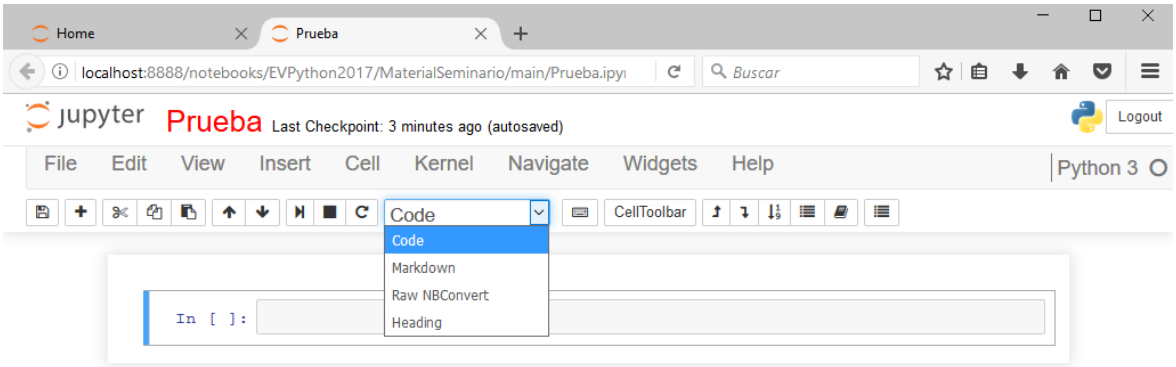
Introducción a Jupyter Notebook

Los notebooks de Jupyter son archivos con extensión .ipyb. En ellos podemos escribir código python ejecutable, texto, dibujar gráficas y mucho más. ¡Este entorno te va a enamorar!.

Un notebook de IPython es un fichero que contiene un conjunto de celdas donde cada celda puede ser de distintos tipos:

- **Markdown**: sirven para escribir texto.



- **Code**: sirven para escribir código ejecutable. Están marcadas por la palabra **In [n]** y están numeradas. El resultado de la ejecución de cada celda se muestra en el campo **Out[n]**, también numerado.

A continuación se muestra un ejemplo de una celda de código:

```
# celda de código. Esta línea es un comentario
3 + 4
```

```
7
```

```
3 + 4
```

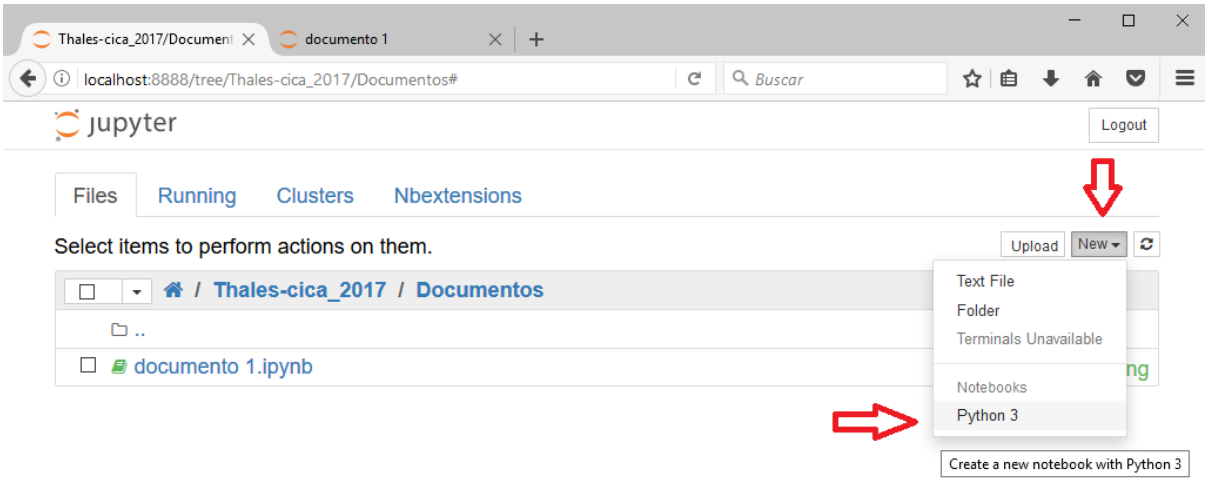
```
7
```

Como podemos ver en el ejemplo anterior, ambas celdas tienen el mismo número, indicando que la segunda es el resultado de la primera. La primera celda **In [ ]** contiene código a ejecutar y la segunda celda **Out[ ]** se genera automáticamente como resultado de la ejecución de la primera.

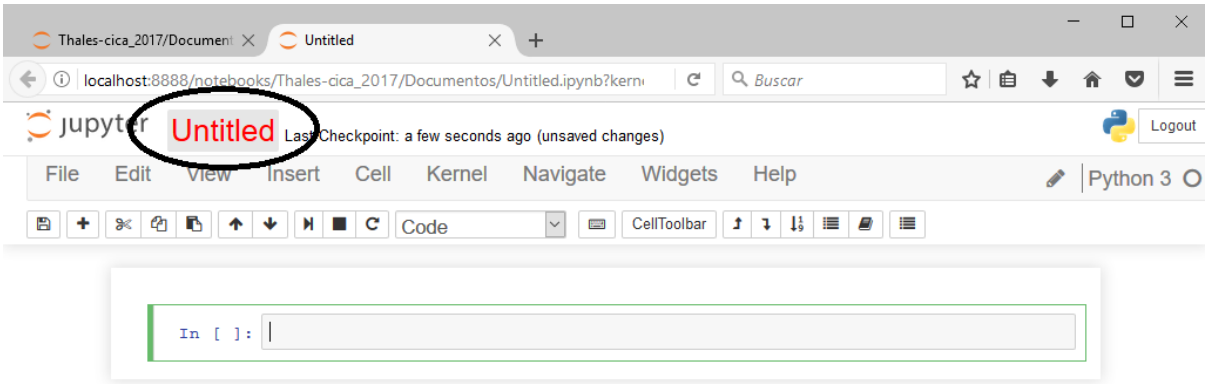
Los **comentarios** dentro de secciones de código se hace poniendo el símbolo de gato, #, al inicio de la línea.

- **Raw NBConvert**: Son celdas que permiten escribir fragmentos de código sin que sean ejecutados.

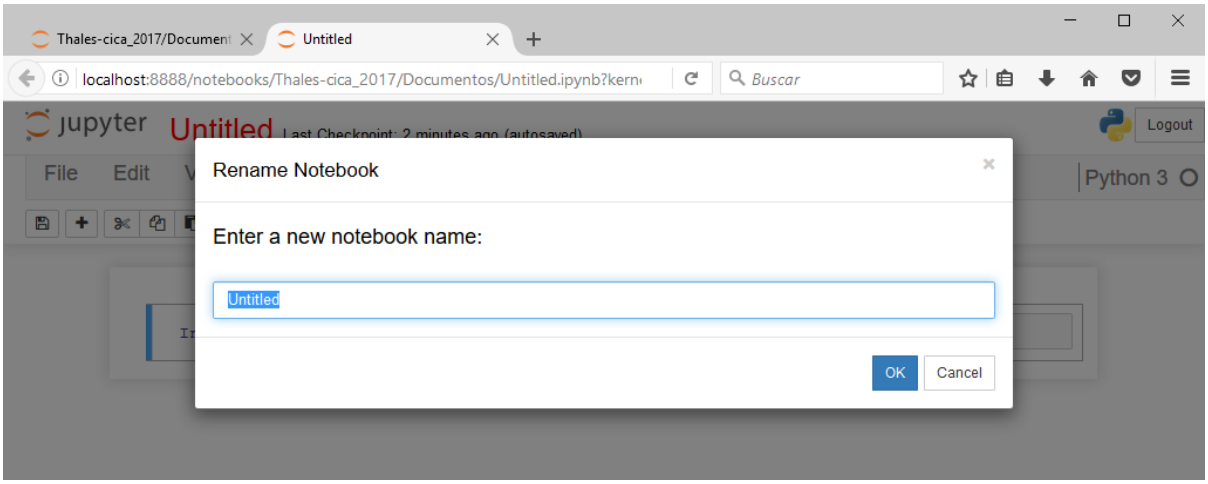
Para crear un nuevo notebook es necesario desplegar el botón `New` y posteriormente seleccionar *Python 3*.



El efecto es que se abrirá otra pestaña en el navegador con un documento nuevo y sin nombre. Mejor dicho Jupyter le da un nombre por defecto (untitled) y que nosotros debemos cambiar una vez creado.



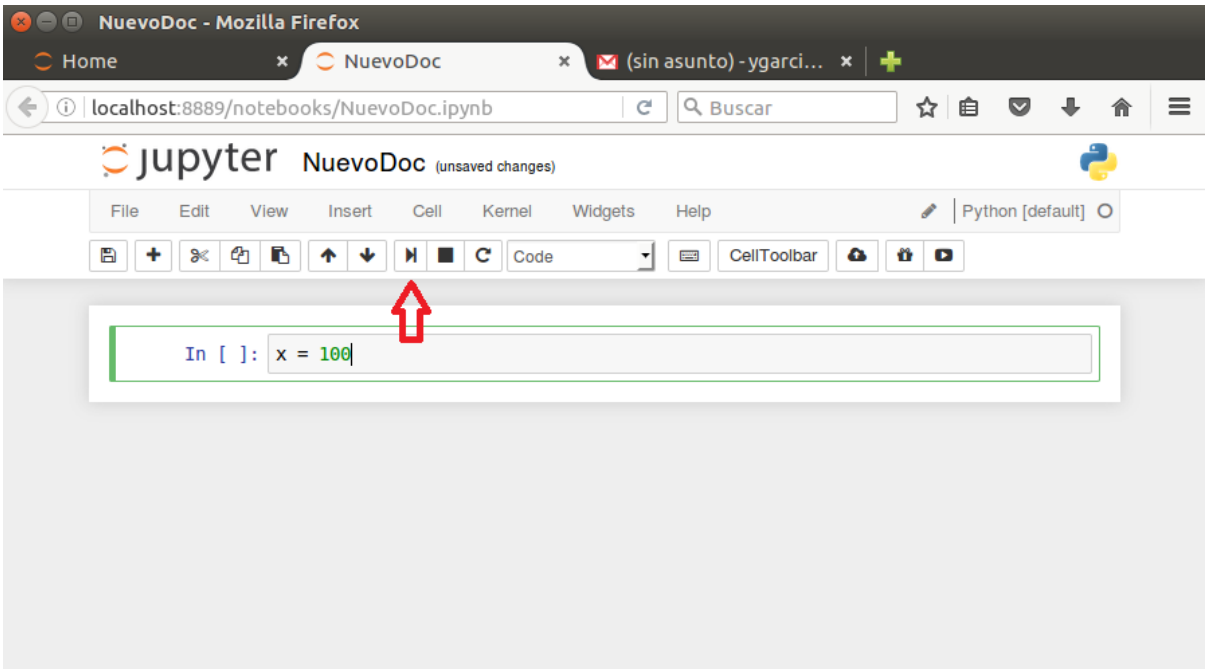
Para cambiar el nombre es necesario pinchar sobre la palabra `Untitled` y aparecerá una nueva ventana donde podremos escribir el nuevo nombre.



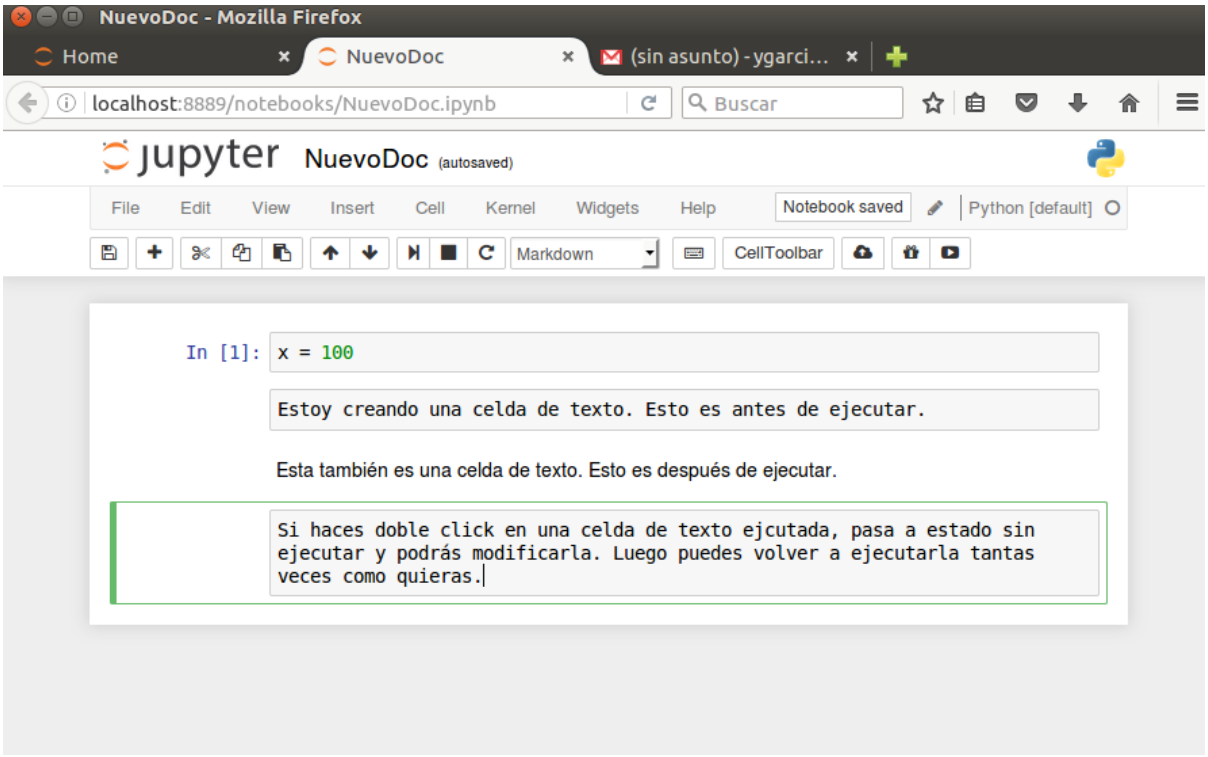
Ejecución de las celdas

Todas las celdas son susceptibles de ser ejecutadas. La ejecución de una celda de código Python, ejecutará el código escrito en la celda y producirá una salida, que será el resultado de la ejecución. La ejecución de celdas de tipo Markdown, dará formato al texto.

Para ejecutar una celda tienes que posicionarte en la celda y posteriormente pulsar el botón cuyo icono es un triángulo mirando a la derecha.



Para crear celdas nuevas pulsaremos el botón con el icono del signo +. En la siguiente imagen puedes ver distintas celdas de texto. La primera celda es de código, la segunda y cuarta son celdas de texto sin ejecutar, mientras que la tercera es una celda de texto ya ejecutada. Si haces doble click en una celda ya ejecutada, pasa a ser una celda no ejecutada y podrás modificarla añadiendo o eliminando texto. Luego podrás volver a ejecutarla.



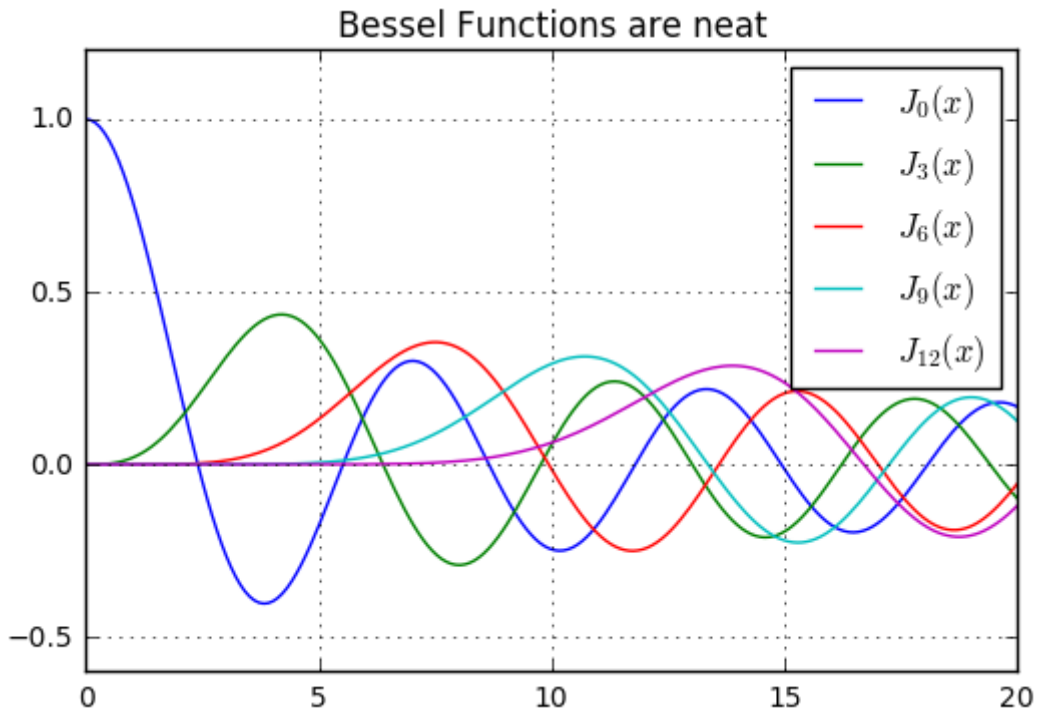
Como puedes observar, el menú es bastante intuitivo. Navega un poco por él a ver qué descubres.

¿Qué se puede hacer en un Notebook de IPython?

Dibujar gráficas

```
import scipy.special as spec
%pylab inline
x = np.linspace(0, 20, 200)
for n in range(0,13,3):
    plt.plot(x, spec.jn(n, x), label=r'$J_{%i}(x)$' % n)
grid()
legend()
title('Bessel Functions are neat');
```

```
Populating the interactive namespace from numpy and matplotlib
```



Incrustar Imágenes

```
from IPython.display import Image
Image(filename='./images/setas.jpg')
```

Incrustar vídeos

```
from IPython.display import YouTubeVideo
YouTubeVideo('pQNG9ePYSSc')
```

Escribir y representar código HTML

```
from IPython.display import HTML
s = """<table>
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>"""
h = HTML(s)
h
```

Header 1	Header 2
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Escribir código científico

```
from IPython.display import Math
Math(r'F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx')
```

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{2\pi i k} dx$$

$$c = \sqrt{a^2 + b^2}$$

$$c = \sqrt{a^2 + b^2}$$

Características del entorno

- **Introspección**: Usando el símbolo `?` es posible conocer información de un objeto:

```
b = [1,2,3]
b?
# Para probarlo ejecutar esta celda
```

- Si usamos el símbolo `?` sobre un método, se visualiza la información de lo que hace el método:

```
b.append?
# Para probarlo ejecutar esta celda
```

- Y si usamos dos símbolo `??`, se muestra el código del método cuando sea posible.
- También se puede combinar con el símbolo `*` para realizar búsquedas:

```
b.*p*?
# Para probarlo ejecutar esta celda
```

Uso del tabulador (tab completion)

- Mientras se escribe una expresión, si se pulsa el tabulador IPython ofrece ayuda para completar la expresión.
- El uso del tabulador también sirve para buscar los métodos de un objeto
- También se usa para completar rutas de ficheros.

```
b.remove?
```

```
b.append(2)
b
```

```
[1, 2, 3, 2]
```

Comandos mágicos

- Son comandos especiales que permiten al usuario tener un control del comportamiento del sistema.
- Están precedidos por el símbolo `%`.
- Para listar todas las funciones mágicas, basta con escribir el comando `%magic`:

```
%magic
# Para probarlo ejecutar esta celda
```

Ejemplos:

%reset : Limpia el espacio de trabajo actual y borra todos los nombres definidos por el usuario. Cuando traajamos con grandes volúmenes de datos, Jupyter los mantiene en memoria incluso después de hacer `%del`, por lo que no se libera memoria. En estos casos resulta útil.

%time : Sirve para medir el tiempo de ejecución de una instrucción . Ejecuta la instrucción una sola vez y mide el tiempo empleado. Muy útil para comparar procesos con la misma finalidad pero que utilizan algoritmos diferentes.

%timeit : Permite obtener una medida más precisa. En este caso, se ejecuta la instrucción varias veces y se calcula la media de los tiempos.

- Cuando se trabaja con grandes volúmenes de datos, unos pocos milisegundos son muy importantes.

```
b
%reset
## Para probarlo ejecutar esta celda
```

```
b
## Para probarlo ejecutar esta celda para ver qué ha ocurrido con la variable b
```

```
%timeit 3+4
```

```
%time 3+4
```

```
time 3+4
```

SI multiples for second (s)

Submultiples			Multiples		
Value	SI symbol	Name	Value	SI symbol	Name
10^{-1} s	ds	decisecond	10^{1} s	das	decasecond
10^{-2} s	cs	centisecond	10^{2} s	hs	hectosecond
10^{-3} s	**ms**	**millisecond**	10^{3} s	ks	kilosecond
10^{-6} s	**µs**	**microsecond**	10^{6} s	Ms	megasecond
10^{-9} s	**ns**	**nanosecond**	10^{9} s	Gs	gigasecond
10^{-12} s	ps	picosecond	10^{12} s	Ts	terasecond
10^{-15} s	fs	femtosecond	10^{15} s	Ps	petasecond
10^{-18} s	as	attosecond	10^{18} s	Es	exasecond
10^{-21} s	zs	zeptosecond	10^{21} s	Zs	zettasecond
10^{-24} s	ys	yoctosecond	10^{24} s	Ys	yottasecond
Common prefixes are in bold					

Comunicación con el SO

- Es posible ejecutar comandos que se utilizan en las consolas de Windows o Linux.

> **%pwd** : devuelve el nombre del directorio de trabajo
>
> **%cd dir** : cambia el directorio actual de trabajo
>
> **%ls** : devuelve el contenido del directorio actual

```
%pwd
## Para probarlo ejecutar esta celda
```

Referencias

- Jupyter notebook documentation
- IPython's Rich Display System