


# 0 - Introducción a Jupyter

## 0.1 - Corriendo código de Python

En su versión mas sencilla, cada una de las celdas se puede pensar como un **snippet de código que se puede ejecutar independientemente** de las demas. Para ejecutar la siguiente celda, selccionarla y apretar el boton  ubicado en la barra superior, o equivalentemente, presionar **Shift** + **Enter**

```
In [2]: print('Hola notebook!')
```

Hola notebook!

Pueden ver una lista completa de todos los shortcuts disponibles en el menu **Help** **Keyboard Shortcuts** o con el shortcut **H**

Shortcuts importantes:

- Click para seleccionar una celda
- **Shift** + **Enter** para ejecutar una celda y seleccionar la celda siguiente

Cada celda es independiente de las demás, pero **comparten el mismo contexto**. Definiciones de variables, funciones y clases, e importaciones de módulos dentro de una celda, permiten que las siguientes celdas tengan acceso a éstas.

Si no sabes que son las variables, funciones y clases, tranquilo sólo quedate con que: *lo que ejecutes desde la primera celda, estará disponible para usar en las siguientes celdas*

```
In [3]: import getpass
# defino variable en una celda
user = getpass.getuser()
```

```
In [4]: # accedo a la variable definida en la celda anterior
print ("Hola {}".format(user))
```

Hola fsanmartin!

Una diferencia notable con un script.py es que **no estamos obligados a ejecutar celdas en un orden preestablecido**. De todas maneras, es recomendable ordenar un notebook según el orden de lectura estandar para evitar confusiones.

## 0.2 - Más que solo Python

El contenido de las celdas no está limitado a solo código de Python. Una alternativa posible es **Markdown**, que es muy útil para explicar las ideas detras de cada snippet de código y para guiar al lector en la ejecución del notebook (como venimos haciendo hasta ahora).

Y extendiendolo aún mas, es posible insertar código html para insertar imágenes, audios, videos y todas las alternativas que esto provee. Esto es especialmente útil para visualizar información y generar gráficos con datos generados por los snippets de código del notebook.

En modo Markdown está habilitada la interpretación de código html, con algunas excepciones por temas de seguridad y consistencia del notebook (ej: interpretación de `<script>`, `<style>` y atributos relacionados).

## 0.3 - Integración con bash

De forma nativa jupyter tiene integración con comandos de bash. Estos son reconocidos mediante el prefijo `!`. Por ej, si quisiera ver qué versión de python estoy corriendo, podría correr:

```
In [6]: !python --version
```

```
Python 3.7.3
```

Notar que la celda anterior es interpretada como una celda de código al igual que las celdas de que contiene código python. Esto permite que dentro de una celda conviva código de python y de bash sin problemas.

Para ejemplificar esto, usemos el comando **ls** que lista los archivos de la carpeta en la cual estemos parados:

```
In [8]: files_in_this_folder = !ls
        for filename in files_in_this_folder:
            if filename[-6:] == '.ipynb':
                print(filename)

0 - IntroducciÃ³n a Jupyter.ipynb
01_Cargar un Zip en Jupyter.ipynb
02_Limpieza_de_Datos.ipynb
03_Data_Wrangling.ipynb
04_Conceptos_bÃ¡sicos_de_estadÃsticas.ipynb
05_RegresiÃ³n_Lineal.ipynb
06_RegresiÃ³n_LogÃstica.ipynb
07_Clustering_y_clasificaciÃ³n.ipynb
```

## 0.4 - Jupyter Magic

Extendiendo la capacidad de jupyter de interpretar Python, Bash y Markdown, existen comandos especiales que son parte de la *magia de jupyter*. Estos comandos empiezan con `%` - para evaluaciones en una sola línea - y con `%%` para evaluaciones multi línea. Un ejemplo de esto es la posibilidad de insertar html excediendo las capacidad de Markdown y tener la posibilidad de insertar scripts de javascript:

```
In [9]: %%html
<button id="click-button">Clickeame</button>
<span id="click-counter">No me has clickeado todavía.</span>
<script>
    $(function() {
        var click_counter = 0;
        $('#click-button').click(function() {
            click_counter += 1;
            var message = 'Me haz clickeado ' + click_counter + (click_counter
== 1 ? ' vez' : ' veces')
            $('#click-counter').text(message)
        })
    })
</script>
```

Clickeame No me has clickeado todavía.

Para listar por completo todas las magias con las que viene jupyter, existe el comando `% lsmagic`

```
In [ ]: % lsmagic
```

## 0.5 - Servidor remoto

La principal interfaz de Jupyter involucra un navegador para proveer la interfaz al usuario, pero la ejecución de código se realiza del lado del servidor. Esto permite que, habiendo hecho las configuraciones pertinentes, el usuario acceda de forma remota al servidor. Por ej, en la necesidad de procesar una gran cantidad de información, se puede dejar corriendo un servidor de jupyter en una computadora con mejores recursos de hardware y acceder a éste desde cualquier computadora que cuente con un navegador y una conexión a internet.

Para mas información, leer la [documentación oficial de jupyter \(http://jupyter-notebook.readthedocs.io/en/latest/public\\_server.html\)](http://jupyter-notebook.readthedocs.io/en/latest/public_server.html) sobre este tema

## 0.6 - Extensiones de la comunidad

Con la rápida adopción y la creciente comunidad de usuarios de Jupyter, han aparecido una gran cantidad de extensiones que le agregan funcionalidades. Éstas se encuentran en el [repositorio de extensiones](https://github.com/ipython-contrib/jupyter_contrib_nbextensions) ([https://github.com/ipython-contrib/jupyter\\_contrib\\_nbextensions](https://github.com/ipython-contrib/jupyter_contrib_nbextensions)) de Jupyter.

Las extensiones disponibles ofrecen un amplio rango de funcionalidades agregadas. Desde correctores ortográficos hasta generadores automáticos de tablas de contenidos:



## Referencias

- <https://github.com/datosgobar/taller-analisis-datos-101> (<https://github.com/datosgobar/taller-analisis-datos-101>)

In [ ]: