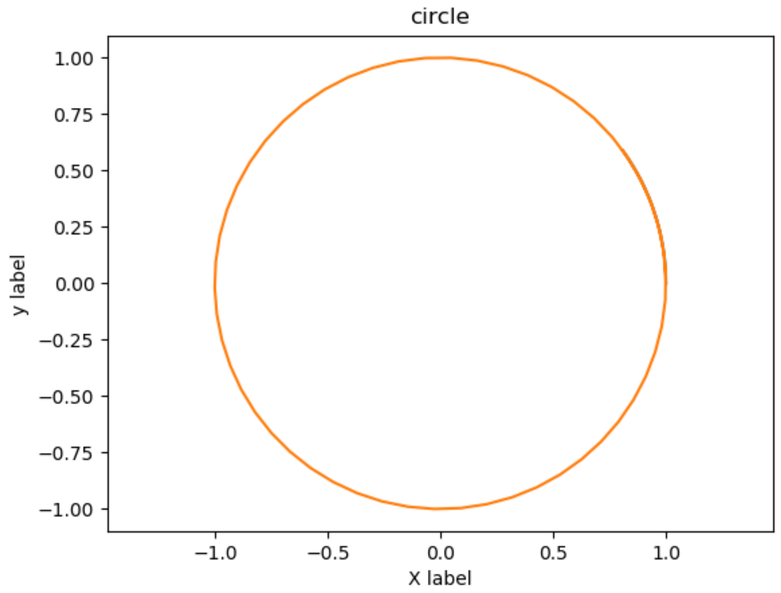
EXERCISE

Make the plot of cos (wt) vs sin(wt)
With w=2*pi and t varying between -60 to 60 with a step of 0.1

1D plot

EXERCISE

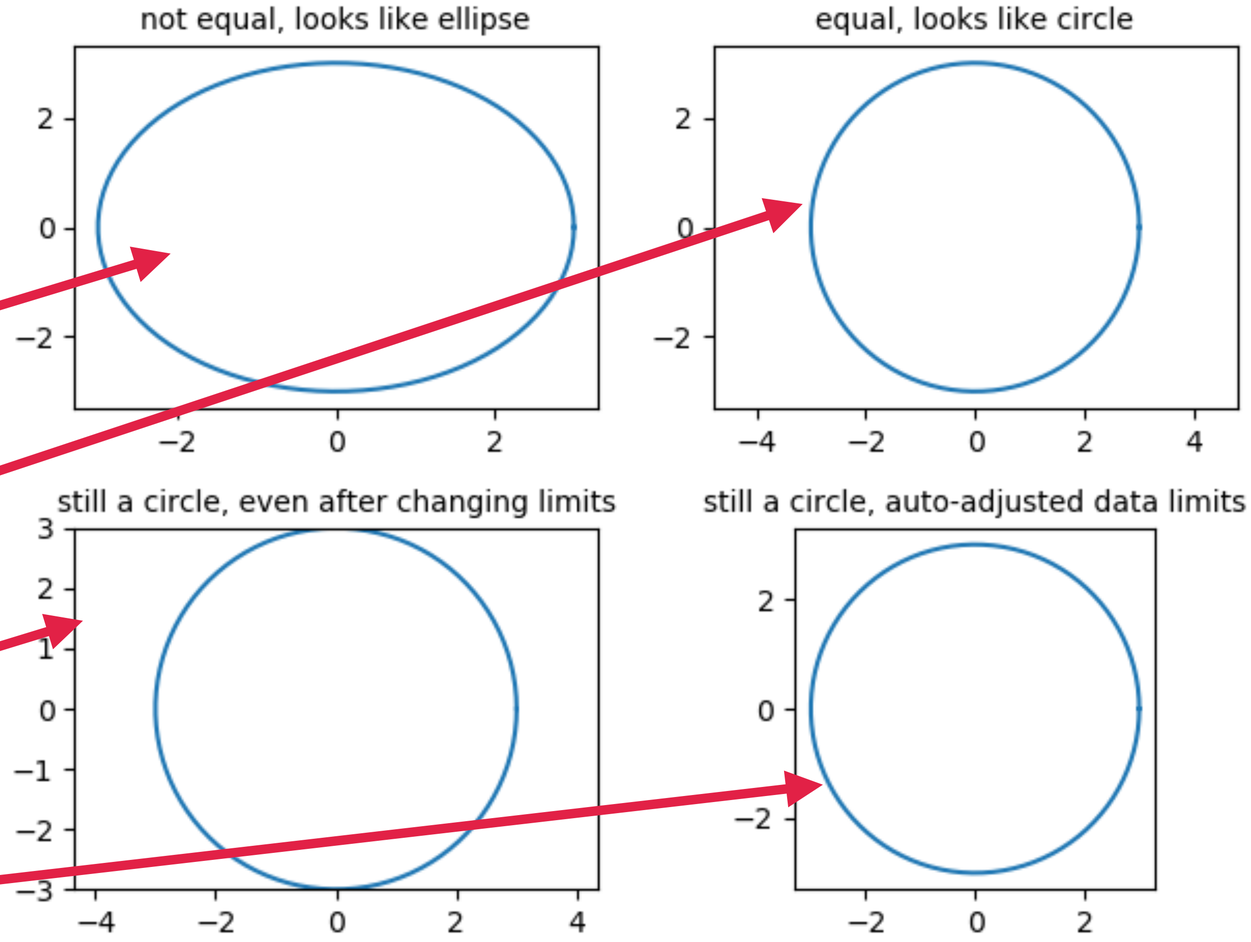
Make the plot of cos (wt) vs sin(wt)
With w=2*pi and t varying between -60 to 60 with a step of 0.1




```
import matplotlib.pyplot as plt
import numpy as np

t=np.linspace(-60,0.1, 60)
w=2*np.pi
plt.plot(np.cos(w*t),np.sin(w*t))
plt.title("circle")
plt.axis("equal")
plt.xlabel("X label")
plt.ylabel("y label")
plt.show()
```

Example (subplots)

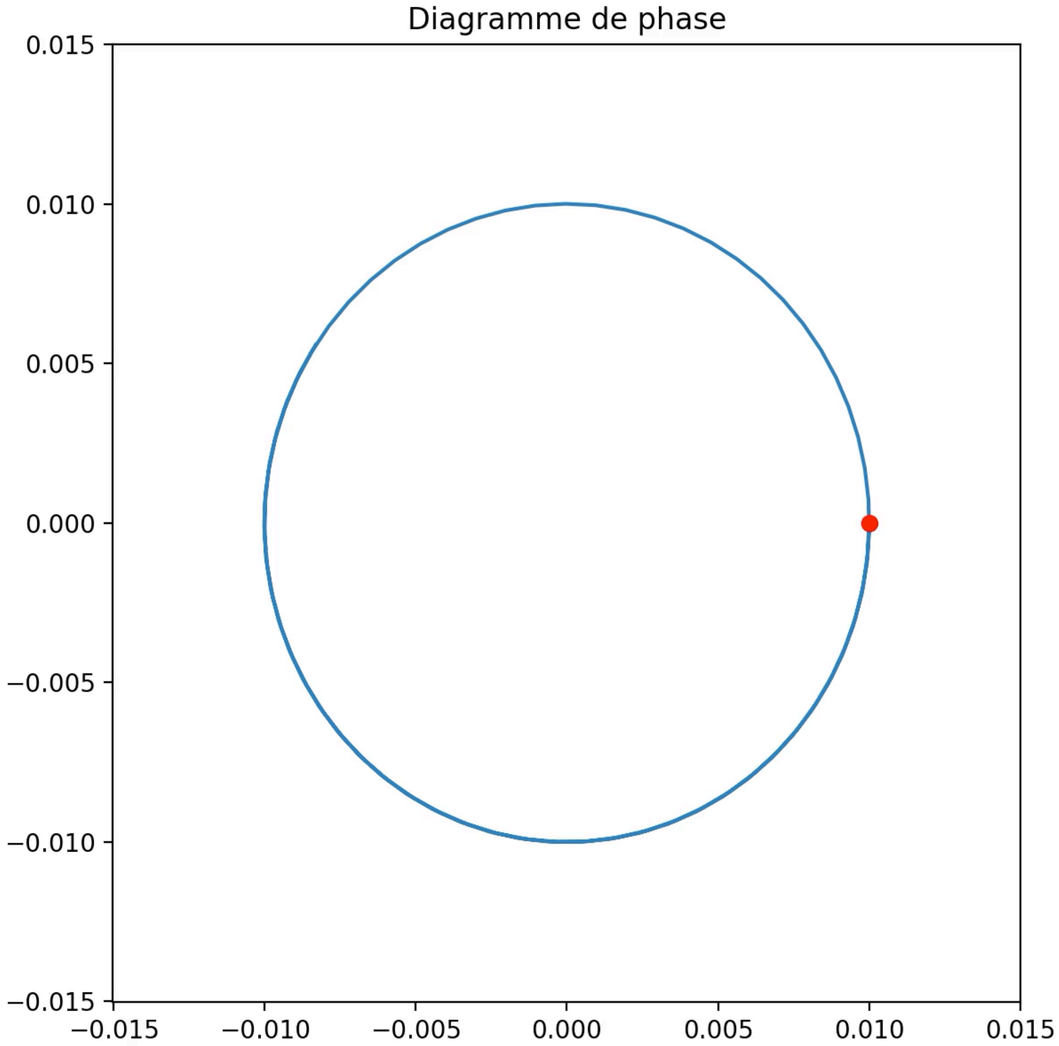
```
"""
===============
Axis Equal Demo
===============

How to set and adjust plots with equal axis ratios.
"""

import matplotlib.pyplot as plt
import numpy as np

# Plot circle of radius 3.

an = np.linspace(0, 2 * np.pi, 100)
fig, axs = plt.subplots(2, 2)

axs[0, 0].plot(3 * np.cos(an), 3 * np.sin(an))
axs[0, 0].set_title('not equal, looks like ellipse', fontsize=10)

axs[0, 1].plot(3 * np.cos(an), 3 * np.sin(an))
axs[0, 1].axis('equal')
axs[0, 1].set_title('equal, looks like circle', fontsize=10)

axs[1, 0].plot(3 * np.cos(an), 3 * np.sin(an))
axs[1, 0].axis('equal')
axs[1, 0].axis([-3, 3, -3, 3])
axs[1, 0].set_title('still a circle, even after changing limits', fontsize=10)

axs[1, 1].plot(3 * np.cos(an), 3 * np.sin(an))
axs[1, 1].set_aspect('equal', 'box')
axs[1, 1].set_title('still a circle, auto-adjusted data limits', fontsize=10)

fig.tight_layout()

plt.show()
```



Control the pad between the different subplots

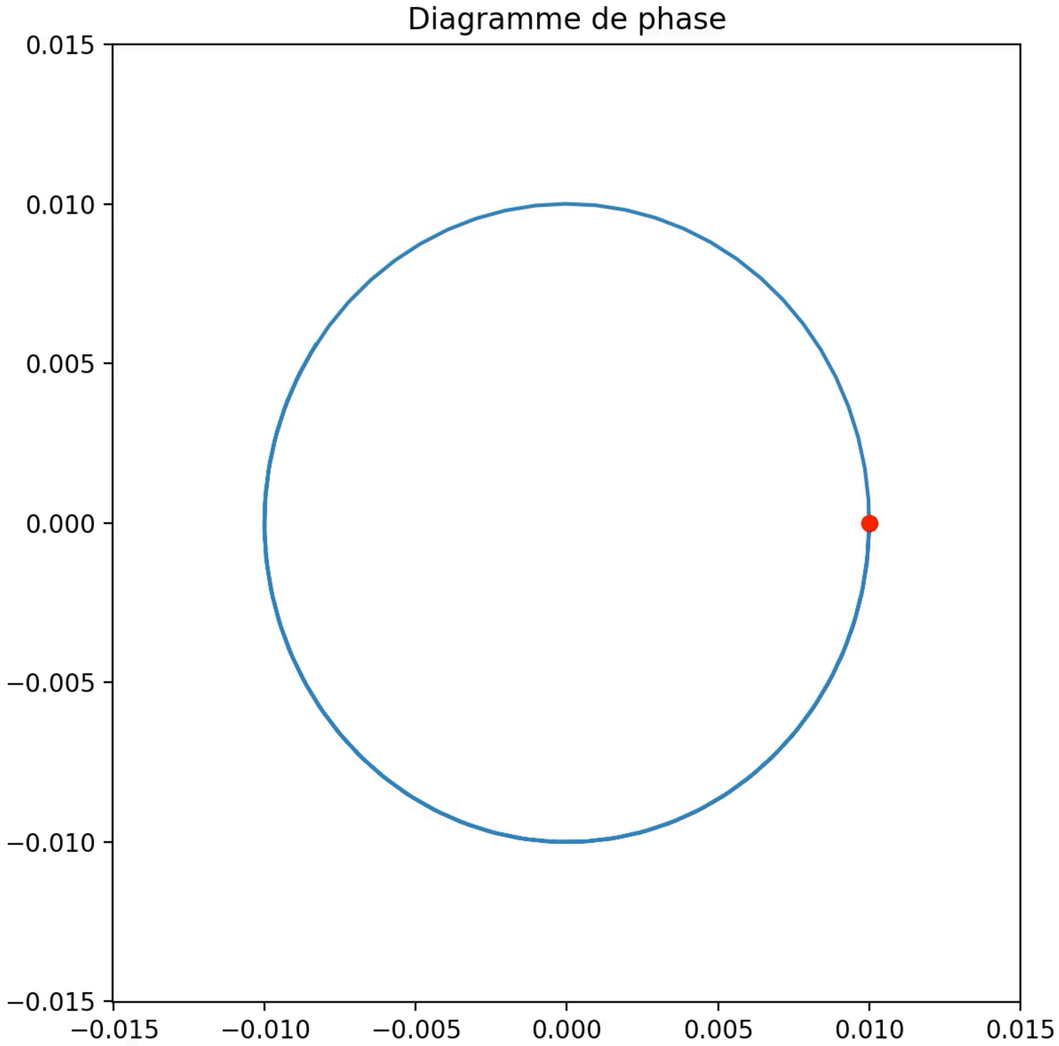
Exercise

- Just do this for fun (use animation sub package in matplotlib)



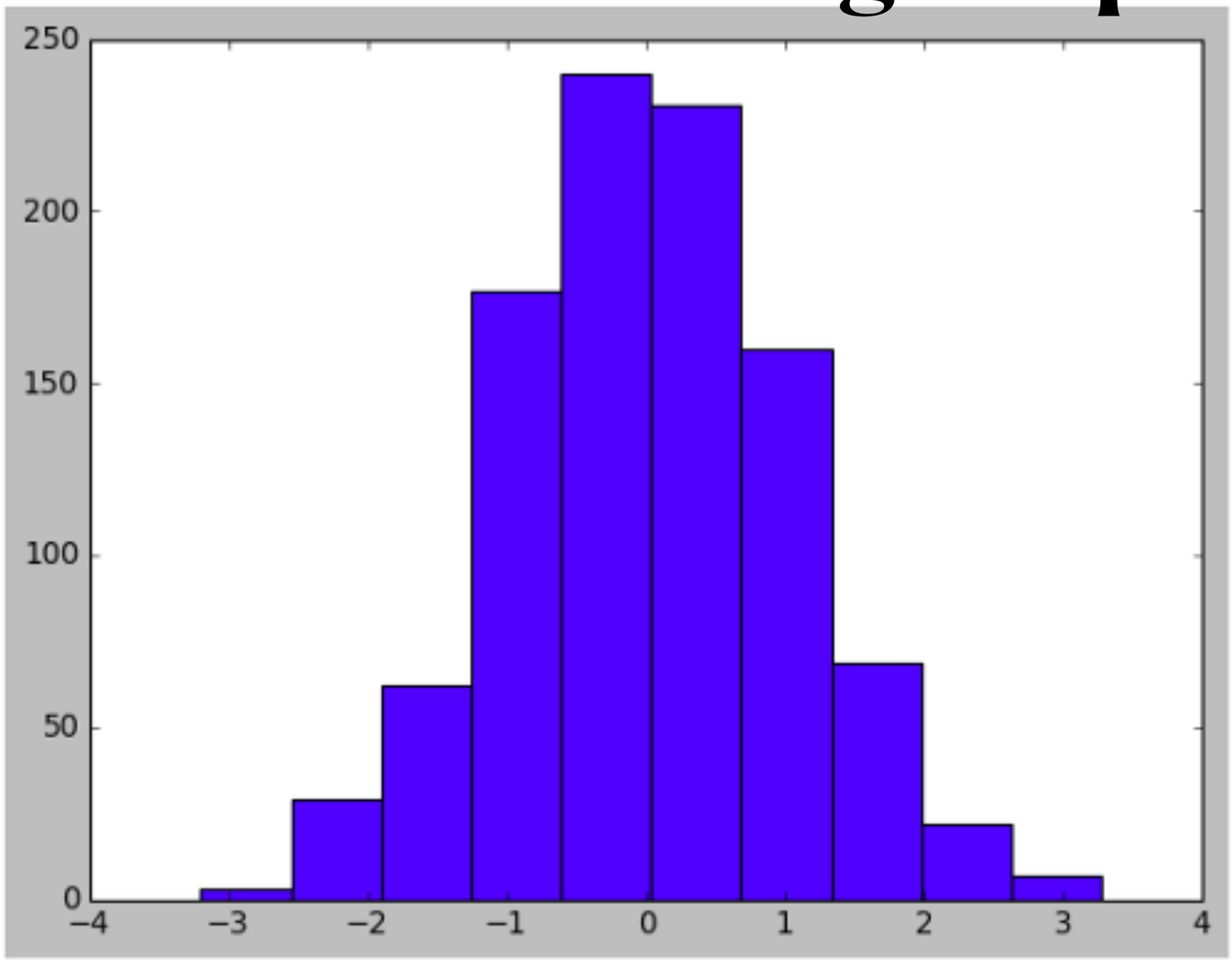
Exercise

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.animation as animation

'x is your cos and p is your sin, 1 D variable'

fig =plt.figure(figsize=(7, 7))
plt.xlim(np.min(x)*1.5, np.max(x)*1.5)
plt.ylim(np.min(x)*1.5, np.max(x)*1.5)
plt.plot(x,p)
plt.title('Diagramme de phase')
line2,= plt.plot(x[0], p[0], 'or')

def animate(i):
    line2.set_xdata(x[i])
    line2.set_ydata(p[i])

ani = animation.FuncAnimation(
    fig, animate, interval=200, frames=np.arange(0,100))
```



Histogram plot

```
In [90]: x = np.random.randn(1000)
         plt.hist(x);
```



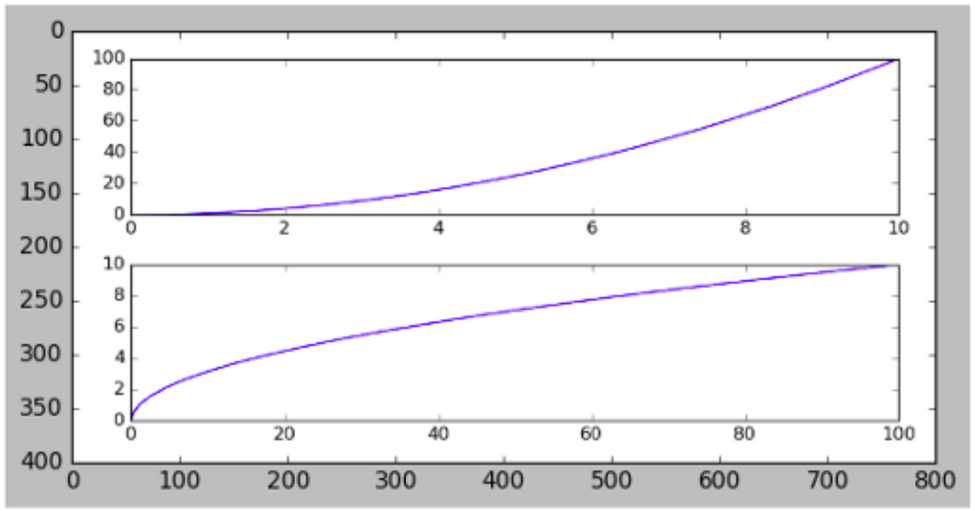
You can add bins, by bins=100 for example

How to save my figure and plot it again

```
fig.savefig('my_figure.png')
```

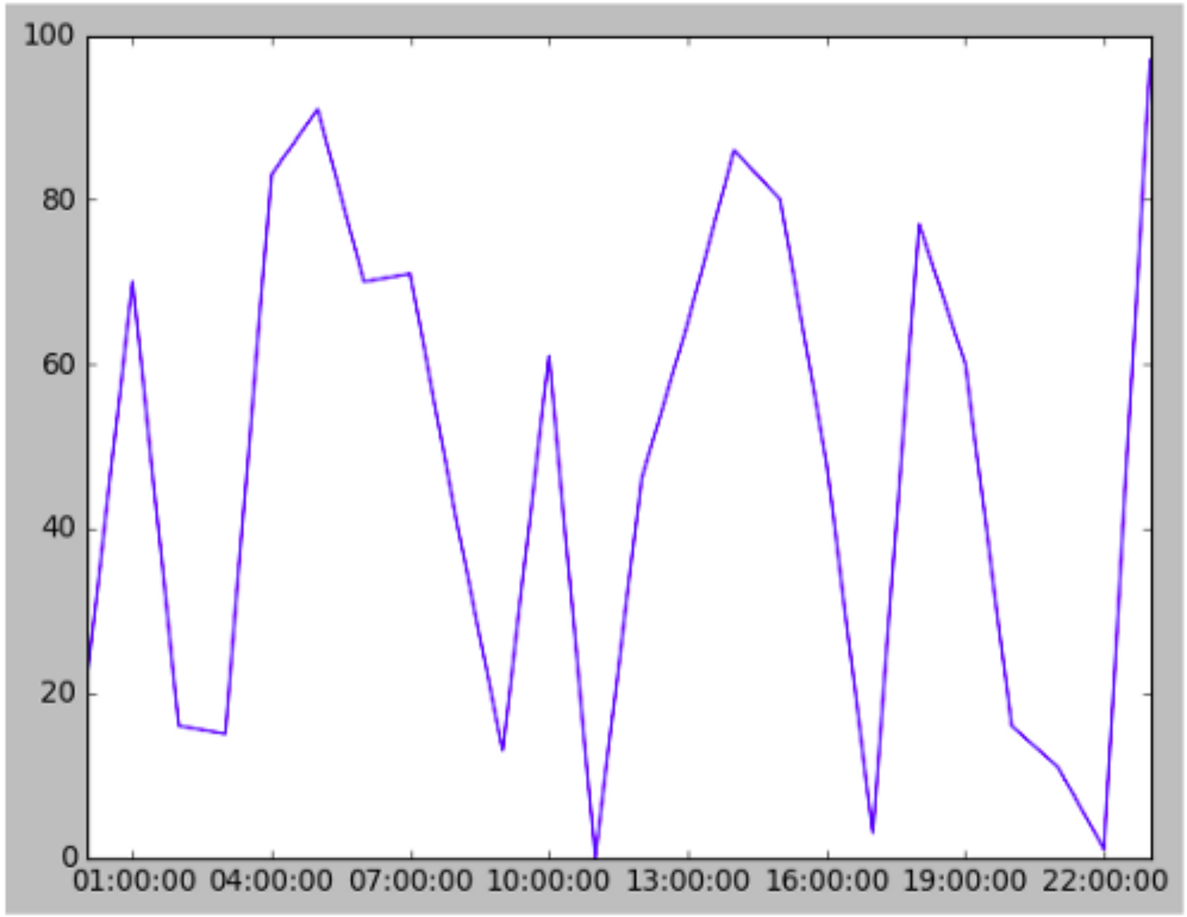
In [52]:
```
import matplotlib.image as mpimg

plt.imshow(mpimg.imread('my_figure.png'))
```

Out[52]:
```
<matplotlib.image.AxesImage at 0x7f00629afc88>
```



Datetime plot

In [227]:
```
import matplotlib.pyplot as plt
import datetime
import numpy as np

x = np.array([datetime.datetime(2018, 9, 28, i, 0) for i in range(24)])
y = np.random.randint(100, size=x.shape)

plt.plot(x,y)
plt.show()
```



2d plot and maps

```
from pylab import *     # import numpy and matplotlib
from datetime import datetime,timedelta,date

from mpl_toolkits.basemap import *          # mapping
```

(conda install -c anaconda basemap)

Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (library, ascii, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Content of the lecture

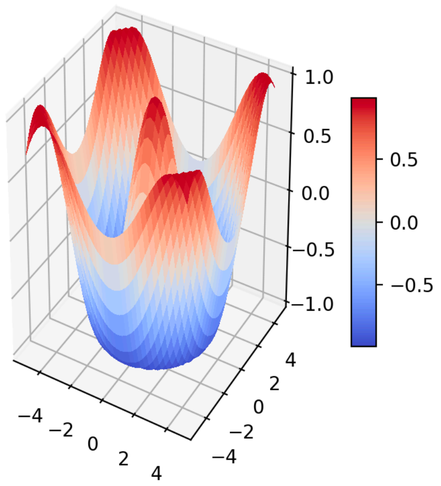
1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. **Some examples/exercises**
6. How to read a file (library, ascii, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

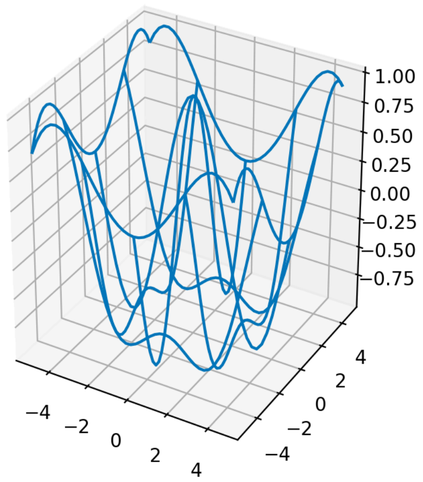
Graphics: matplotlib

Goal: plot this flower

$$z = cos(\sqrt{x^2 + y^2})$$

Where x and y are between -5 and +5 with a step of .25

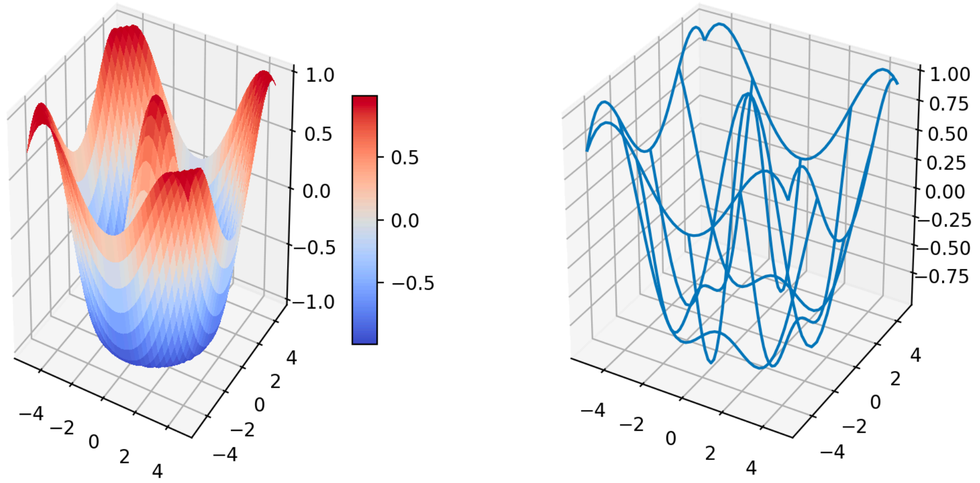




Help:
numpy (meshgrid)
matplotlib (figure, surface, colorbar, etc..)

Graphics: matplotlib

```
import matplotlib.pyplot as plt
import numpy as np

# imports specific to the plots in this example
from matplotlib import cm


# Twice as wide as it is tall.
fig = plt.figure(figsize=plt.figaspect(0.5))

#---- First subplot
ax = fig.add_subplot(1, 2, 1, projection='3d')
X = np.arange(-5, 5, 0.25)
Y = np.arange(-5, 5, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X**2 + Y**2)
Z = np.cos(R)
surf = ax.plot_surface(X, Y, Z, rstride=1, cstride=1, cmap=cm.coolwa
                       linewidth=0, antialiased=False)
ax.set_zlim3d(-1.01, 1.01)

fig.colorbar(surf, shrink=0.5, aspect=10)

#---- Second subplot
ax = fig.add_subplot(1, 2, 2, projection='3d')
#X, Y, Z = get_test_data(0.02)
ax.plot_wireframe(X, Y, Z, rstride=10, cstride=10)

plt.show()
```

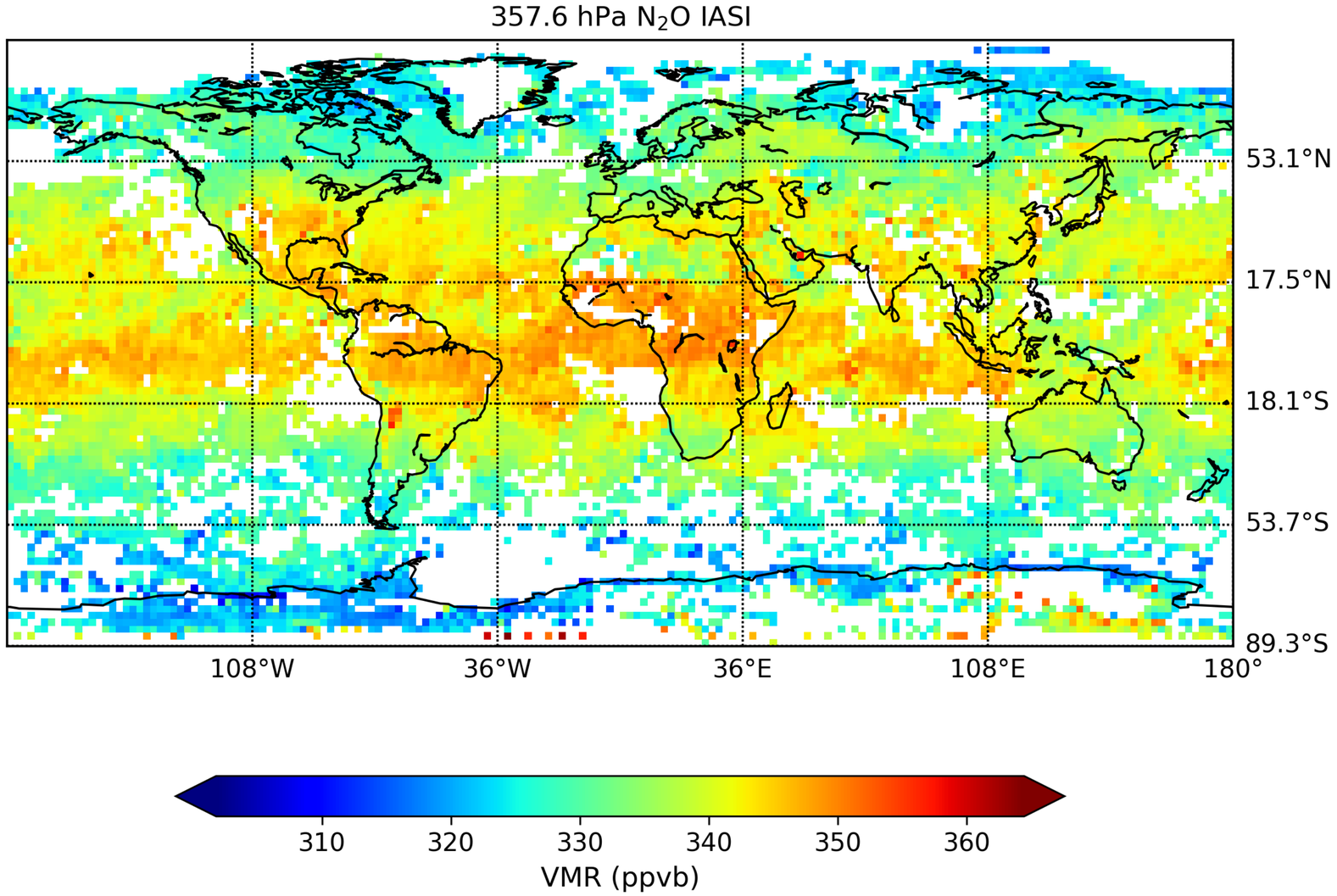


Basemap/2D plot

```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap, cm, shiftgrid, addcyclic

colmap = plt.cm.jet

 m = Basemap(projection='cyl',lat_o=lato,lon_o=lono,resolution='c',llcrnrlat=lat_min,urcrnrlat=lat_max,\
          llcrnrlon=lon_min,urcrnrlon=lon_max)


 var = m.pcolormesh(lon_plot, lat_plot , \
                  map_var.T,shading='flat',cmap=colmap, vmin = vmin, \
                  vmax = vmax, alpha=alpha
```

Graphics: matplotlib

Goal: plot the given netcdf file



Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (ascii, csv, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. **How to read a file (ascii, csv, netcdf, hdf files)**
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Main file formats

- Netcdf file
- hdf file
- CSV file
- Ascii file
- Image file (jpg, png, gif, ..)

Reading a netcdf file

Structure

ncdump -h iasi_ret_L2_n2o_ch4_2011_11_30.nc

```
netcdf iasi_ret_L2_n2o_ch4_2011_11_30 {
dimensions:
  n_pixels = 108887 ;
  n_levels = 14 ;
  n_level_P_std = 13 ;
variables:
  int year ;
  int month ;
  int day ;
  int hour(n_pixels) ;
  int minute(n_pixels) ;
  int millisecs(n_pixels) ;
  double lat(n_pixels) ;
    lat:unit = "degree north" ;
  double lon(n_pixels) ;
    lon:unit = "degree east" ;
  int day_night_index(n_pixels) ;
    day_night_index:description = "day = 0, night = 1, twilight = 2" ;
  int land_sea_index(n_pixels) ;
    land_sea_index:descrition = "land = 0, sea = 1, sea ice = 2" ;
  double xhi2(n_pixels) ;
  double level(n_level_P_std) ;
    level:unit = "hPa" ;
  double Surf_P(n_pixels) ;
    Surf_P:unit = "hPa" ;
  int Surf_P_id(n_pixels) ;
  double n2o_retrieval(n_pixels, n_levels) ;
    n2o_retrieval:unit = "ppbv" ;
  double n2o_apriori(n_levels) ;
    n2o_apriori:unit = "ppbv" ;
    n2o_apriori:description = "Mean profile from HIPPO 1-5" ;
  double n2o_AVK(n_pixels, n_levels, n_levels) ;
    n2o_AVK:description = "Averaging Kernel computed with log([n2o](ppmv
  double n2o_error(n_pixels, n_levels) ;
    n2o_error:unit = "ppbv" ;
  double n2o_Ss_error(n_pixels, n_levels) ;
  double n2o_Sm_error(n_pixels, n_levels) ;
  double ch4_retrieval(n_pixels, n_levels) ;
    ch4_retrieval:unit = "ppmv" ;
  double ch4_apriori(n_pixels, n_levels) ;
    ch4_apriori:unit = "ppmv" ;
    ch4_apriori:description = "MACC profiles" ;
  double ch4_AVK(n_pixels, n_levels, n_levels) ;
    ch4_AVK:description = "Averaging Kernel computed with log([ch4](ppmv
  double ch4_error(n_pixels, n_levels) ;
  double ch4_Ss_error(n_pixels, n_levels) ;
  double ch4_Sm_error(n_pixels, n_levels) ;

// global attributes:
    :description = "Daily n2o and ch4 retrieval data from IASI radiances
    :history = "Created Sun May 31 03:20:46 2020" ;
```

Reading a netcdf file

See the available datasets in the file

Network common data form (NetCDF) is commonly used to store **multidimensional geographic data**

Ncdf library

```
import netcdf4 as nc
```

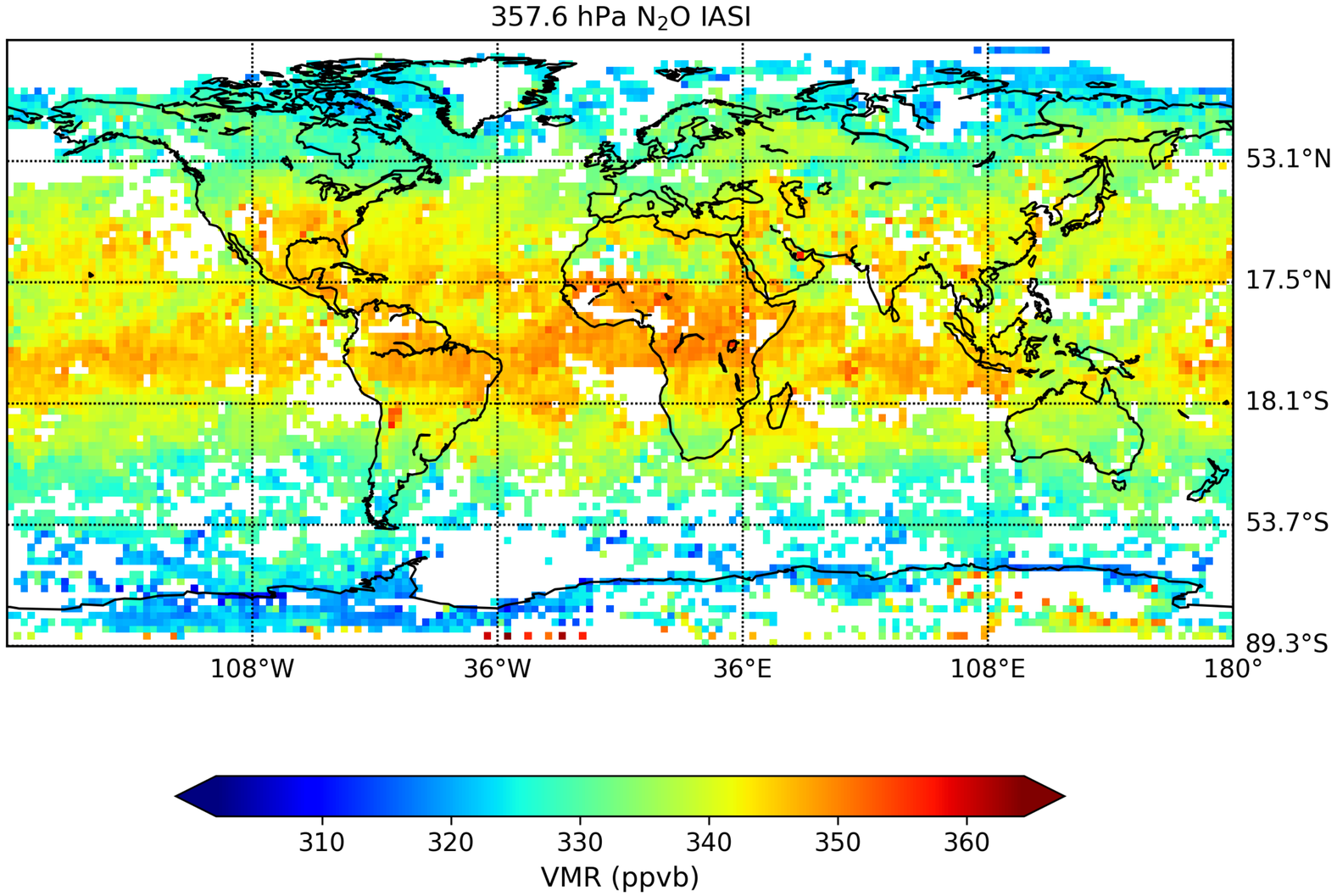
Read data

```
f = 'file.nc4'
ds = nc.Dataset(f)

% print data
print(ds)

print(ds.__dict__)

print(ds.__dict__[‘start_year']
for dim in ds.dimensions.values():
    print(dim)

for var in ds.variables.values():
    print(var)
```

```
for i in f.variables:
    print(i)

day = np.array(f.variables["day"][:])
hour = np.array(f.variables[« hour"][:])
minute= np.array(f.variables[« minute »][:])

lat = np.array(f.variables['lat'][:])
lon = np.array(f.variables[‘lon'][:])

data = np.array(f.variables['n2o_retrieval'])
data_units = f.variables["n2o_retrieval"].unit
```

Graphics: matplotlib

Goal: plot the given netcdf file



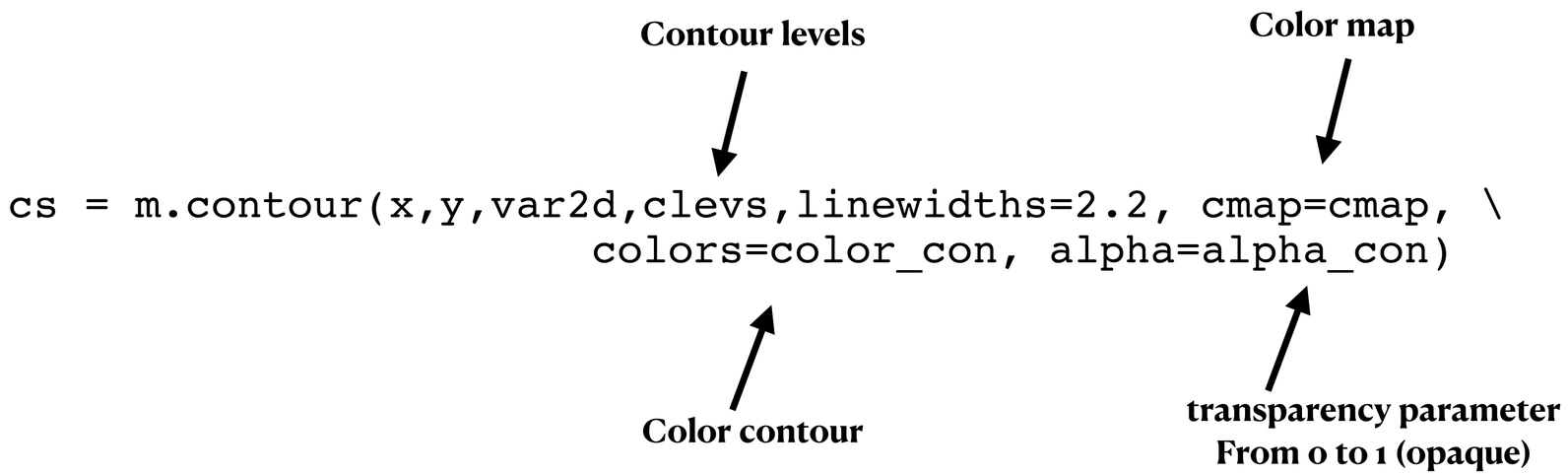
Graphics: matplotlib

Plot the map : use of basemap

```
import matplotlib.colors as mcolors
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap, cm, shiftgrid, addcyclic  # mapping


#---------------------------------------------------------------------
# Cylindrical Projection

m = Basemap(projection='cyl',lat_0=lat0,lon_0=lon0,resolution='c',
            llcrnrlat=lat_min,urcrnrlat=lat_max,\
            llcrnrlon=lon_min,urcrnrlon=lon_max)
```

Graphics: matplotlib

Plot a contour

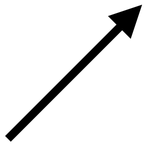


Graphics: matplotlib

Plot the map : use of basemap

```
var = m.pcolormesh(lon_plot[a], lat_plot[b] , \
                   map_var.T,shading='flat',cmap=colmap, vmin = vmin, \
                   vmax = vmax, alpha=alpha)
```

```
cbar = m.colorbar(location='right', size='5%', pad='2%')
cbar.set_label(zlabel,size=12)
```



Plot a color bar

Graphics: matplotlib

Plot the map : draw boundaries, coastlines, mask, parallels, etc..

```
m.drawcoastlines(linewidth=1.5, linestyle='solid', color='k', antialiased=1,\
                 zorder=2)

#    m.drawmapboundary()
#    m.drawstates()
     m.drawcountries()

#    m.drawlsmask(land_color='coral',ocean_color='aqua',lakes=True)

#    m.drawlsmask(land_color='none', ocean_color='aqua', zorder=1)
#    m.drawlsmask(land_color=land_color, ocean_color='none', zorder=1)
#        m.bluemarble(zorder=2)

m.drawlsmask(land_color='none', ocean_color=sea_color, zorder=1)

parallels = np.arange(-90,90 +0.01,30)
meridians = np.arange(-180,180 +0.01,60)

m.drawparallels(parallels,labels=[True,False,False,False],fontsize=11)
m.drawmeridians(meridians,labels=[False,False,False,True],fontsize=11)

plt.show()
```

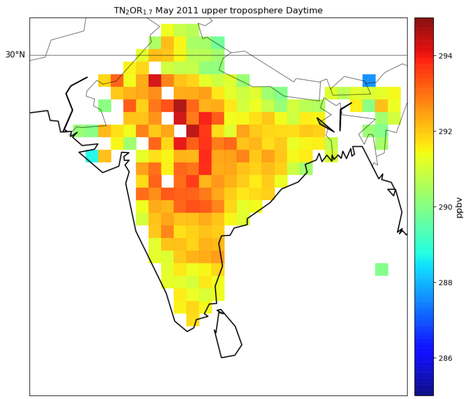
Draw on the map:
Coastlines,
boundaries,
states,
countries,
mask

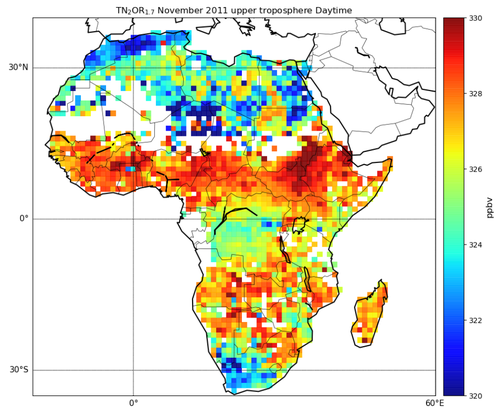
Draw on the map:
Parallels and
Meridians

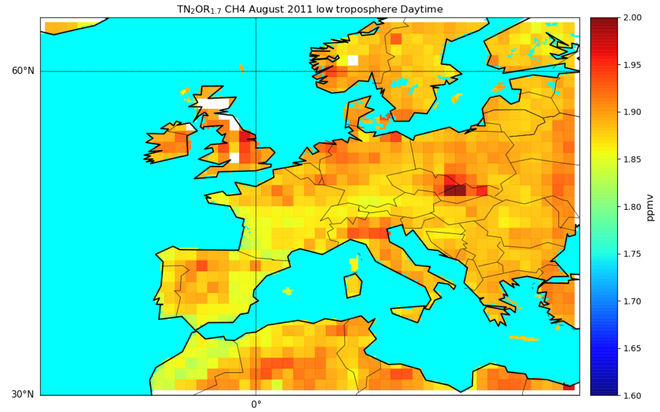
Graphics: matplotlib

Plot the map : draw boundaries, coastlines, mask, parallels, etc..

1. Plot the netcdf file
2. Superimpose the contour lines
3. Add countries
4. Select a part of the globe (such as South America, Africa, Australia)







Reading a hdf file

1. List the available datasets in the file

```
# hdf 5 file library
import h5py


# Read the hdf file
with h5py.File('file.hdf', mode='r') as f:
        ## List available datasets.
        print (f.keys())
```

hdf file library

keys is the function to list the available dataset

Reading a hdf file

See the available datasets in the file

```
# hdf 5 file library
import h5py


# Read the hdf file
with h5py.File(‘file.hdf’, mode='r') as f:
        ## List available datasets.
        print (f.keys())
```

hdf library

keys is the function to list the available dataset

Read/Write a CSV file

Numerous other ways to create and format your CSV file. See following link

```
Voir https://docs.python.org/fr/3/library/csv.html
```

Read/Write a CSV file

CSV format (Comma Separated Values) is the most common format for import and export of data sheet and database

See *https://docs.python.org/fr/3/library/csv.html*

Two libraries

`Library csv`

`Library pandas`

Read/Write a CSV file

csv.reader(csvfile, dialect='excel', **fmtparams)¶

```
import csv
>>> with open('eggs.csv', newline='') as csvfile:
...     spamreader = csv.reader(csvfile, delimiter=' ', quotechar='|')
...     for row in spamreader:
...         print(', '.join(row))
```

Read/Write a CSV file

csv.writer(csvfile, dialect='excel', **fmtparams)¶

CSV file library

```
import csv

with open('eggs.csv', 'w', newline='') as csvfile:
    spamwriter = csv.writer(csvfile, delimiter=' ',
                            quotechar='|', quoting=csv.QUOTE_MINIMAL)
    spamwriter.writerow(['Spam'] * 5 + ['Baked Beans'])
    spamwriter.writerow(['Spam', 'Lovely Spam', 'Wonderful Spam'])
```

Reading a CSV file

Print the available datasets from the csv file

```
import pandas as pd

chunksize = 10 ** 8

for chunk in pd.read_csv("fire_nrt_V1_106471.csv", chunksize=chunksize):

    chunk.sort_values(by=['acq_date'])
    latitude = chunk.loc[:]['latitude']
    longitude = chunk.loc[:]['longitude']
    date = chunk.loc[:][‘acq_date']

print(latitude, longitude, date)
```

Pandas CSV library

sort_values and loc are the commands to take the available dataset

Reading an /ascii/image file

See the available datasets in the file

This is to read almost all image formats

```
import numpy as np
import matplotlib.pyplot as plt
```

Typical libraries

```
img=plt.imread('great_lakes.jpg')
print(img)
```

Image reading (jpg, png, tif)

```
from astropy.extern.six.moves.urllib import request
url = 'http://python4astronomers.github.com/_downloads/data.txt'
open('data.txt', 'wb').write(request.urlopen(url).read())
```

Download an ascii file

```
f = open('data.txt', 'r')
```

Open the ascii file

```
f.read()
```

Print the data

Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (ascii, csv, netcdf, hdf files)
7. Some examples/exercises
8. How to save a file
9. Useful Python libraries

Content of the lecture

1. Structure of a Python script
2. Some examples/exercises
3. Mathematics: Numpy library
4. Graphics: Matplotlib
5. Some examples/exercises
6. How to read a file (ascii, csv, netcdf, hdf files)
7. **Some examples/exercises**
8. How to save a file
9. Useful Python libraries

Exercise

Goal : read a hdf/netcdf file and plot the data

What do we need ?

1. The hdf file
2. The adequate libraries
3. The adequate commands
4. To write the code
5. Check the code step by step (printing or plotting intermediary results)
6. Plot the data
7. Check the consistency of the plot

Exercise 1

Goal : read a image file and plot the data

What do we need ?

1. The hdf file
2. The adequate libraries
3. The adequate commands
4. To write the code
5. Check the code step by step (printing or plotting intermediary results)
6. Plot the data
7. Check the consistency of the plot

Exercise

Goal : Filtering an image (convolution)

The 2D discrete convolution

The 2D discrete convolution is an extension of the 1D discrete convolution but the convolution is done both horizontally and vertically in a 2D spatial domain. These calculations are used in the image processing domain. For example you can smooth an image, make it net or detect different boundaries within the image.

The 2D discrete convolution equation can be written as:

$$y(m, n)=\sum_{k}\sum_{l} x(m-k, n-l)\, h(k,l)$$

In this case the function h (called kernel) is centered which means the central point of h is h[0,0] and x is the 2D image.

1) Use a satellite image from internet (made in the Visible)

- Remind the convolution theorem and use the most efficient way to calculate the convolution of an image.

- Take h as a Gaussian (see next equation) and calculate the 2D convolution. The Gaussian is defined as:

```
h[k,l]=exp(-(n1[k]**2+n2[l]**2)/(2*sigma**2));
```

- Change the sigma parameter and conclude.
- Try different kernels given during the class. Explain why a kernel h can blurry the image?
- Try different images you can download from internet. Conclusions.

The convolution theorem states that

$$\mathcal{F}\{f * g\} = k \cdot \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

Python commands : fft2, ifft2,

Exercise 2

Goal : Filtering an image (convolution) using fft2, ifft2

```
def fft_convolve2d(x,y):
    """ 2D convolution, using FFT"""
    fr = fft2(x)
    fr2 = fft2(np.flipud(np.fliplr(y)))
    m,n = fr.shape
    cc = np.real(ifft2(fr*fr2))
    cc = np.roll(cc, -m/2+1,axis=0)
    cc = np.roll(cc, -n/2+1,axis=1)
    return cc
```

For this year one topic :
N_2O and CH_4

Year 2011

1-Link between nitrous oxide and vegetation index for different seasons
2-Validation of IASI CH4 using FTIR data (NDACC data)
3-Link between nitrous oxide and soil humidity as measured by satellites
4-Link between nitrous oxide and biomass burning/evergreen forest over Africa
5-Link between nitrous oxide and biomass burning/evergreen forest over South America
6-Impact of clouds on IASI nitrous oxide measurements over India.

Different periods could be investigated and other continents such as India, Australia or South East Asia.
In addition, surface parameters that could be used are NDVI, LAI, Soil humidity, use of fertilizers, agriculture areas …

Concerning the biomass burning fire accounts and CO measurements (MOPITT) could be used easily and downloaded from the Web.

UE 52 topics - 2020

Link between Nitrous oxide and NDVI for 2 seasons (winter and summer)	Link between Nitrous oxide and NDVI for 2 seasons (fall and spring)	Link between N2O and soil humidity for 2 seasons (summer and Spring)	Link between N2O and biomass burning over Africa (winter and summer)	Link between N2O and biomass burning over Africa (summer and spring)	Link between N2O and biomass burning over SA	Link between N2O and biomass burning over Africa (fall and spring)	Impact of clouds on IASI N2O over India
NOREILLE CALVINHAC	CURNELLE BARRAFITTE	MRAD BANGMA	KADOURAN DESTARAC	AUDEMARD LOUBET	SANDER BRYSON	DIOUF FIALTON KHALID	Maestro Redondo, Hartmann, Paranjape

Useful Web sites

Satellite Data

https://land.copernicus.eu/global/index.html

https://eoportal.eumetsat.int/

www.acom.ucar.edu/mopitt/webview/plot-options.shtml

www.temis.nl

https://www.ncdc.noaa.gov

www.noaa.gov

https://search.earthdata.nasa.gov/search

hdfeos.org

Modelling data

https://www.ecmwf.int (global modelling data)
https://ready.arl.noaa.gov/HYSPLIT.php (Trajectories run)
https://www2.prevair.org (Air quality in France)

Information SITE

www.meteofrance.fr
http://omer7a.obs-mip.fr

Lock down lecture

How do we do today ?

- Today 14h-17h (We have 12 h together for this semester)
- Focus on the N2O satellite data
    1. Read the N2O data using the code I already gave to you
    2. Average the data in a variable box at global scale
    3. Plot the data
- Keep working on your N2O project
    1. Ask me any question at any time
    2. Share your screen and discussion for the best solution

Lock down lecture

For the next lectures ?

Questions

- What is the best time slot (2h or 3h) ? (Remaining 9 h until beginning of March 2021)
- Frequency ? (Nov, Dec, Jan, Feb) 1 or 2 per month.
- When ? (in the morning, afternoon, late afternoon)
- Send me any questions by email before the lecture slot and I could prepare and answer on slides. I’ll try to answer as soon as possible anyway.

Reading a netcdf file

Structure

ncdump -h iasi_ret_L2_n2o_ch4_2011_11_30.nc

```
netcdf iasi_ret_L2_n2o_ch4_2011_11_30 {
dimensions:
  n_pixels = 108887 ;
  n_levels = 14 ;
  n_level_P_std = 13 ;
variables:
  int year ;
  int month ;
  int day ;
  int hour(n_pixels) ;
  int minute(n_pixels) ;
  int millisecs(n_pixels) ;
  double lat(n_pixels) ;
    lat:unit = "degree north" ;
  double lon(n_pixels) ;
    lon:unit = "degree east" ;
  int day_night_index(n_pixels) ;
    day_night_index:description = "day = 0, night = 1, twilight = 2" ;
  int land_sea_index(n_pixels) ;
    land_sea_index:descrition = "land = 0, sea = 1, sea ice = 2" ;
  double xhi2(n_pixels) ;
  double level(n_level_P_std) ;
    level:unit = "hPa" ;
  double Surf_P(n_pixels) ;
    Surf_P:unit = "hPa" ;
  int Surf_P_id(n_pixels) ;
  double n2o_retrieval(n_pixels, n_levels) ;
    n2o_retrieval:unit = "ppbv" ;
  double n2o_apriori(n_levels) ;
    n2o_apriori:unit = "ppbv" ;
    n2o_apriori:description = "Mean profile from HIPPO 1-5" ;
  double n2o_AVK(n_pixels, n_levels, n_levels) ;
    n2o_AVK:description = "Averaging Kernel computed with log([n2o](ppmv
  double n2o_error(n_pixels, n_levels) ;
    n2o_error:unit = "ppbv" ;
  double n2o_Ss_error(n_pixels, n_levels) ;
  double n2o_Sm_error(n_pixels, n_levels) ;
  double ch4_retrieval(n_pixels, n_levels) ;
    ch4_retrieval:unit = "ppmv" ;
  double ch4_apriori(n_pixels, n_levels) ;
    ch4_apriori:unit = "ppmv" ;
    ch4_apriori:description = "MACC profiles" ;
  double ch4_AVK(n_pixels, n_levels, n_levels) ;
    ch4_AVK:description = "Averaging Kernel computed with log([ch4](ppmv
  double ch4_error(n_pixels, n_levels) ;
  double ch4_Ss_error(n_pixels, n_levels) ;
  double ch4_Sm_error(n_pixels, n_levels) ;

// global attributes:
    :description = "Daily n2o and ch4 retrieval data from IASI radiances
    :history = "Created Sun May 31 03:20:46 2020" ;
```

Reading a netcdf file

See the available datasets in the file

Network common data form (NetCDF) is commonly used to store **multidimensional geographic data**

```
import netcdf4 as nc
```

Ncdf library

```
file = 'file.nc4'
f = nc.Dataset(file)
```

Read data

```
% print data
print(ds)
```

```
for i in f.variables:
    print(i)

day = np.array(f.variables["day"][:])
hour = np.array(f.variables[« hour"][:])
minute= np.array(f.variables[« minute »][:])

lat = np.array(f.variables['lat'][:])
lon = np.array(f.variables[‘lon'][:])

data = np.array(f.variables['n2o_retrieval'])
data_units = f.variables["n2o_retrieval"].unit
```

Reading a netcdf file

Use of read_IASI.py

```
import numpy as np
from collections import namedtuple
from netCDF4 import Dataset

#--------------------------------------------------

def read_L2_IASI_nc(filename, variable_name=None, var=None, test_xhi2=None, \
                    daytime=None, nighttime=None, land=None, sea=None, \
                    treshold_dof=None):
```

Reading a netcdf file

Use of read_IASI.py

```
f=Dataset(filename)
print('FILENAME: ', filename)

#Print variables from the file (uncomment to see the variables that you access from the data file)
"""
for i in f.variables:
    print(i)
"""
# f.variables[<your_variable>] allows to access to your_variable, you can use it as a base to read units or length of each variable
#Exemple to print the unit of the variable "retrieval_n2o" and the length of the variable "month"
"""
print(f.variables["n2o_retrieval"].unit)

"""
```

Reading a netcdf file

Use of read_IASI.py

```
#Get the values of a variable in numpy array

year = np.array(f.variables["year"][:])
month = np.array(f.variables["month"][:])
day = np.array(f.variables["day"][:])
hour = np.array(f.variables["hour"][:])
minute= np.array(f.variables["minute"][:])
millisecs = np.array(f.variables["millisecs"][:])
lat = np.array(f.variables['lat'][:])
lon = np.array(f.variables['lon'][:])
xhi2 = np.array(f.variables['xhi2'][:])
day_night_index=np.array(f.variables['day_night_index'][:])
land_sea_index=np.array(f.variables['land_sea_index'][:])
p=np.array(f.variables['level'][:])
sp=np.array(f.variables['Surf_P'][:])
```

Usually we take less than 2

day=0, night=1

land=0, sea=1

Reading a netcdf file

Use of read_IASI.py

```
if land != None :
    select=np.where(land_sea_index==0)[0]
    data=data[select, :]
    lat=lat[select] ; lon=lon[select]
    sp=sp[select] ; hdec=hdec[select]; day_night_index=day_night_index[select]
    land_sea_index=land_sea_index[select]; avk=avk[select,:,:]

if sea != None :
    select=np.where(land_sea_index==1)[0]
    data=data[select, :]
    lat=lat[select] ; lon=lon[select]
    sp=sp[select] ; hdec=hdec[select]; day_night_index=day_night_index[select]
    land_sea_index=land_sea_index[select]; avk=avk[select,:,:]

# Calculation of dof
  dof=np.nansum(np.diagonal(avk, axis1=1, axis2=2), axis=1)
```

land=0, sea=1

Calculation of degrees of Freedom
Which shows the sensitivity of the data on the vertical
You can choose a specific dof (treshold_dof)

If dof > 1 more than 1 information on the vertical

Reading a netcdf file

Use of read_IASI.py - output

```
from collections import namedtuple


Data = namedtuple('Data', 'data, hdec, dof, lat, lon, p, sp, year, day, \
                  avk, month, error')

return Data(data=data, lat=lat, lon=lon, hdec=hdec, p=p, sp=sp, \
        year=year, day=day, month=month, dof=dof, avk=avk, error=error)
```

N2O=read_L2_IASI_nc(filename='file.nc', varname='n2o_retrieval', ...)

N2O.data, N2O.lat

Box Average of IASI data

```
#-----------------------------------------------------------
# Remove nan in var (IASI data from read_IASI.py
#-----------------------------------------------------------
   n_lat, n_lon=len(lat_grid), len(lon_grid)
#    test if there is nan
   test=np.isnan(var) == False
   var=var[test]
   lati=lati[test]
   longi=longi[test]

   nb=len(lati)    # Number of elements or number of pixels

#-----------------------------------------------------------
#Average and regrid the IASI data
#-----------------------------------------------------------
```

```
lat_min=-90; lat_max=90; lon_min=-180; lon_max=180
lat_grid = arange(lat_min + res/2, lat_max,res)
lon_grid = arange(lon_min + res/2, lon_max,res)
```

You can make a function !!

Box Average of IASI data

```
#-----------------------------------------------------------
#Average and regrid the IASI data
#-----------------------------------------------------------
   average=np.zeros((n_lon,n_lat))
   count=np.zeros((n_lon, n_lat))

   for k in range(0,nb,1):
      diff_lat=np.abs(lati[k]-lat_grid)
      diff_lon=np.abs(longi[k]-lon_grid)
      ind_lat=np.where(diff_lat == min(diff_lat))[0]
      ind_lon=np.where(diff_lon == min(diff_lon))[0]
      if var[k] != np.nan:
         average[ind_lon, ind_lat] +=var[k]
         count[ind_lon, ind_lat]+=1.

   average=average/count
   test=np.where(count == 0)
   average[test]=np.nan
```

This code is time consuming
Python challenge: make a code faster than this one

```
import time
start_time = time.time()

    Your code or my code

print('Temps=',(time.time()-start_time)/60,'min')
```

Plot the N2O data

```
m = Basemap(projection='cyl',lat_0=lato,lon_0=lono,resolution='c',llcrnrlat=lat_min,urcrnrlat=lat_max,\
            llcrnrlon=lon_min,urcrnrlon=lon_max)

# --- Add the figure on the map
fig = plt.figure(figsize=(12,9))
ax = fig.add_axes([0.1,0.1,0.8,0.8])

# --- Contour plot Make the grid axes and prepare the 2D file for the contour in a lon_min, lon_max x lat_min, lat_max box
a = np.where((lon_con>=lon_min) & (lon_con<=lon_max))[0]
b= np.where((lat_con>=lat_min) & (lat_con<=lat_max))[0]
contour_var = contour_var2d[a[:,None], b[None,:]]
lons, lats = np.meshgrid(lon_con[a],lat_con[b])
x, y = m(lons, lats)

 cs = m.contour(x, y, contour_var.T , clevs,  linewidths=2.2,   cmap=cmap, \
                colors=color_con, animated=True, alpha=alpha_con)
```

Plot the N2O data

```
# --- Plot color field

# Clip the map within the lat lon boundaries
a = np.where((lon_plot>=lon_min) & (lon_plot<=lon_max))[0]
b= np.where((lat_plot>=lat_min) & (lat_plot<=lat_max))[0]
map_var = var2d[a[:,None], b[None,:]]

var = m.pcolormesh(lon_plot[a], lat_plot[b] , \
                   map_var.T,shading='flat',cmap=colmap, vmin = vmin, \
                   vmax = vmax, alpha=alpha)
```