Preparar nuestro entorno de trabajo

Objetivo

Instalar Anaconda en nuestras máquinas

Comprender qué son IPython, Jupyter y los notebooks.

Aprender a utilizar el notebook con efectividad.

Conocer el concepto de *Literate Programming*

Anaconda

Anaconda es una **distribución** de Python.

Especialmente enfocada a Data science, instala por defecto muchos paquetes que nos serán muy útiles.

Tiene 3 aspectos que nos facilitan la vida:

- Distribución con ~150 paquetes ampliamente utilizados
- El gestor de paquetes conda
- Gestión integrada de dependencias y entornos (*environments*)

Anaconda

Vamos a instalar Anaconda en nuestras máquinas.

Dirigíos a https://www.anaconda.com/download/ y descargad la versión apropiada para vuestro sistema operativo.

Descargad la versión de **Python 3**

[![Anaconda](figs/anaconda-screenshot.png)](https://www.anaconda.com/download/)

Mientras descarga, vamos a comentar qué es y por qué nos interesa

Conda

conda es el **gestor de paquetes y entornos** de Anaconda. Es por tanto sólo un componente de la distribución.

Nos permite instalar fácilmente paquetes que tienen, por ejemplo, dependencias escritas en otros lenguajes.

Por ejemplo, **numpy**

numpy es un paquete de computación numérica que usaremos mucho en el máster.

Aquí tenéis un ejemplo de los problemas que suelen surgir al intentar instalarlo por nuestra cuenta.

Ventajas de **conda**

Podemos utilizarlo en modo usuario.

Simplifica la gestión de dependencias externas a Python

Nos permite crear entornos de trabajo aislados para distintos proyectos

Uso de conda

En este curso, casi exclusivamente vamos a usar:

```
conda install package
```

Instala la version más reciente del paquete solicitado. Esto nos valdrá el 95% de las veces.

```
conda search package
```

Nos permite buscar un paquete. Útil si no lo encontramos por defecto.

```
conda env export > environment.yml
```

Exporta al archivo environment.yml la descripción del entorno actual

Environments

Los environments nos permiten mantener aisladas las dependencias de distintos proyectos que conviven en una máquina.

Nos permiten también describir a los usuarios de nuestro proyecto qué necesitan para correr nuestro código

Instalar Anaconda

La descarga debería haber terminado. Ahora, seguid las instrucciones en la página de Anaconda para vuestro sistema operativo: Linux, macOS o Windows.

Qué es una IDE?

***I**ntegrated **D**evelopment **E**nvironment*.

Una IDE normalmente cuenta con:

- Editor de código fuente
- Explorador de variables
- Línea de comandos del intérprete
- Historial de comandos
- Explorador de ficheros
- Depurador
- Otras herramientas

Spyder

Una IDE pensada especialmente para el desarrollo de Data Science.

En este curso no la vamos a utilizar, pero vosotros deberíais probarla y decidir si la preferís.

Ipython

Interactive Python

Es un *intérprete* alternativo al estándar

Creado por Fernándo Pérez, entonces doctorando en Física

Diseñado para facilitar un estilo de exploración

Literate Programming

Literate programming is a programming paradigm introduced by Donald Knuth in which a program is given as an explanation of the program logic in a natural language, such as English, interspersed with snippets of macros and traditional source code, from which a compilable source code can be generated.

De https://en.wikipedia.org/wiki/Literate_programming

Literate programming: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.

Donald Knuth (1984)

Literate Programming en Data Science

El objectivo del programador en Data Science es tanto generar modelos como generar explicaciones.

Los modelos pueden ser programas para ser ejecutados en un ordenador, pero las explicaciones tienen como público otros humanos.

Incluso en el caso de que sólo estemos generando modelos para consumo vía API, por ejemplo, tener juntos código y resultados nos permite comunicar con efectividad y precisión los detalles de nuestro análisis.

Por ello el paradigma del literate programming se adapta especialmente bien a la Data Science.

Todavía más si estamos en un entorno didáctico!

El notebook

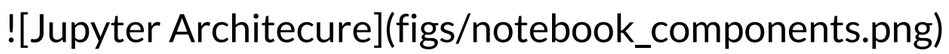
La herramienta que nos va a permitir desarrollar de acuerdo con este paradigma es el *notebook*

https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks

El notebook es un formato de archivo y toda una arquitectura asociada que nos permite mostrar, en un mismo documento:

- Texto formateado con Markdown
- Fórmulas matemáticas formateadas con $\LaTeX$
- Código, originalmente sólo en Python pero actualmente en muchos lenguajes
- Los resultados de evaluar ese código
- Figuras

El notebook



Cómo ejecutamos un notebook

Tecleamos en nuestra terminal:

```
jupyter notebook
```

Esto lanza el servidor de notebooks. Para comunicarnos con él sólo necesitamos introducir la url que nos indica en un navegador web.

Veremos un listado de archivos y directorios contenidos **en el directorio desde donde lanzamos el servidor**

A partir de ahí, podemos crear nuevos notebooks o abrir alguno ya existente que nos hayamos descargado

El notebook

El notebook se ordena en celdas.

Cada celda es una unidad de ejecución independiente. Las hay de dos tipos: de Markdown y de código.

Lo más importante que hay que recordar es el notebook es **modal**. Tiene dos modos:

- En *Edit mode*, modificamos el contenido de las celdas: cortamos y pegamos texto o código, ejecutamos celdas...
- En *Command mode*, cambiamos el tipo de celda, cortamos y pegamos celdas, añadimos celdas...

Al usarlo es importante recordar en qué modo estamos y que en *command mode* al pulsar la tecla h obtenemos una lista de atajos de teclado

In [3]:
```
r = 2
pi = 3.1416
A = pi * r ** 2

print(A)
```

```
12.5664
```

El notebook

Es importante también, cuando estamos elaborando o interpretando un notebook, fijarse en el *orden de ejecución* de las celdas.

Aparece a la izquierda de las celdas de código como In [n] y Out [n]. Este último sólo aparecerá si la evaluación de la última celda devuelve un valor distinto de None.

Markdown

Es un lenguaje de markup, como HTML

Fácil de leer para humanos en su forma cruda

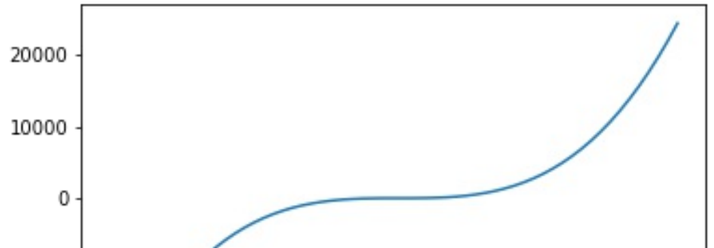
Plotear con matplotlib

El notebook nos permite incorporar fácilmente gráficos en línea, junto al código que los genera.

Matplotlib es la librería más común de gráficos en Python, pero no es ni mucho menos la única.

In [5]:
```
# A matplotlib example:

import matplotlib.pyplot as plt
%matplotlib inline


xs = range(-30,30)
ys = [x**3 for x in xs]



plt.plot(xs, ys)
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x7f36ecec5390>]
```



LaTeX en el notebook

$\LaTeX$ es un lenguaje de markup especialmente diseñado para el maquetado de ecuaciones matemáticas, muy usado en la academia

No lo vamos a aprender en este curso en detalle

Nos permite escribir ecuaciones como ésta:

$$\mu = \frac{\sum_{i=1}^{n} x_i}{n}$$

Para saber más

https://docs.continuum.io/anaconda/

https://stackoverflow.com/questions/42096280/how-is-anaconda-related-to-python

https://conda.io/docs/user-guide/tasks/manage-environments.html

http://www.literateprogramming.com/knuthweb.pdf (PDF)

https://en.wikipedia.org/wiki/Literate_programming

http://www.literateprogramming.com/

https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks

https://www.youtube.com/watch?v=g8xQRI3E8r8