

Trabajando con Python en Ciencia e Ingeniería

Mabel Delgado (@mabeldelgadob)

PyDay Granada 2018

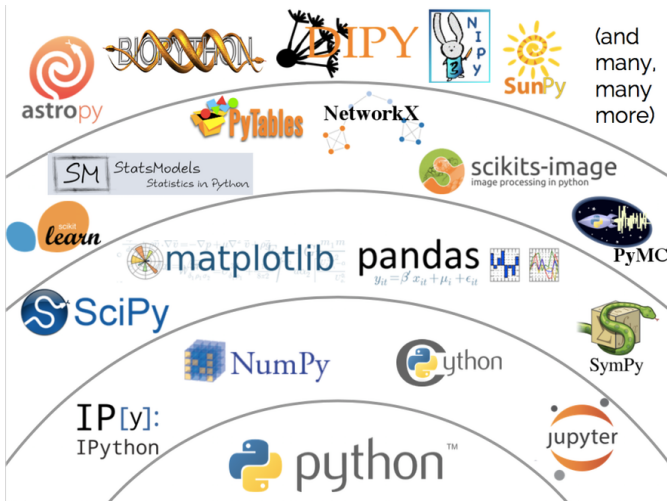
15 de diciembre de 2018

Un poco sobre mí...

- Ingeniera aeronáutica.
- Métodos y herramientas para ensayos en vuelo de aviones.
- Programación, software libre.
- PyLadies Madrid.
- AeroPython.
- Asociación Python España.
- Django Girls (Cáceres, Palma, Málaga).
- Python Software Foundation.



Ecosistema Python para ciencia e ingeniería



<https://www.datacamp.com/community/blog/python-scientific-computing-case>

Qué vamos a ver hoy...

- **NumPy** → arrays n-dimensionales.
- **SciPy** → algoritmos y funciones matemáticas.
- **Matplotlib** → representaciones gráficas 2D.
- **Jupyter (notebook, lab)** → entorno interactivo de trabajo.

Pero hay otros muchos paquetes que proporcionan desde funcionalidades generales (e.g. Pandas, SimPy, Numba, Bokeh...), a otras más específicas (Astropy, PyFiltering...).

NumPy - Introducción

NumPy es un paquete fundamental para la programación científica con Python, que proporciona un **objeto array n-dimensional**, para almacenar datos de forma eficiente, así como **funciones para trabajar con estos arrays**.

También incluye un módulo de álgebra lineal (**linalg**) para calcular; determinante, inversa, autovalores y autovectores, etc.



NumPy - Qué es un array

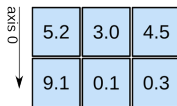
1D array



axis 0 →

shape: (4,)

2D array

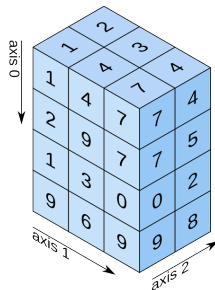


axis 0 ↓

axis 1 →

shape: (2, 3)

3D array



axis 0 ↓

axis 1 →

axis 2 ↗

shape: (4, 3, 2)

NumPy - Creación de arrays (I)

- Básica \Rightarrow `array`, `asarray`
- Valores \Rightarrow `ones`, `zeros`, `full`, `empty`
- Valores y forma \Rightarrow `ones_like`, `zeros_like`, `full_like`, `empty_like`
- Disposiciones especiales \Rightarrow `identity`, `eye`, `diag`
- Secuencias \Rightarrow `arange`, `linspace`, `logspace`
- Aleatorios \Rightarrow `numpy.random.randn`, `numpy.random.randint...`
- Desde fichero \Rightarrow `loadtxt` (cuando no faltan valores),
`genfromtxt` (cuando sí faltan valores)

NumPy - Creación de arrays (II)

```
>>> import numpy as np
>>> np.array([1, 2+3j, 'hi', 7.5])
array(['1', '(2+3j)', 'hi', '7.5'], dtype='<U64')
>>> np.array([1, 2+3j, 7.5])
array([1. +0.j, 2. +3.j, 7.5+0.j])
```

¡¡Todos los elementos de un array son del mismo tipo!!

Si se mezclan tipos en su creación, se usa el más restrictivo. Esto recibe el nombre de upcasting.

NumPy - Operaciones con arrays

```
>>> import numpy as np
>>> a = np.array([[1, 2],
                  [3, 4]])
>>> b = np.array([[4, 3],
                  [2, 1]])
>>> a * b          # Multiplicacion elemen-wise
array([[4, 6],
       [6, 4]])
>>> a @ b          # Multiplicacion matricial
array([[ 8,  5],
       [20, 13]])
```

Las operaciones; +, -, *, /, ... entre arrays son elemento a elemento

La multiplicación matricial puede hacerse con `np.dot(a, b)` o con `@`

NumPy - Slicing (I)

- El slicing permite extraer secciones de un array.

```
array[indice_inicial:indice_final:paso])
```

```
>>> import numpy as np
>>> a = np.array([[1, 2, 3, 4],
                  [5, 6, 7, 8],
                  [9, 10, 11, 12]])
>>> a[0:3:2, ::3]
array([[1,    4],
       [ 9,   12]])
```

NumPy - Slicing (II)

```
>>> b = a[0:3:2, ::3]
array([[1,      4],
       [  9,    12]])
>>> b[0, 0] = 9999
>>> b
array([[9999,      4],
       [  9,    12]])
>>> a
array([[9999,      2,      3,      4],
       [  5,      6,      7,      8],
       [  9,     10,     11,     12]])
```

!!Slicing devuelven una VIEW del array y NO una copia!!

Si se modifican elementos de esa VIEW, se modifican también en el array inicial. Para que esto no suceda, hay que usar `.copy()`

NumPy - Reshape (I)

- La función reshape permite ver el array con una nueva forma.

```
>>> import numpy as np
>>> c = np.arange(6)
>>> c
array([0, 1, 2, 3, 4, 5])
>>> np.reshape(c, (2, 3))
array([[0, 1, 2],
       [3, 4, 5]])
```

¡¡El reshape devuelve una VIEW del array inicial y NO una copia!!

Si se modifican elementos del reshape, se modifican en el array inicial. Para que esto no suceda, hay que usar `.copy()`

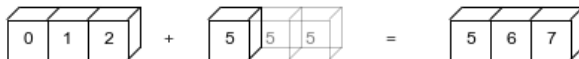
NumPy - Reshape (II)

- Utilización de reshape dejando 'libre' una dimensión

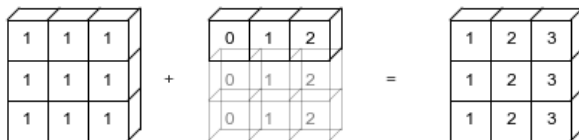
```
>>> np.reshape(c, (-1, 3))    # -1 dim libre  
array([[0, 1, 2],  
       [3, 4, 5]])  
>>> np.reshape(c, (-1, 2))  
array([[0, 1],  
       [2, 3],  
       [4, 5]])
```

NumPy - Broadcasting

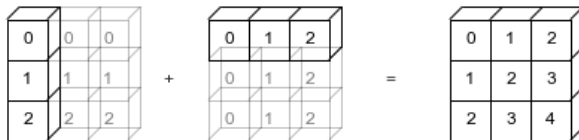
`np.arange(3)+5`



`np.ones((3,3))+np.arange(3)`



`np.arange(3).reshape((3,1))+np.arange(3)`



<https://jakevdp.github.io/PythonDataScienceHandbook/02.05-computation-on-arrays-broadcasting.html>

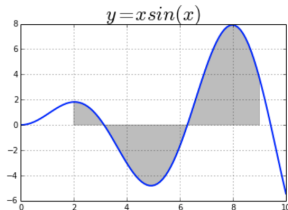
SciPy es un paquete de Python que proporciona **funciones y algoritmos matemáticos** destinadas al cálculo científico e ingenieril. Está construido sobre el paquete NumPy, y se estructura en diferentes submódulos; **constants, linalg, optimize, etc.**



SciPy - Submódulos disponibles en SciPy

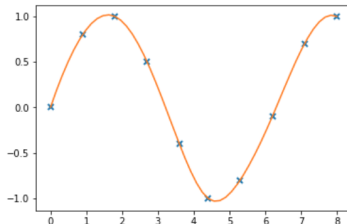
- Funciones especiales (`scipy.special`)
- Integración (`scipy.integrate`)
- Optimización (`scipy.optimize`)
- Interpolación (`scipy.interpolate`)
- Transformadas de Fourier (`scipy.fftpack`)
- Procesamiento de señal (`scipy.signal`)
- Álgebra lineal (`scipy.linalg`)
- Matrices sparse (`scipy.sparse`)
- Estadística (`scipy.stats`)
- ...

SciPy - Integración



```
>>> from scipy import integrate
>>> def fun(x): return x*np.sin(x)
>>> integrate.quad(fun, 2, 9)
(6.870699742283883, 2.864870105641461e-13)
>>> x = np.linspace(2, 9, 100)
>>> integrate.trapz(fun(x), x)
6.870667562274471
>>> integrate.simps(fun(x), x)
6.870699624669227
```

SciPy - Interpolación

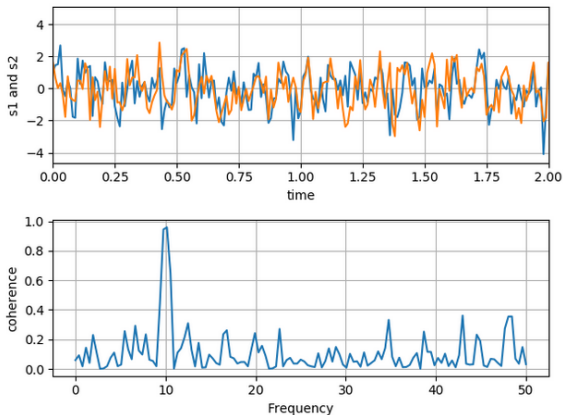


```
>>> from scipy import interpolate
>>> x_i = [0., 0.9, 1.8, 2.7, 3.6, 4.4, 5.3, 6.2, 7.1, 8.]
>>> y_i = [0., 0.8, 1., 0.5, -0.4, -1., -0.8, -0.1, 0.7, 1.]
>>> f_interp = interpolate.interp1d(x_i, y_i, kind='cubic')
>>> x = np.linspace(0, 8)
>>> y = f_interp(x)
```

`matplotlib` es un paquete de **representación gráfica 2D** con Python, que permite realizar figuras en una gran variedad de formatos, incluidos entornos interactivos (e.g. jupyter notebook). Es muy interesante su documentación, donde hay una galería con un gran número de ejemplos y el código que los genera.



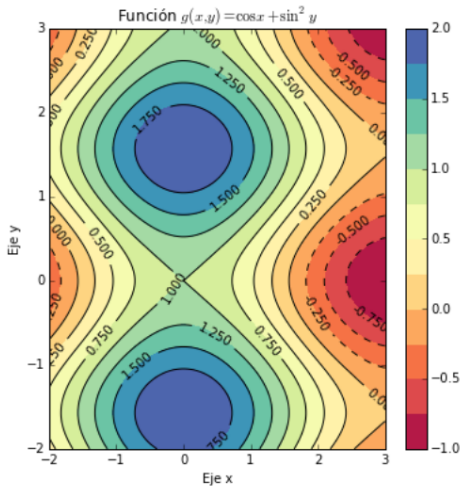
matplotlib - Subplots



- subplots
- xlabel, ylabel
- tight_layout

https://matplotlib.org/gallery/lines_bars_and_markers/cohere.html#sphx-glr-gallery-lines-bars-and-markers-cohere-py

matplotlib - Mapa de contorno



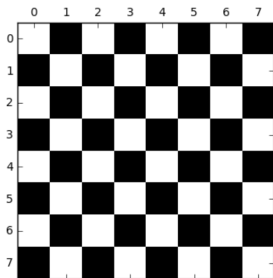
- `contour`
- `contourf`
- `colorbar`
- `cmap`
- `clabel`

Además...

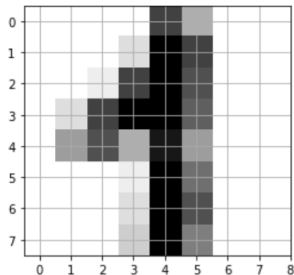
- `np.linspace`
- `np.meshgrid`

https://github.com/AeroPython/Curso_AeroPython

matplotlib - Matrices e Imágenes



- `matshow`
- `cmap` invertido (`gray_r`)

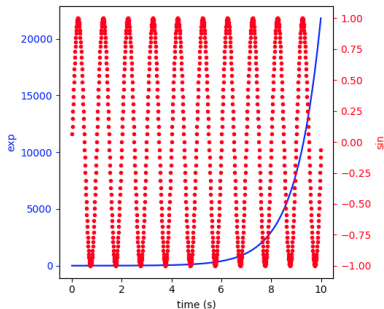


- `imshow`
- `grid`
- `xticks`, `yticks`

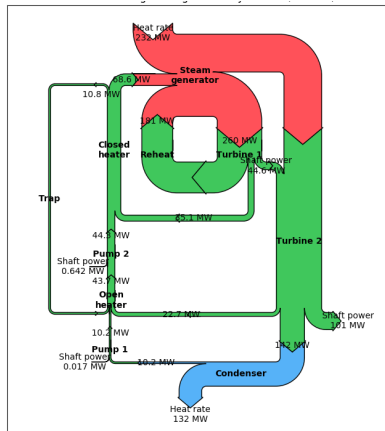
https://github.com/AeroPython/Curso_AeroPython

[https://matplotlib.org/gallery/images_contours_and_fields/
interpolation_methods.html](https://matplotlib.org/gallery/images_contours_and_fields/interpolation_methods.html)

matplotlib - Curiosidades



- twinx
- color yticks
- color ylabel



- sankey

https://matplotlib.org/examples/api/two_scales.html

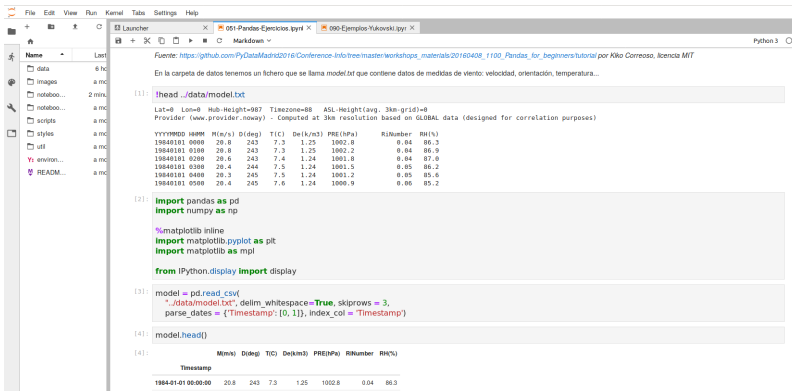
https://matplotlib.org/examples/api/sankey_demo_rankine.html

Jupyter notebook / JupyterLab - Introducción

[Jupyter notebook](#) y [JupyterLab](#) son dos entornos interactivos que permiten crear y compartir documentos que contienen código, texto, visualizaciones, etc. Ambos pertenecen al proyecto jupyter, y aunque pueden ejecutar dentro distintos kernels, se necesita haber instalado jupyter previamente con Python.



Jupyter notebook / JupyterLab - Interfaz



The screenshot shows a JupyterLab window with a file browser on the left and a notebook editor on the right. The notebook contains the following code cells:

```
[1]: !head ./data/model.txt
```

Lat=0 Lon=0 Hub-Height=987 Timezone=88 ASL-Height(avg, 3km-grid)=0
Provider (www.provider.noway) - Computed at 3km resolution based on GLOBAL data (designed for correlation purposes)

YYYYMMDD	WMM	H(s/s)	D(deg)	T(C)	De(k/m3)	PRE(hPa)	RjNumber	RH(%)
19840101	0000	20.8	243	7.3	1.25	1002.8	0.04	86.3
19840101	0100	20.8	243	7.3	1.25	1002.2	0.04	86.9
19840101	0200	20.6	243	7.4	1.24	1001.8	0.04	87.0
19840101	0300	20.4	244	7.5	1.24	1001.5	0.05	86.2
19840101	0400	20.3	245	7.5	1.24	1001.2	0.05	85.6
19840101	0500	20.4	245	7.6	1.24	1000.9	0.06	85.2

```
[2]: import pandas as pd
import numpy as np

%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib as mpl

from IPython.display import display

[3]: model = pd.read_csv(
    './data/model.txt', delim_whitespace=True, skiprows = 3,
    parse_dates = {'Timestamp': [0, 1]}, index_col = 'Timestamp')

[4]: model.head()
```

Timestamp	M(m/s)	D(deg)	T(C)	De(k/m3)	PRE(hPa)	RjNumber	RH(%)
1984-01-01 00:00:00	20.8	243	7.3	1.25	1002.8	0.04	86.3

https://github.com/AeroPython/Curso_AeroPython

- Formado por **celdas** que ejecuta el kernel.
- **Diferentes lenguajes** para el kernel; Python, R, Scala...
- Celdas de **código, texto (markdown), imágenes, vídeos...**
- **Plots** se representan integrados en el notebook.
- Representaciones interactivas (**widgets**).
- **Interactuar con la terminal** del sistema operativo (! < *comando* >). e.g. !head o !type, !ls o !dir, ...
- **'Magic commands'** que dan funcionalidad extra (% < *comando* >). e.g. %matplotlib notebook, %timeit, ...
- Ejemplos de uso: aprendizaje, prototipado, representación de datos (plots y widgets), preparación lectura de ficheros ...

¿Dudas? ¿Preguntas?



¡Gracias por asistir!