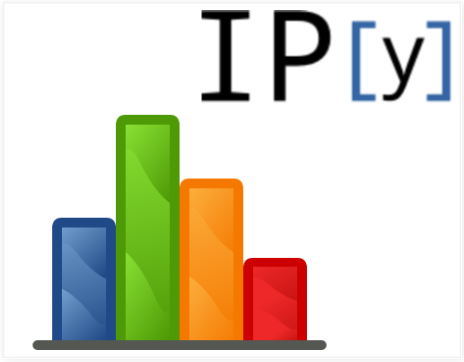
Python 3 para impacientes

"Simple es mejor que complejo" (Tim Peters)

jueves, 21 de agosto de 2014

Gráficos en IPython



Unos de los motivos que inspiraron el desarrollo de IPython fue contar con una herramienta que uniera la posibilidad de realizar cálculos avanzados con la obtención de representaciones gráficas de los propios datos que facilitaran su análisis.

IPython utiliza **Pylab** para realizar los gráficos. **Pylab** es una API para Python de la biblioteca gráfica **Matplotlib** (matplotlib.pyplot) que utiliza **Numpy**, un módulo matemático que añade funciones para operar con vectores o matrices.

Para trabajar con representaciones gráficas de datos tenemos varias posibilidades con IPython:

Iniciar la consola interactiva y el interfaz gráfico orientado a objetos:

$ ipython3 --pylab

Iniciar la consola interactiva QT y el interfaz gráfico oriento a objetos:

$ ipython3 qtconsole --pylab

Iniciar Jupyter Notebook y el interfaz gráfico orientado a objetos:

$ jupyter notebook

En un cuaderno utilizar las funciones `%pylab` o `%matplotlib`.

Cuando Notebook formaba parte de Ipython:

$ ipython3 notebook

A continuación, se muestran una serie de ejemplos que pretenden ilustrar la forma de trabajar con gráficos en IPython y Jupyter.

En primer lugar iniciaremos la consola con:

$ ipython3 --pylab

El argumento **--pylab** hace que al iniciar la sesión se carguen los módulos **matplotlib.pylab** y **NumPy** con los alias **mpl** y **np**, respetivamente. Para obtener ayuda de estos módulos en IPython:

: numpy? ó numpy??
: pylab? ó pylab??

Dibujar un gráfico de líneas

El siguiente ejemplo crea un gráfico a partir de una lista de valores. En el eje "x" se representan 8 elementos numerados del 0 al 7 y en el eje "y" los valores de cada uno de ellos siguiendo el

Buscar

Buscar

Python para impacientes

Python
IPython
EasyGUI
Tkinter
JupyterLab
Numpy

Anexos

Guía urgente de MySQL
Guía rápida de SQLite3

Entradas + populares

Dar color a las salidas en la consola
En Python para dar color a las salidas en la consola (o en la terminal de texto) existen varias posibilidades. Hay un método basado ...

Instalación de Python, paso a paso
Instalación de Python 3.6 A finales de 2016 se produjo el lanzamiento de Python 3.6 . El propósito de esta entrada es mostrar, pas...

Variables de control en Tkinter
Variables de control Las variables de control son objetos especiales que se asocian a los widgets para almacenar sus valore...

Añadir, consultar, modificar y suprimir elementos en Numpy
Acceder a los elementos de un array. [], [,], ... Acceder a un elemento de un array. Para acceder a un elemento se utiliz...

Cálculo con arrays Numpy
Numpy ofrece todo lo necesario para obtener un buen rendimiento cuando se trata de hacer cálculos con arrays. Por como está concebid...

Operaciones con fechas y horas. Calendarios
Los módulos datetime y calendar amplían las posibilidades del módulo time que provee funciones para manipular expresiones de ti...

Tkinter: interfaces gráficas en Python
Introducción Con Python hay muchas posibilidades para programar una interfaz gráfica de usuario (GUI) pero Tkinter es fácil d...

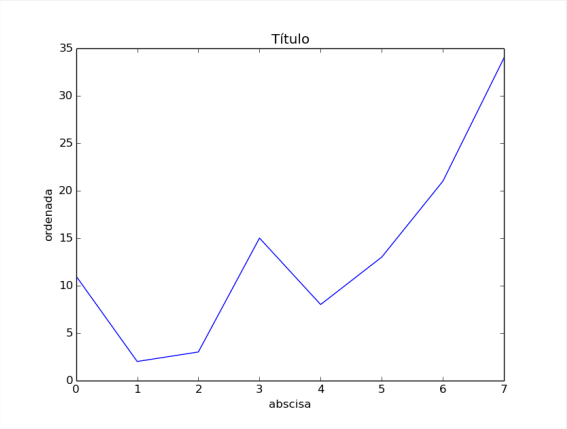
Convertir, copiar, ordenar, unir y dividir arrays Numpy
Esta entrada trata sobre algunos métodos que se utilizan en Numpy para convertir listas en arrays y viceversa; para copiar arrays d...

Tkinter: Tipos de ventanas
Ventanas de aplicación y de diálogos En la entrada anterior tratamos los distintos gestores de geometría que se utilizan para di...

El módulo random
El módulo random de la librería estándar de Python incluye un conjunto de funciones

orden de la lista.

```
: lista1 = [11,2,3,15,8,13,21,34]  # Declara lista1 con 8 valores
: plt.plot(lista1)  # Dibuja el gráfico
: plt.title("Título")  # Establece el título del gráfico
: plt.xlabel("abscisa")  # Establece el título del eje x
: plt.ylabel("ordenada")  # Establece el título del eje y
```



Este tipo de gráficos pueden crearse también con el intérprete interactivo de Python si se importan los módulos adecuados. Hay dos modos de hacerlo:

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> lista1 = [11,2,3,15,8,13,21,34]
>>> plt.plot(lista1)
>>> plt.show()
```

o bien,

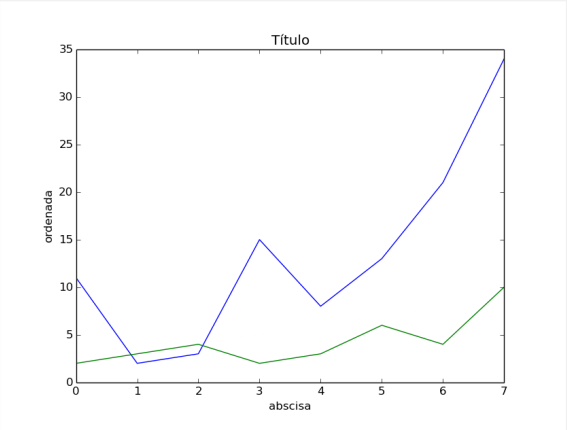
```
>>> from pylab import *
>>> lista1 = [11,2,3,15,8,13,21,34]
>>> plt.plot(lista1)
>>> plt.show()
```

En el ejemplo anterior aunque se importa el módulo **Numpy** no es necesario porque no se utiliza. Se incluye la línea porque dicho módulo suele usarse con frecuencia cuando se trabaja con gráficos.

Activar y desactivar el modo interactivo de dibujo

A continuación, se utilizan los métodos **ion()** y **ioff()** que activan o desactivan el modo interactivo de dibujo para añadir, o no, nuevos datos sobre el gráfico actual.

```
: lista1 = [11,2,3,15,8,13,21,34]  # Declara lista1 con 8 valores
: plt.plot(lista1)  # Dibuja el gráfico
: plt.xlabel("abscisa")  # Inserta el título del eje X
: plt.ylabel("ordenada")  # Inserta el título del eje Y
: plt.ioff()  # Desactiva modo interactivo de dibujo
: lista2 = [2,3,4,2,3,6,4,10]  # Declara lista2 con 8 valores
: plt.plot(lista2)  # No dibuja datos de lista2
: plt.ion()  # Activa modo interactivo de dibujo
: plt.plot(lista2)  # Dibuja datos de lista2 sin borrar datos de lista1
```



Para conocer en un momento dado qué modo está activo:

que permiten obtener de distintos modos números a...

Archivo

agosto 2014 (15)

python.org

pypi.org

Sitios

- ActivePython
- Anaconda
- Bpython
- Django
- Flask
- Ipython
- IronPython
- Matplotlib
- MicroPython
- Numpy
- Pandas
- Pillow
- PortablePython
- PyBrain
- PyCharm
- PyDev
- PyGame
- Pypi
- PyPy
- Pyramid
- Python.org
- PyTorch
- SciPy.org
- Spyder
- Tensorflow
- TurboGears

: plt.isinteractive() # La función devolverá True o False

Los métodos **show()** o **draw()** fuerzan que la información se muestre en el gráfico (datos, títulos, etiquetas, etc.) aunque el modo interactivo esté desactivado:

: plt.ioff() # Desactiva modo interactivo
: lista3 = [9,15,9,15,9,15,9,15] # Declara lista3 con 8 valores
: plt.plot(lista3) # No dibuja datos de lista3
: plt.show() # Fuerza dibujo de datos de lista3
: plt.title("Gráfica") # Establece nuevo título pero no muestra en gráfico
: plt.show() # Actualiza gráfico con nuevo título
: plt.grid(True) # Activa cuadrícula del gráfico pero no se muestra
: plt.show() # Muestra cuadrícula del gráfico
: plt.ion() # Activa modo interactivo de dibujo

Añadir leyendas a un gráfico

Para añadir leyendas al gráfico anterior asignar al parámetro "**label=**" de **plot()** el literal de la leyenda a mostrar. Y después, ejecutar el método **legend()**.

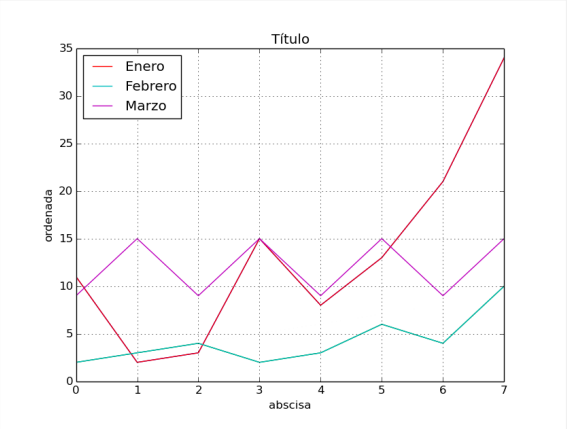
: plt.plot(lista1, label = "Enero")
: plt.plot(lista2, label = "Febrero")
: plt.plot(lista3, label = "Marzo")
: plt.legend()

Posición de leyendas (loc=):

- upper, arriba
- lower, abajo
- center, centro
- left, izquierda y
- right, derecha

Para colocar la leyenda en la parte superior del gráfico y a la izquierda:

: plt.legend(loc="upper left")



Cambiar estilos de línea, marcadores y colores de un gráfico

Para dar claridad a los gráficos pueden establecerse distintos estilos de líneas, marcadores y colores:

Estilos de Líneas (linestyle=):

- -, Línea Sólida
- --, Línea discontinua
- :, Línea punteada
- -., Línea punteada discontinua. y
- None, Ninguna línea

Marcadores (marker=):

- +, Cruz
- ., Punto
- o,Círculo
- *, Estrellas
- p, Pentágonos
- s, cuadrados
- x, Tachados

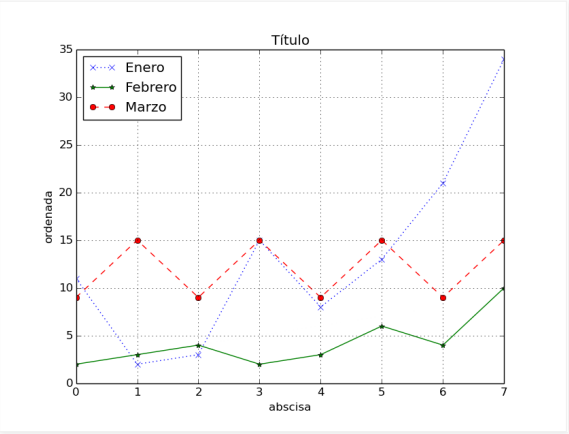
- D, Diamantes
- h, Hexágonos y
- ^, Triángulos

Colores (color=):

- b, blue
- g, green
- r, red
- c, cyan
- m, magenta
- y, yellow
- k, black
- w, white

Ejemplo:

```
: plt.plot(lista1, marker='x', linestyle=':', color='b', label = "Enero")
: plt.plot(lista2, marker='*', linestyle='-', color='g', label = "Febrero")
: plt.plot(lista3, marker='o', linestyle='--', color='r', label = "Marzo")
: plt.legend(loc="upper left")
```



También, es posible cambiar las fuentes de los textos, los grosores de las líneas y otras propiedades que modifican el aspecto de un gráfico.

Añadir rótulos a los ejes

En el siguiente ejemplo se crea un gráfico nuevo con el método **figure()** donde se representa sólo una lista de valores y se definen los rótulos de los ejes.

```
: plt.figure()  # Comenzamos un nuevo gráfico (figura)
: lista1 = [11,2,3,15,8,13,21,34]
: plt.title("Título")
: plt.xlabel("abscisa")
: plt.ylabel("ordenada")
: indice = np.arange(8)   # Declara un array
: plt.xticks(indice, ("A", "B", "C", "D", "E", "F", "G", "H"))
: plt.yticks(np.arange(0,51,10))
: plt.plot(lista1)
```

Se han generado dos arrays con la funcion **arange()** del módulo **Numpy** con los siguientes valores:

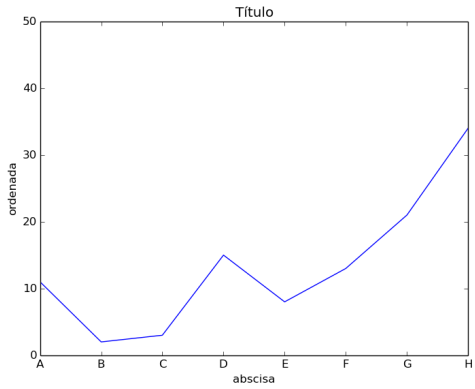
- El array que se obtiene con np.arange(8) contiene los siguiente valores:

array([0, 1, 2, 3, 4, 5, 6, 7])

- El array que se obtiene con np.arange(0,51,10) contiene los siguiente valores:

array([0, 10, 20, 30, 40, 50])

Los arrays se utilizan para definir los rótulos que se mostrarán en ambos ejes. En el eje "x" el valor del índice es sustituido por un carácter alfabético y en el eje "y", directamente, se muestran en la escala los valores del propio array.



Trabajar con varios gráficos

- Activar gráfico por su número:

Si estamos trabajando con varios gráficos a la vez para activar uno concreto indicaremos el número que aparece en su barra de título. Cualquier acción que se realice a continuación se hará sobre dicho gráfico: **plt.figure(*Numero*)**

Ejemplo:

```
: plt.figure()   #  Añade un nuevo gráfico y lo activa
: x = linspace(2,8,4)   # devuelve -> array([ 2.,  4.,  6.,  8.])
: y = x ** 2
: plt.plot(x, y, "r")
: plt.xlabel("x")
: plt.ylabel("y")
: plt.title("Grafico")
: plt.figure(1)   # Activa el gráfico 1
: plt.title("Título")   # Cambia el título del gráfico 1
```

- Activar gráfico por su nombre:

Cuando se trabaja con varios gráficos, otra posibilidad de activar un gráfico determinado es referirse a su nombre:

```
: plt.figure('Regiones')
: plt.figure('Dispersión')
: elementosx = np.random.rand(10)   # Genera array 10 elementos eje x
: elementosy = np.random.rand(10)   # Genera array 10 elementos eje y
: plt.scatter(elementosx, elementosy)
: plt.figure('Regiones')
: plt.plot(elementosx,elementosy)
```

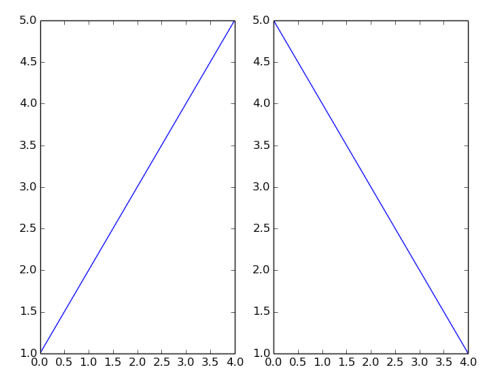
- Generar dos gráficos en la misma ventana:

Dividir la ventana en 1 fila x 2 columnas y dibujar primer gráfico

```
: plt.subplot(1,2,1)
: plt.plot((1,2,3,4,5))
```

Dividir la ventana en 1 fila x 2 columnas y dibujar segundo gráfico

```
: plt.subplot(1,2,2)
: plt.plot((5,4,3,2,1))
```



Continuar "dibujando" o reiniciar el gráfico

Con el método **hold()** se conmutan dos estados: uno para continuar "dibujando" (***True***) sobre un gráfico y otro para reiniciar el gráfico (***False***).

: plt.hold() # Conmutar de *True* a *False* y de *False* a *True*

Para comprobar el estado en un momento dado:

: plt.ishold() # devuelve -> True / False

Borrar un gráfico. Borrar los ejes. Cerrar un gráfico

El método **cla()** borra toda la información relacionada con los ejes de un gráfico y el método **clf()** borra todo el gráfico. Por otro lado, **close()** termina el gráfico cerrando su ventana.

: plt.cla() # Borrar información de los ejes
: plt.clf() # Borrar un gráfico completo
: plt.close() # Terminar un gráfico

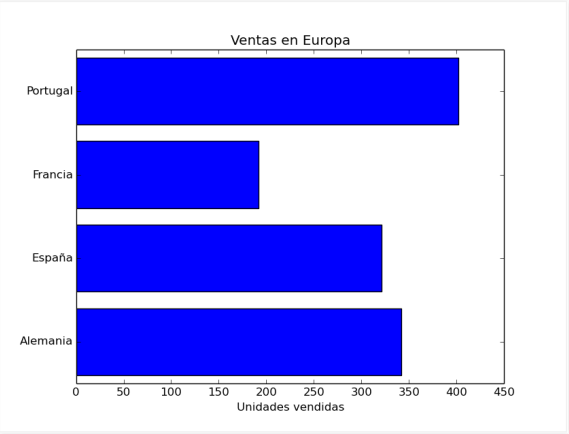
Guardar un gráfico como .png/.pdf

Para guardar la imagen de un gráfico en un archivo:

: savefig("archivo.png") # Guardar en formato .png
: savefig("archivo.pdf") # Guardar en formato .pdf

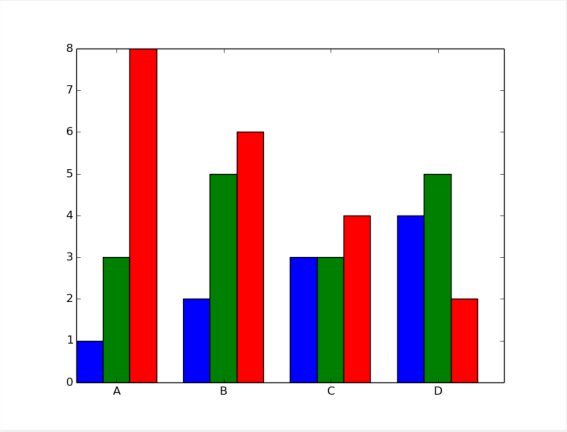
Dibujar un gráfico de barras horizontales

```
: paises = ("Alemania", "España", "Francia", "Portugal")
: posicion_y = np.arange(len(paises))
: unidades = (342, 321, 192, 402)
: plt.barh(posicion_y, unidades, align = "center")
: plt.yticks(posicion_y, paises)
: plt.xlabel('Unidades vendidas')
: plt.title("Ventas en Europa")
```



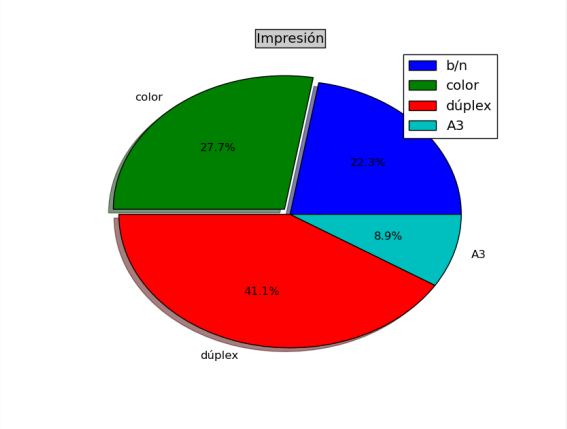
Dibujar un gráfico de mútiples barras verticales

```
: datos = [[1, 2, 3, 4], [3, 5, 3, 5], [8, 6, 4, 2]]
: X = np.arange(4)
: plt.bar(X + 0.00, datos[0], color = "b", width = 0.25)
: plt.bar(X + 0.25, datos[1], color = "g", width = 0.25)
: plt.bar(X + 0.50, datos[2], color = "r", width = 0.25)
: plt.xticks(X+0.38, ["A","B","C","D"])
```



Dibujar un gráfico circular

```
: impr = ["b/n", "color", "dúplex", "A3"]
: vol = [25, 31, 46, 10]
: expl =(0, 0.05, 0, 0)
: pie(vol, explode=expl, labels=impr, autopct='%1.1f%%', shadow=True)
: plt.title("Impresión", bbox={"facecolor":"0.8", "pad":5})
: plt.legend()
```



Relacionado:

- Gráficos con GooPyCharts

Ir al índice del tutorial de IPython

Publicado por Pherkad en 16:10

Etiquetas: IPython, Jupyter

Entrada más reciente Inicio Entrada antigua