0-intro_python

October 22, 2020

1 Introducción a IPython

1.1 Cosas que debes de hacer para las clases siguientes

Nota: siempre consulta el `README.md` del curso para ver las fechas de entrega y el formato cuando no estén especificados.

1.1.1 Proyecto Euler:

- Multiples of 3 and 5
- Even Fibonacci numbers

1.1.2 Actividades recomendadas:

- Primeros seis módulos de Codecademy de *python* para el Jueves 4 de septiembre, 2020.
- Módulos del 7 al 11 de Codecademy de *python* para el Martes de 8 de septiembnre, 2020.

1.2 Instrucciones

- En **tu carpeta** (la que está dentro de `alumnos`) crea un archivo llamado `1-python_tour`.
- En la primera celda pon el formato `Heading 1` y teclea `Tour de Python`.
- Luego teclea (**no copies y pegues**) el código mostrado aquí y ve ejecutándolo.

1.3 1. Algunos términos

Python Lenguaje multiparadigma muy versátil que estaremos usando en el curso.

Ipython = *Interactive python*

- La documentación completa se encuentra aquí.

Ipython notebook donde estás tecleando, es una manera de visualizar y compartir programas de `python` en la web.

1.3.1 Otros términos

Termino	Significado
Interpretado	No es necesario (aunque es posible en el caso de `python`) ningún paso de compilación.
Objetos	Estructura de datos que contiene métodos (*comportamiento*) y atributos (*características*).

Termino	Significado
Dinámico	Las variables pueden cambiar de tipo durante la ejecución del programa.
Estructura de datos	Forma de manipular y/o guardar datos.
Script	Programa que controla otros programas.
Sintáxis	Gramática que define el lenguaje.
Librería	Colección de código (generalmente objetos o funciones) reutilizables. (En español el término correcto sería *biblioteca*)

1.4 2. Introducción rápida al Ipython notebook

Aquí practicaremos algunos elementos de la barra de menú

- **Shift + Enter** Ejecuta la celda actual y se mueve a la celda siguiente.
- **Ctrl + Enter** Ejecuta la celda actual y permanece en la celda actual.
- **Alt + Enter** Ejecuta la celda actual e inserta una celda inmediatamente después.

[4]:
```
print("Hola ¿Cómo estás?")
```

```
Hola ¿Cómo estás?
```

Para obtener ayuda teclea `?`

[5]:
```
?
```

1.5 3. Navegación con el teclado

El `ipython notebook` es un editor **modal**, es decir, la interacción del teclado con el editor depende del **modo** en el cual se encuentre. Existen dos modos: *modo de edición* y *modo de comando*.

1.5.1 Modo de edición

El *modo de edición* está indicado por una línea verde alrededor de la celda. Cuando está en este modo, la interacción con la celda es como si estuvieras en un editor de texto normal.

Puedes ingresar al modo de edición presionando `Enter` o haciendo click en la celda con el *mouse*.

Los comando del teclado en el modo de edición se muestran en la siguiente figura:

1.5.2 Modo de comando

En el *modo de comando* el `notebook` permite una navegación eficaz por toda la `notebook`. Está marcado por una línea gris alrededor de la celda.

`Se puede cambiar al modo de comando presionando la tecla` `` `Esc` ``.

Los comando del teclado en el modo de comando se muestra a continuación:

1.6 4. Salida Enriquecida

Es posible desplegar la salida de la ejecución de la celda de forma enriquecida, esto se logra usando la función `display()`, la cual es muy similar a la función `print` que usamos arriba. Para usarla, debemos importar la librería.

[6]:
```
from IPython.display import display
```

1.6.1 4.1 Imágenes

[7]:
```
from IPython.display import Image
```

[8]:
```
i = Image(filename = './imagenes/ipython_logo.png')
```

Regresar el objeto automáticamente lo despliega

[9]:
```
i
```

[9]:

O puedes usar explícitamente la función display

[10]:
```
display(i)
```

También puedes utilizar URL para mostrar imágenes

[11]:
```
Image(url='http://python.org/images/python-logo.gif')
```

[11]:
```
<IPython.core.display.Image object>
```

1.6.2 4.2 HTML

La manera más sencilla es usar *magia* de Ipython notebook para lograrlo

[12]:
```
from IPython.display import HTML
```

[13]:
```
%%html
<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>
```

```
<IPython.core.display.HTML object>
```

1.6.3 4.3 Javascript

[14]:
```
from IPython.display import Javascript
```

[15]:
```
%%javascript

alert("hi");
```

```
<IPython.core.display.Javascript object>
```

1.6.4 4.4 LaTeX

LaTeX se despliega en el ipython notebook usando MathJax

[16]:
```
%%latex
\begin{align}
\nabla \times \vec{\mathbf{B}} -\, \frac1c\,␣
 ↪\frac{\partial\vec{\mathbf{E}}}{\partial t} & =␣
 ↪\frac{4\pi}{c}\vec{\mathbf{j}} \\
\nabla \cdot \vec{\mathbf{E}} & = 4 \pi \rho \\
\nabla \times \vec{\mathbf{E}}\, +\, \frac1c\,␣
 ↪\frac{\partial\vec{\mathbf{B}}}{\partial t} & = \vec{\mathbf{0}} \\
\nabla \cdot \vec{\mathbf{B}} & = 0
\end{align}
```

$$\nabla \times \vec{\mathbf{B}} - \frac{1}{c}\frac{\partial \vec{\mathbf{E}}}{\partial t} = \frac{4\pi}{c}\vec{\mathbf{j}} \tag{1}$$

$$\nabla \cdot \vec{\mathbf{E}} = 4\pi\rho \tag{2}$$

$$\nabla \times \vec{\mathbf{E}} + \frac{1}{c}\frac{\partial \vec{\mathbf{B}}}{\partial t} = \vec{\mathbf{0}} \tag{3}$$

$$\nabla \cdot \vec{\mathbf{B}} = 0 \tag{4}$$

1.6.5 4.5 Audio

[17]:
```
from IPython.display import Audio
```

[19]:
```
Audio(url="http://en.wikipedia.org/wiki/File:Edison_cylinder_Lost_Chord.ogg")
```

[19]:
```
<IPython.lib.display.Audio object>
```

1.6.6 4.6 Videos

[20]:
```
from IPython.display import YouTubeVideo
```

[21]:
```
YouTubeVideo("iTT85DOwEXY")
```

[21]:

Existen otros formatos de salida enriquecidos, pero estos son los básicos.

1.7 5. Interacción enriquecida

También es posible interactuar con el notebook usando datos del usuario. En esta clase usaremos la función interact (IPython.html.widgets.interact) la cual crea elementos de automáticos de UI que permiten explorar los datos y el código interactivamente.

[22]:
```
from ipywidgets import interact, fixed
from IPython.html import widgets
```

```
/opt/conda/lib/python3.8/site-packages/IPython/html.py:12: ShimWarning: The
`IPython.html` package has been deprecated since IPython 4.0. You should import
from `notebook` instead. `IPython.html.widgets` has moved to `ipywidgets`.
  warn("The `IPython.html` package has been deprecated since IPython 4.0. "
```

De manera muy básica, interact generará una UI para los argumentos de una función. Cuando el usuario interactúe con el widget, interact invocará la función.

Declaremos una función llamada foo que sólo reciba un argumento y lo imprima

[23]:
```
def foo(x):
    print (x)
```

Invoquemos manualmente la función

[24]:
```
foo(5)
```

```
5
```

Si ahora usamos interact con la función foo como primer argumento y un entero como segundo, se generará automáticamente un *slider*

[25]:
```
interact(foo, x=10);
```

```
interactive(children=(IntSlider(value=10, description='x', max=30, min=-10), Output()), _dom_cl
```

Si el segundo parámetro es un boolean,obtendrás un *checkbox*

[26]:
```
interact(foo, x=True);
```

```
interactive(children=(Checkbox(value=True, description='x'), Output()), _dom_classes=('widget-
```

Si es una cadena de texto, un *text field*

[27]:
```
interact(foo, x=u"Hola ¿Cómo estás?");
```

```
interactive(children=(Text(value='Hola ¿Cómo estás?', description='x'), Output()), _dom_classes
```

Al pasar una *lista* como variable, obtenemos un *dropbox*

[28]:
```
interact(foo, x=["hola", "adios"]);
```

```
interactive(children=(Dropdown(description='x', options=('hola', 'adios'), value='hola'), Outpu
```

O si queremos pasar valores que no sean cadenas, usamos un *diccionario*

[30]:
```
interact(foo, x={"hola":1, "adios":2});
```

```
interactive(children=(Dropdown(description='x', options={'hola': 1, 'adios': 2}, value=1), Outp
```

Existen ocasiones donde quieres explorar el comportamiento de una función, pero manteniendo uno de sus argumentos fijo

[31]:
```
def bar(p, q):
    print (p, q)
```

[32]:
```
bar(10, 20)
```

```
10 20
```

[33]:
```
interact(bar, p=5, q=fixed(20));
```

```
interactive(children=(IntSlider(value=5, description='p', max=15, min=-5), Output()), _dom_clas
```

1.8 6. Usando el Sistema Operativo

[34]:
```
!pwd
```

```
/home/jovyan/Matematicas-computacionales-fall2020/alumnos/FerLango/2.Intro-
python
```

[35]:
```
!whoami
```

```
jovyan
```

[36]:
```
files = !ls
print ("Los archivos en esta carpeta son:")
print (files)
```

```
Los archivos en esta carpeta son:
['0-intro_python.ipynb', '1-python_tour.ipynb', 'imagenes',
'transformaciones.py']
```

1.9 7. Funciones mágicas

La ayuda sobre todas las funciones *mágicas* del ipython notebook pueden encontrase como sigue

[37]:
```
%magic
```

Por ejemplo, la función mágica %timeit ¿Qué crees que haga?

[38]:
```
%timeit range(10)
```

```
280 ns ± 74.7 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
```

Es posible ejecutar bash, ruby y otros lenguajes:

[39]:
```
%%bash
echo "Mi shell es:" $SHELL
echo "El uso de memoria es:"
free
```

```
Mi shell es: /bin/bash
El uso de memoria es:
              total        used        free      shared  buff/cache   available
Mem:        9682960      607896     8173340        1208      901724     8841436
Swap:       3145728           0     3145728
```

Es posible **crear archivos** localmente:

[40]:
```
%%file utilerias.py
def sort_string(s):
    s = sorted(s)
    print(''.join(s))
```

```
Writing utilerias.py
```

[41]:
```
!cat utilerias.py
```

```
def sort_string(s):
    s = sorted(s)
    print(''.join(s))
```

``Cargar'' un archivo desde disco

[43]:
```
# %load utilerias.py
def sort_string(s):
    s = sorted(s)
    print(''.join(s))
```

[44]:
```
def sort_string(s):
    s = sorted(s)
    print(''.join(s))
```

[45]:
```
from utilerias import sort_string
sort_string("GOAL")
```

```
AGLO
```

[47]:
```
#Elimina el archivo
!rm utilerias.py
```

```
rm: cannot remove 'utilerias.py': No such file or directory
```

1.10 8. Graficando

Aunque después vamos a dedicarle mucho tiempo a la librería matplotlib, es importante verla

[48]:
```
%matplotlib inline
```

[49]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[50]:
```
x = np.linspace(0, 2*np.pi, 300)
y = np.cos(x**3)
plt.plot(x, y)
plt.title(u"¡Hola desde plot!");
```