



UNIVERSIDAD  
DE MÁLAGA



CURSOS ONLINE

FUNDACIÓN GENERAL UNIVERSIDAD DE MÁLAGA

# Visualización de Datos

## Unidad 12

Python para principiantes  
[dgm@uma.es](mailto:dgm@uma.es)



# Visualización de Datos

- Una de las características por la que Python se ha hecho tan popular en ámbitos muy diversos es por su facilidad para **recopilar, analizar y representar datos**
- En esta presentación veremos un par de ejemplos sobre cómo **recuperar información de Internet**, procesarla y crear gráficas que aporten **algo más de valor** sobre los datos originales
- También veremos cómo podemos instalar librerías externas usando el comando **pip** y utilizaremos las librerías **requests** y **matplotlib**
  - La librería requests permite acceder a recursos de Internet
  - La librería matplotlib se utiliza para visualización de datos

# Requisitos Previos

- En primer lugar, vamos a comprobar si tenemos las librerías que necesitamos instaladas. Para ello, abre la consola Python y escribe lo siguiente:

```
>>> import requests
```

Y a continuación

```
>>> import pylab
```

- Si no están instaladas (y en una instalación limpia de Python no lo están), verás errores como los siguientes:

```
>>> import requests
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ModuleNotFoundError: No module named 'requests'

```
>>> import pylab
```

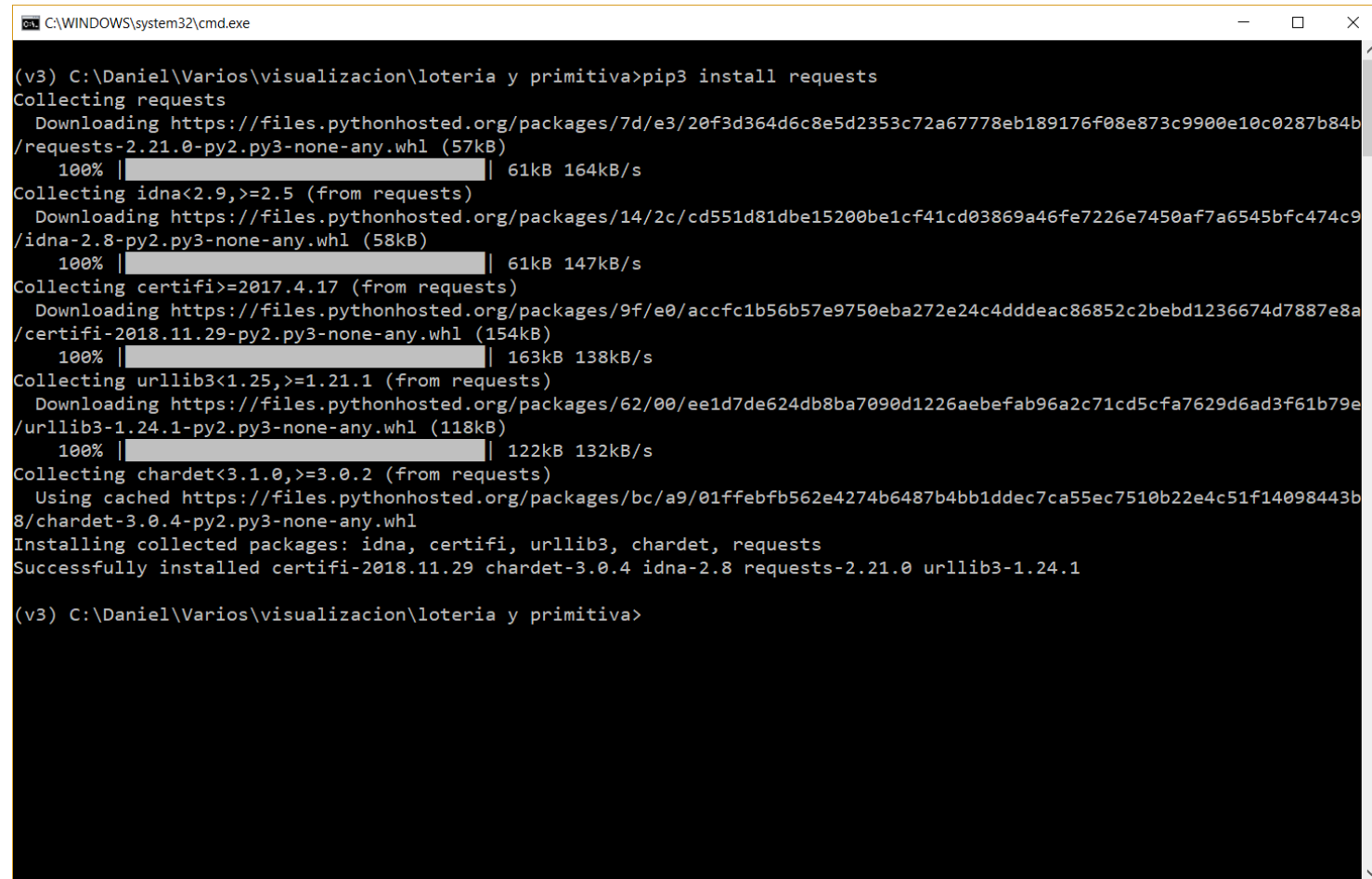
Traceback (most recent call last):

File "<stdin>", line 1, in <module>

ModuleNotFoundError: No module named 'pylab'

# Utilización de pip

- El comando pip permite **instalar librerías** de terceros en tu instalación python.
- Para instalar las librerías que necesitamos, ejecuta sobre tu terminal (**del Sistema Operativo**, no de Python!) lo siguiente:
  - **pip3 install requests**
  - **pip3 install matplotlib**
- Si todo va bien, deberías ver algo similar a la figura de la derecha
- pip3 es para Python 3. También deberías saber que hay otras alternativas más limpias (**virtualenv**), aunque no vamos a usarlas en estos ejemplos



```
C:\WINDOWS\system32\cmd.exe

(v3) C:\Daniel\Varios\visualizacion\loteria y primitiva>pip3 install requests
Collecting requests
  Downloading https://files.pythonhosted.org/packages/7d/e3/20f3d364d6c8e5d2353c72a67778eb189176f08e873c9900e10c0287b84b
/requests-2.21.0-py2.py3-none-any.whl (57kB)
    100% |#####| 61kB 164kB/s
Collecting idna<2.9,>=2.5 (from requests)
  Downloading https://files.pythonhosted.org/packages/14/2c/cd551d81dbe15200be1cf41cd03869a46fe7226e7450af7a6545bfc474c9
/idna-2.8-py2.py3-none-any.whl (58kB)
    100% |#####| 61kB 147kB/s
Collecting certifi>=2017.4.17 (from requests)
  Downloading https://files.pythonhosted.org/packages/9f/e0/accfc1b56b57e9750eba272e24c4dddeac86852c2bebd1236674d7887e8a
/certifi-2018.11.29-py2.py3-none-any.whl (154kB)
    100% |#####| 163kB 138kB/s
Collecting urllib3<1.25,>=1.21.1 (from requests)
  Downloading https://files.pythonhosted.org/packages/62/00/ee1d7de624db8ba7090d1226aebefab96a2c71cd5cfa7629d6ad3f61b79e
/urllib3-1.24.1-py2.py3-none-any.whl (118kB)
    100% |#####| 122kB 132kB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests)
  Using cached https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b
8/chardet-3.0.4-py2.py3-none-any.whl
Installing collected packages: idna, certifi, urllib3, chardet, requests
Successfully installed certifi-2018.11.29 chardet-3.0.4 idna-2.8 requests-2.21.0 urllib3-1.24.1

(v3) C:\Daniel\Varios\visualizacion\loteria y primitiva>
```

# Caso de uso 1: Evolución Euribor

- En este primer ejemplo, vamos a ver cómo podemos utilizar Python para estudiar la evolución del Euribor a lo largo del tiempo. Para ello vamos a utilizar datos originalmente disponibles de la siguiente web:

- <https://www.emmi-benchmarks.eu/euribor-org/euribor-rates.html>

- Tendrás el código definitivo al final**



The screenshot displays the EMMI website's 'Euribor® Rates' page. At the top, there is a navigation bar with the EMMI logo and a menu containing 'EMMI', 'Euribor', 'Eonia', and 'Step'. Below the navigation bar, a sidebar on the left lists various links: 'Back To EMMI', 'Euribor®', 'History', 'Euribor Reform', 'Rates' (which is highlighted), 'Re-Fixing', 'Panel Banks', and 'Steering Committee'. The main content area features the heading 'Euribor® Rates' and a teal box stating: 'In order to be compliant with new web security standards EMMI and STEP websites are now in HTTPS protocol. Should you have any issue accessing the sites or the data please do not hesitate to contact us'. Below this, an orange box notes: 'The EMMI website only displays rates on a 24-hour delayed basis.' A paragraph follows, explaining that Euribor is calculated daily based on the 'Euribor Code of Conduct' and requires panel banks. On the right side of the page, there is a search bar, a 'what's new' section with a link to a 'Second Public Consultation on Hybrid Methodology for Euribor' from October 2018, and two promotional banners for 'Euribor Reform' and 'EMMI Workshops'.

# Caso de uso 1: Evolución Euribor

- Actualización 2021: Tras los últimos cambios de la web anterior. Los datos, no pueden descargarse directamente en formato CSV, por lo que usaremos un repositorio externo (la URL se indicará a continuación)
- En nuestro caso **descargaremos los datos directamente** desde nuestro programa Python
  - Esto tiene la ventaja de que si hubiera cambios (en algún otro ejemplo), no tenemos que volver a descargar los archivos

# Caso de uso 1: Evolución Euribor

- En la dirección [https://github.com/dgarridouma/python-course/blob/main/EURIBOR\\_2017.csv](https://github.com/dgarridouma/python-course/blob/main/EURIBOR_2017.csv) podéis encontrar los datos relativos a 2017:

,Jan17,Feb17,Mar17,Apr17,May17,Jun17,Jul17,Aug17,Sep17,Oct17,Nov17,Dec17,2017  
1w,-0.378,-0.379,-0.379,-0.379,-0.379,-0.379,-0.379,-0.379,-0.379,-0.38,-0.379,-0.377,-0.379  
2w,-0.373,-0.372,-0.372,-0.372,-0.373,-0.373,-0.376,-0.376,-0.377,-0.377,-0.376,-0.372,-0.374  
1m,-0.371,-0.372,-0.372,-0.372,-0.373,-0.373,-0.373,-0.372,-0.372,-0.372,-0.372,-0.369,-0.372  
2m,-0.339,-0.341,-0.34,-0.34,-0.341,-0.342,-0.341,-0.34,-0.34,-0.34,-0.341,-0.339,-0.34  
3m,-0.326,-0.329,-0.329,-0.33,-0.329,-0.33,-0.33,-0.329,-0.329,-0.33,-0.329,-0.328,-0.329  
6m,-0.236,-0.241,-0.241,-0.246,-0.251,-0.267,-0.273,-0.272,-0.273,-0.274,-0.274,-0.271,-0.26  
9m,-0.152,-0.165,-0.171,-0.179,-0.179,-0.195,-0.206,-0.211,-0.218,-0.221,-0.219,-0.22,-0.195  
12m,-0.095,-0.106,-0.11,-0.119,-0.127,-0.149,-0.154,-0.156,-0.168,-0.18,-0.189,-0.19,-0.145

# Caso de uso 1: Evolución Euribor

- Los datos que podemos encontrarnos en Internet pueden ser **de muy diferentes** tipos, y en cada uno de ellos tendremos que **afrontarlo de una manera diferente**.
- En muchas ocasiones los datos se encuentran **incompletos, con errores o no se ajustan a nuestras necesidades**, por lo que tendremos que adaptarnos a lo que encontremos.
- En el caso que nos ocupa, encontramos en cada línea (si ignoramos la primera), la evolución del índice Euribor para 1 semana, 2 semanas, 1 mes, 2 meses, 3 meses, 6 meses, 9 meses y un año para cada uno de los meses del año
- Si nos damos cuenta, un archivo csv no es más que un archivo de texto donde la información se encuentra separada (en este caso) por comas ','
  - Por ejemplo, la línea:  
1w,-0.378,-0.379,-0.379,-0.379,-0.379,-0.379,-0.379,-0.379,-0.38,-0.379,-0.377,-0.379  
Nos indica que en enero de 2017 el valor del índice para 1 semana fue -0.378, para febrero -0.379 y así sucesivamente
- Con lo que hemos visto en unidades anteriores, seríamos capaces de procesar esta información para representar en una gráfica de líneas cómo ha ido evolucionando el índice



# Recuperación de la información - Euribor

- Para recuperar la información en nuestro programa vamos a usar la **librería requests**

```
import requests
```

- En la variable url indicamos la dirección donde están nuestros datos. Para recuperarlos, tenemos que hacer una **petición get (protocolo http)**.

```
url = 'https://raw.githubusercontent.com/dgarridouma/python-course/main/EURIBOR_2017.csv'
```

```
response=requests.get(url)
```

- La respuesta queda almacenada en la variable response, y si queremos tener nuestro texto dividido en líneas, podemos hacer lo siguiente para tener una lista de strings!

```
texto=response.text.splitlines()
```

# Procesamiento de la información - Euribor

- Una vez tenemos el texto dividido en líneas, solo tenemos que ir recuperando los valores. Usaremos para ello un **bucle for**, el **método split** y **compresión de listas**!:

```
for linea in texto[1:]: # descartamos la primera línea de cabecera
    palabras_linea=linea.split(',')

    # creamos una lista de flotantes con los valores de la linea
    # descartamos el primer y ultimo valor de la lista por contener
    # el nombre del indicador y la media del año respectivamente
    valores=[float(x) for x in palabras_linea[1:len(palabras_linea)-1]]
```

Si quisiéramos verlos por pantalla, no tendríamos más que hacer:

```
print(valores)
```

Pero hemos dicho que queríamos verlos en una gráfica!

# Visualización de la información - Euribor

- La librería matplotlib, sobre la que puedes encontrar abundante información en <https://matplotlib.org/index.html> es ampliamente utilizada en Python para representar datos
- Es una librería muy extensa. En este caso de uso nos vamos a centrar en lo que nos interesa. No obstante, un primer ejemplo muy básico sería el siguiente:

```
import pylab as pl
```

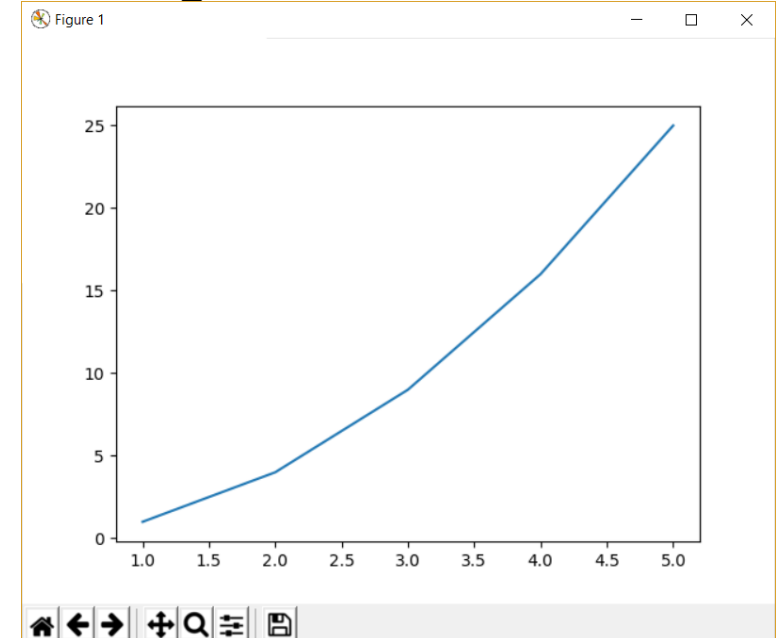
```
x = [1, 2, 3, 4, 5]
```

```
y = [1, 4, 9, 16, 25]
```

```
pl.plot(x, y)
```

```
pl.show()
```

Pruébalo antes en otro programa, y deberías ver algo así:



# Visualización de la información - Euribor

- En el caso que nos ocupa, queremos tener una serie de datos por cada indicador (semanal, mensual, etc.). Para ello bastará con hacer sucesivos "plot" en el bucle for anterior, que quedaría así:

```
for linea in texto[1:]:  
    palabras_linea=linea.split(',')  
    valores=[float(x) for x in  
palabras_linea[1:len(palabras_linea)-1]]  
    pl.plot(range(1,13),valores,  
marker='o',label=palabras_linea[0])
```

Añadir al comienzo  
del programa:  
import pylab as pl

Hemos indicado también que queremos utilizar un marcador ('o') y que queremos poner un nombre a nuestra serie de datos (label)

# Visualización de la información - Euribor

- Nos faltan aún algunos detalles más:

```
pl.title("Evolucion EURIBOR 2017")
```

```
pl.xlabel("Mes")
```

```
pl.ylabel("Valor indicador")
```

```
pl.xticks(range(13),[x[:3] for x in list(calendar.month_name)])
```

```
pl.grid(True)
```

```
pl.legend()
```

Y finalmente:

```
pl.show()
```

Añadir al comienzo  
del programa:  
`import calendar`

Estamos usando compresión de listas y la  
librería calendar para recuperar los  
nombres del mes

# Visualización de la información - Euribor

- Un último detalle. Si habéis probado el programa anterior, los nombres del mes (las 3 primeras letras) ¡salen en inglés!
- Para solucionarlo podemos usar la librería locale y decirle que use los nombres en español:

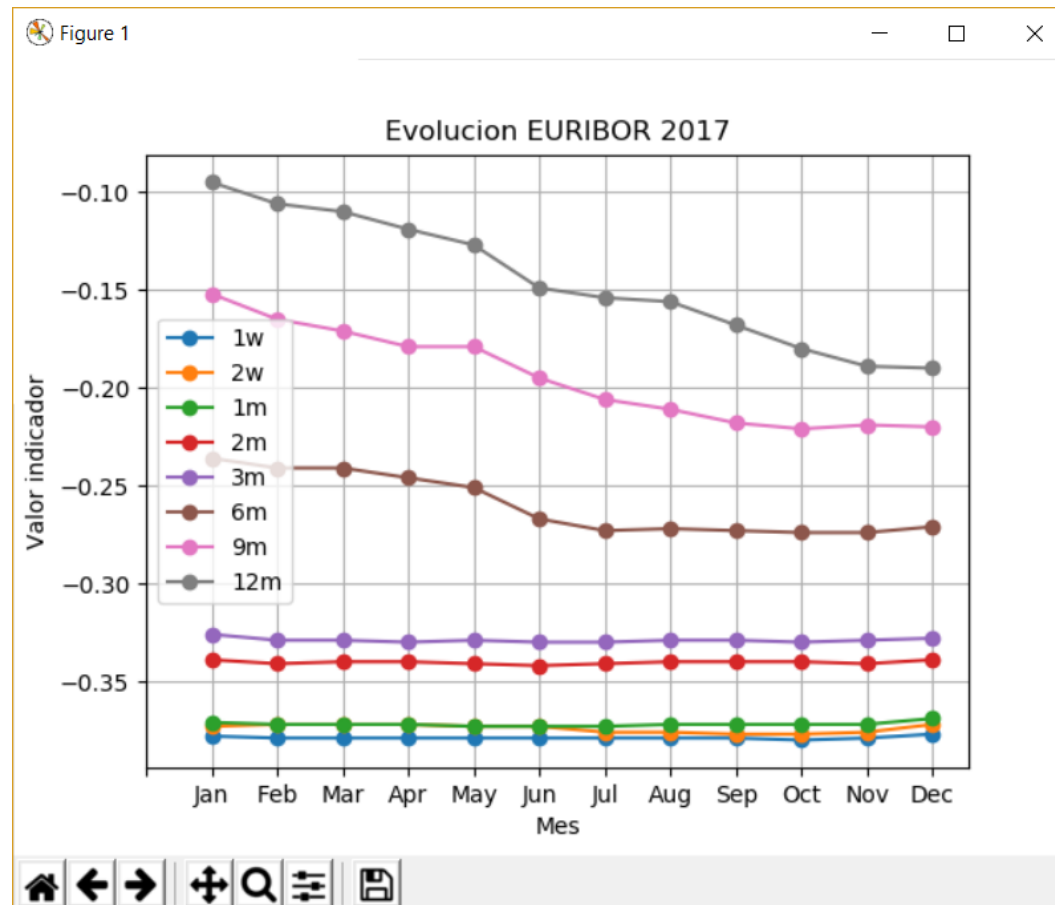
```
import locale
```

```
...
```

```
locale.setlocale(locale.LC_ALL, 'es_ES')
```

# Visualización de la información - Euribor

- Si todo ha ido bien, al ejecutar el programa deberías obtener lo siguiente:



# Código definitivo - Euribor

```
import pylab as pl
import requests
import calendar
import locale

locale.setlocale(locale.LC_ALL,'es_ES') # para obtener nombres meses
url = 'https://raw.githubusercontent.com/dgarridouma/python-course/main/EURIBOR_2017.csv'

response=requests.get(url)
texto=response.text.splitlines()
for linea in texto[1:]:
    palabras_linea=linea.split(',')

    # creamos una lista de flotantes con los valores de la linea
    # descartamos el primer y ultimo valor de la lista por contener
    # el nombre del indicador y la media del año respectivamente
    valores=[float(x) for x in palabras_linea[1:len(palabras_linea)-1]]
    pl.plot(range(1,13),valores, marker='o',label=palabras_linea[0])
```



# Código definitivo - Euribor

```
... (viene de la diapositiva anterior)
pl.title("Evolucion EURIBOR 2017")
pl.xlabel("Mes")
pl.ylabel("Valor indicador")
pl.xticks(range(13), [x[:3] for x in list(calendar.month_name)])
pl.grid(True)
pl.legend()
pl.show()
```

# Caso de uso 2: Lotería primitiva

- En este segundo ejemplo, vamos a ver algo más lúdico (lotería primitiva ;-), analizando qué números salen con más frecuencia en los sorteos
- ¡Posiblemente no nos vamos a hacer ricos!, pero al menos veremos cómo hacer **gráficas de barras**
- En esta ocasión, hemos obtenido los datos de la siguiente dirección:

<http://www.lotoideas.com/primitiva-resultados-historicos-de-todos-los-sorteos/>



Histórico de los resultados de La Primitiva

Actualizado frecuentemente

Resultados históricos de todos los sorteos de La Primitiva desde su inicio.  
Totalmente gratis y listos para su descarga en varios formatos.

Lotoideas.com - Histórico de Resultados - Primitiva

| FECHA      | COMBINACIÓN GANADORA |    |    |    |    |    | COMP. | R. | JOKER   |
|------------|----------------------|----|----|----|----|----|-------|----|---------|
| 22/12/2018 | 08                   | 11 | 32 | 41 | 44 | 45 | 38    | 3  | 1092798 |
| 20/12/2018 | 08                   | 24 | 25 | 31 | 43 | 46 | 12    | 8  | 7952896 |
| 15/12/2018 | 06                   | 09 | 10 | 24 | 26 | 32 | 25    | 5  | 3459017 |
| 13/12/2018 | 02                   | 10 | 41 | 42 | 44 | 45 | 33    | 3  | 3661223 |
| 8/12/2018  | 08                   | 09 | 12 | 19 | 24 | 41 | 39    | 4  | 9759406 |
| 6/12/2018  | 19                   | 20 | 32 | 37 | 42 | 49 | 46    | 0  | 6931344 |
| 1/12/2018  | 02                   | 12 | 19 | 39 | 44 | 45 | 22    | 9  | 1687972 |
| 29/11/2018 | 08                   | 18 | 26 | 40 | 44 | 49 | 04    | 7  | 3594429 |
| 24/11/2018 | 02                   | 10 | 14 | 19 | 24 | 35 | 32    | 9  | 7216216 |
| 22/11/2018 | 09                   | 22 | 39 | 43 | 45 | 47 | 04    | 4  | 0336382 |
| 17/11/2018 | 02                   | 14 | 23 | 29 | 30 | 39 | 37    | 2  | 5045646 |
| 15/11/2018 | 06                   | 26 | 36 | 39 | 40 | 45 | 12    | 8  | 5892581 |

# Formato de los datos - Primitiva

- En esta ocasión, los datos son suministrados nuevamente como archivos csv pero en dos diferentes, con datos desde 1985 a 2012 y desde 2013 al 2018 (o posteriores según cuándo hagas el curso)
- A continuación, veamos un fragmento de estos archivos:

FECHA,COMBINACIÓN GANADORA,,,,,COMP.,R.,JOKER

8/12/2018,08,09,12,19,24,41,39,4,9759406

6/12/2018,19,20,32,37,42,49,46,0,6931344

1/12/2018,02,12,19,39,44,45,22,9,1687972

Los datos vienen ordenados por fecha, a continuación la combinación ganadora (6 números) y luego el complementario, reintegro y joker. Nos vamos a centrar únicamente en los **6 números de la combinación ganadora**

# Imports y obtención de los datos - Primitiva

- Nuevamente usaremos requests y pylab:

```
import requests
import pylab as pl
```

- Como hemos visto, los datos están en 2 archivos diferentes, así que tendremos que hacer 2 peticiones get y unir el resultando:

```
url =
'https://docs.google.com/spreadsheet/pub?key=0AhqMeY8ZOrNKdFJnYkMzX3JzN2lkRl
ZZTXV2aldqS1E&single=true&gid=1&output=csv'
response=requests.get(url)
lineas=response.text.splitlines()
```

```
url='https://docs.google.com/spreadsheet/pub?key=0AhqMeY8ZOrNKdFJnYkMzX3JzN2
lkRlZZTXV2aldqS1E&single=true&gid=0&output=csv'
response=requests.get(url)
lineas=lineas+response.text.splitlines()
```

# Procesamiento de los datos - Primitiva

- El procesamiento de los datos es similar a ejemplos realizados en unidades anteriores. Vamos a tener un **diccionario** donde iremos contabilizando **cuántas veces aparece cada número** en la combinación ganadora

```
for l in lineas[1:]:  
    lsorteo=l.split(',')  
    for num in lsorteo[1:6]:  
        sdict[num]=sdict.get(num,0)+1
```

sdict contiene el número de repeticiones para cada número

# Procesamiento de los datos - Primitiva

- Llegados a este punto, ya hemos construido el diccionario, pero en realidad, queremos tener "algo" ordenado por el número de apariciones de cada número. Vamos a utilizar **compresión de listas** y **sorted** para hacer esto.

```
tmp = list()
for k, v in sdict.items() :
    tmp.append( (v, k) )

top6=sorted(tmp, reverse=True) [:6]
print(top6)
```

Creamos una lista auxiliar con **tuplas** con el valor (el número de repeticiones) como primer componente

top6 almacena los 6 números que han aparecido más veces en los sorteos

# Procesamiento de los datos - Primitiva

- A continuación, preparamos los datos para la gráfica: etiquetas de las barras y valores de las repeticiones usando las tuplas anteriores:

```
top6_nums=[x[1] for x in top6]  
print(top6_nums)
```

```
top6_repeticiones=[int(x[0]) for x in top6]  
print(top6_repeticiones)
```

# Procesamiento de los datos - Primitiva

- Finalmente, creamos la gráfica y la mostramos:

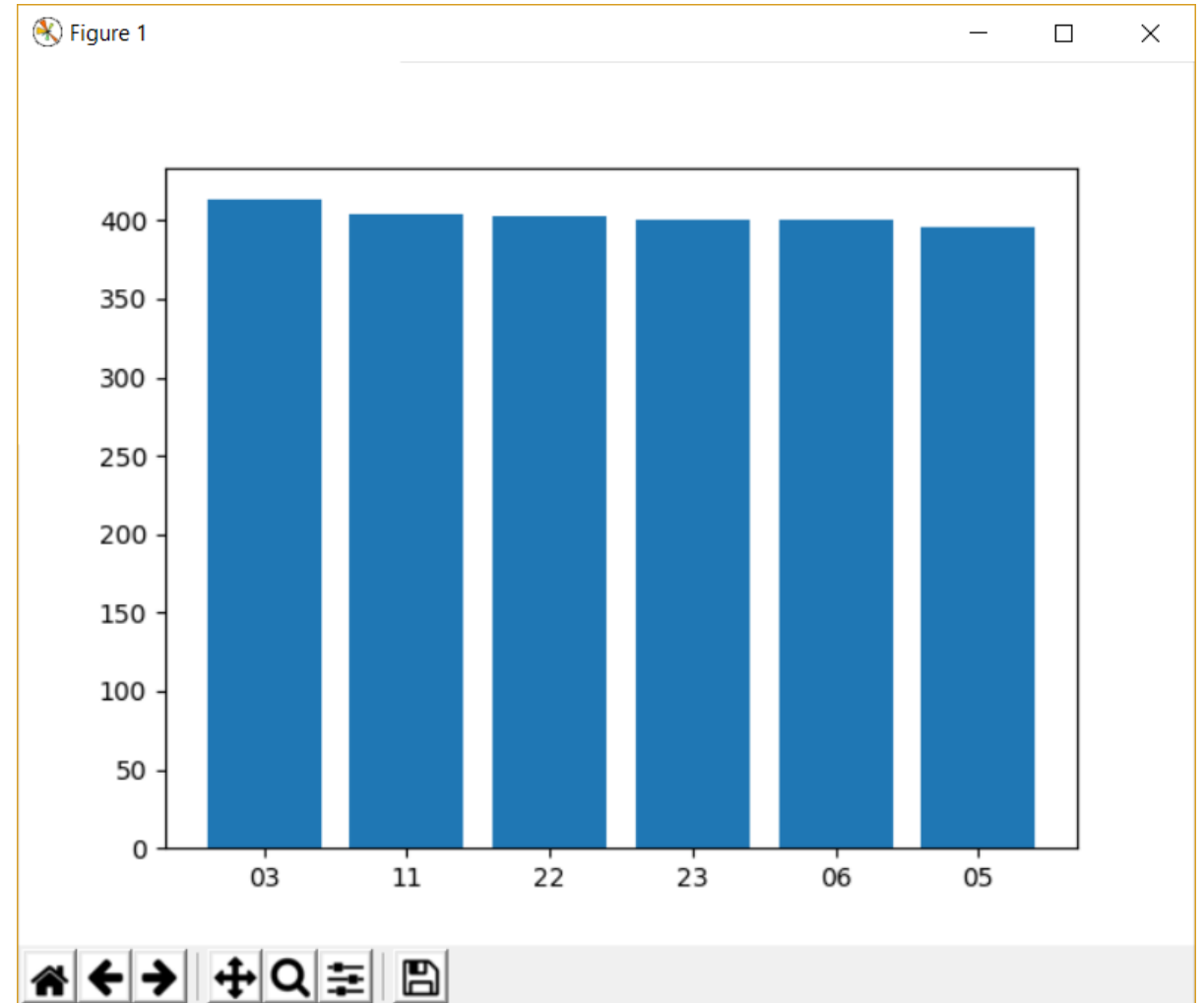
```
pl.bar(range(6), top6_repeticiones,  
align='center')
```

```
puntos=range(6)
```

```
pl.xticks(puntos, top6_nums)
```

```
pl.show()
```

A la derecha podemos ver los 6 números que más veces aparecen, así como el número de repeticiones





# Código definitivo – Lotería Primitiva

```
import requests
import pylab as pl

url =
'https://docs.google.com/spreadsheet/pub?key=0AhqMeY8ZOrNKdFJnYkMzX3JzN2lkRlZZTXV2aldqS1E&
single=true&gid=1&output=csv'
response=requests.get(url)
lineas=response.text.splitlines()

url='https://docs.google.com/spreadsheet/pub?key=0AhqMeY8ZOrNKdFJnYkMzX3JzN2lkRlZZTXV2aldq
S1E&single=true&gid=0&output=csv'
response=requests.get(url)
lineas=lineas+response.text.splitlines()

sdict=dict()
for l in lineas[1:]:
    lsorteo=l.split(',')
    for num in lsorteo[1:6]:
        sdict[num]=sdict.get(num,0)+1
```

# Código definitivo – Lotería Primitiva

```
... (viene de la diapositiva anterior)
tmp = list()
for k, v in sdict.items() :
    tmp.append( (v, k) )

top6=sorted(tmp,reverse=True)[:6]
print(top6)

top6_nums=[x[1] for x in top6]
print(top6_nums)

top6_repeticiones=[int(x[0]) for x in top6]
print(top6_repeticiones)

pl.bar(range(6),top6_repeticiones,align='center')
puntos=range(6)
pl.xticks(puntos,top6_nums)
pl.show()
```

# Resumen

- Instalación de librerías (pip)
- Tratamiento de datos de Internet
- Librería requests
- Peticiones get
- Archivos csv
- Matplotlib
- Gráficas de líneas y barras

# Acknowledgements / Contributions



These slides are Copyright 2018- Daniel Garrido of the University of Málaga and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Continue...

Initial Development: Daniel Garrido, University of Málaga

... Insert new Contributors and Translators here