

Introducción a Python (II)

DOCTOR FUN

6 Apr 2000



Copyright © 2000 David Farley, d-farley@metablab.unc.edu
<http://metablab.unc.edu/Dave/drfun.html>

This cartoon is made available on the Internet for personal viewing only. Opinions expressed herein are solely those of the author.

ORGANIZACIÓN

SESIÓN I

- Introducción a Python
- Tipos de Datos
- Estructuras de control
- Funciones
- Ficheros

SESIÓN II

- Módulo, paquetes y gestores de librerías
- Expresiones regulares
- Entornos virtuales
- Documentación

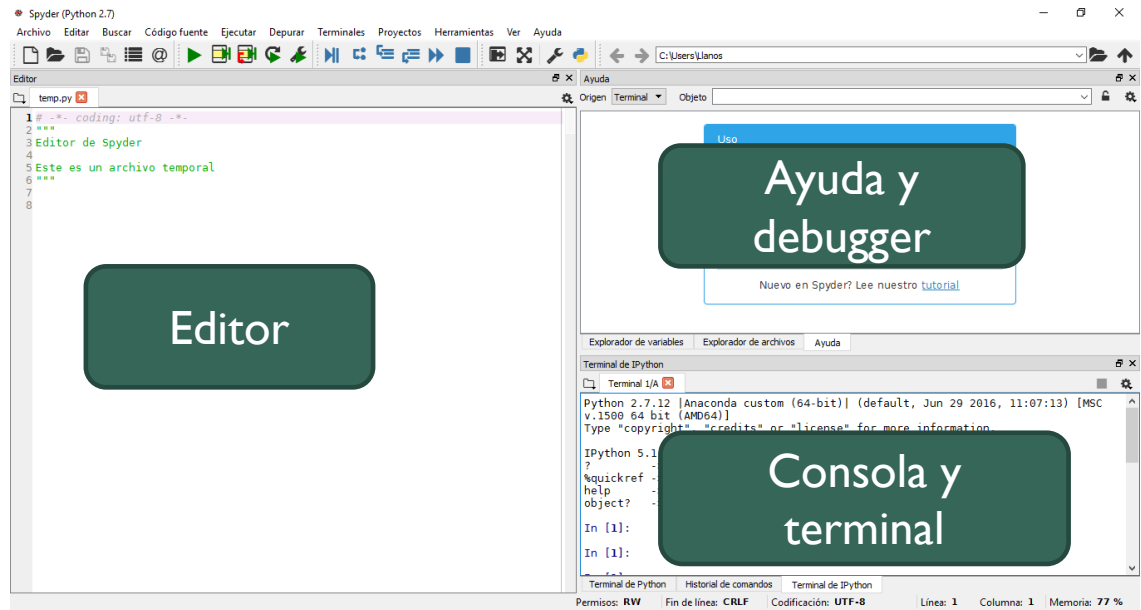
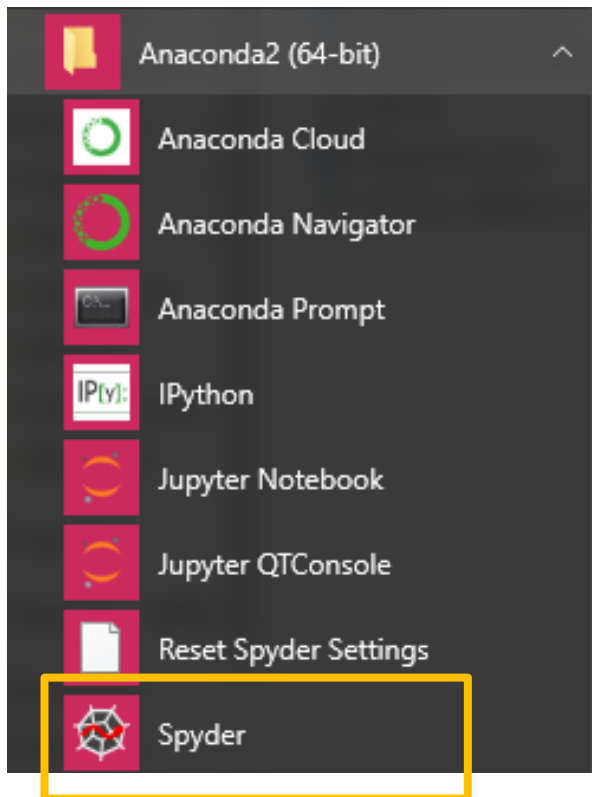
```
#!/usr/bin/python

import sys

for i in range(1,11):
    print i
    if i <= 5:
        print "AHHHHHHH"
    elif i >= 6 and i <= 9:
        print "WHERE ARE ALL THE BRACKETS??"
    else:
        print "HOW DO YOU PEOPLE READ THIS SYNTAX EASILY"
```

SESIÓN II: PRINCIPIOS BÁSICOS

TRABAJO EN SPYDER



MÓDULOS

- Un módulo va a ser un conjunto de funciones y objetos que guardan cierta relación lógica y que agrupamos en un mismo fichero.
 - El nombre del fichero se convierte en el nombre del módulo.
 - Si queremos usar un módulo existente en otro fichero utilizamos la palabra reservada `import` seguido del nombre del módulo.
 - Para distinguir las funciones se llamarán con el nombre del módulo seguido de punto y el nombre de función.

```
#módulo mates.py  
def suma(a,b):  
    return a+b  
  
def resta(a,b):  
    return a-b
```



```
import mates  
  
print mates.suma(5,6)  
print mates.resta(10,2)
```

MÓDULOS

- Podemos importar varios módulos en una misma línea:

```
import módulo1 módulo2 módulo3
```

- En ocasiones no deseamos importar todo el módulo, sino sólo una clase o una función. Ese caso usamos la fórmula `from módulo import función`.
 - En este caso podemos llamar a la función o a la clase directamente sin necesidad de poner el nombre del módulo delante.
- Ejemplo:

```
from random import shuffle  
semaforo=['rojo','amarillo','verde']  
shuffle(semaforo)
```

MÓDULOS

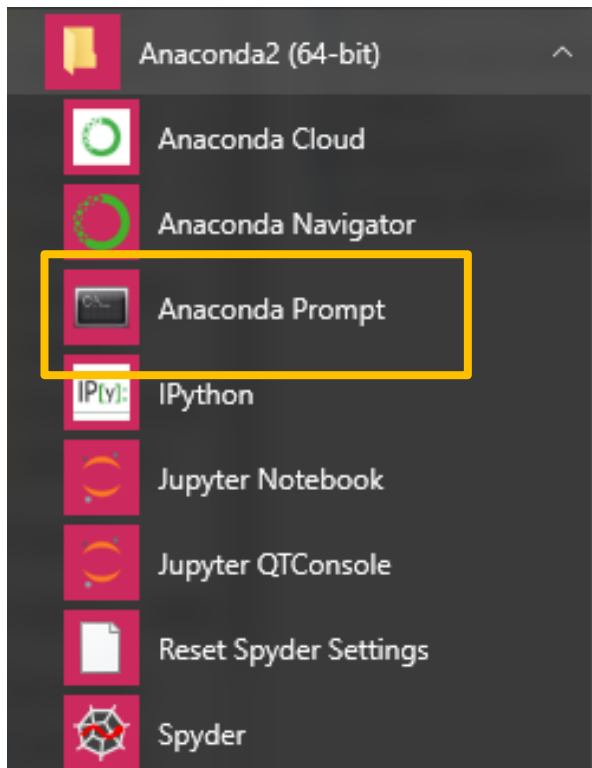
- A la hora de buscar un módulo Python mira en todos los directorios que se definen en la variable de sistema PYTHONPATH además de mirar en el mismo directorio donde se encuentra el fichero .py que incluye la orden import.
- Los módulos también son objetos de Python. De manera que podemos averiguar el nombre de un módulo con el atributo “__name__”.
- Ejemplo:

```
if __name__ == "__main__":  
    print “Este módulo es el principal”
```

PAQUETES

- Un grupo de módulos conforman un paquete.
- Desde el punto de vista práctico un paquete es un directorio que contiene uno o más ficheros .py y un fichero especial llamado `__init__.py`. Este fichero puede estar vacío.
- Con los paquetes también usamos la orden `import`.
`import paquete1.modulo1 paquete2.modulo`
- Cuando queremos importar todos los módulos de un paquete usamos la directiva `from paquete import *`
- Existe un repositorio de paquetes público donde podemos descargarnos paquetes creados por otros usuarios llamado PyPI (<https://pypi.python.org/pypi>). También podemos subir nuestros paquetes.

INSTALAR PAQUETES: PIP



- PIP es una herramienta que va con Python y nos permite descargarnos e instalar librerías de Python.
- Necesitamos conexión a Internet para usarla.
- Es recomendable usarla junto con el Prompt de Anaconda.
- Uso con permisos de administrador.
- Documentación:
<https://pip.pypa.io/en/stable/>

FUNCIONAMIENTO DE PIP

- Instalar un paquetes tanto en PyPI como descargados ya compilados (whl)

`pip install paquete1 paquete2 ...`

- Verificar los archivos instalados

`pip show --files paquete1`

- Actualizar un paquete

`pip install --upgrade paquete`

- Desinstalar un paquete

`pip uninstall paquete`

INSTALAMOS UN PAQUETE

- El paquete requests se utiliza para realizar peticiones a servicios web y servidores HTTP.
- Para instalarlo: `pip install requests`.

```
(env) C:\Users\Llanos\Documents\Python\proyecto>pip install requests
Collecting requests
  Downloading requests-2.18.4-py2.py3-none-any.whl (88kB)
    100% |#####| 92kB 105kB/s
Collecting urllib3<1.23,>=1.21.1 (from requests)
  Downloading urllib3-1.22-py2.py3-none-any.whl (132kB)
    100% |#####| 133kB 152kB/s
Collecting idna<2.7,>=2.5 (from requests)
  Downloading idna-2.6-py2.py3-none-any.whl (56kB)
    100% |#####| 61kB 280kB/s
Collecting chardet<3.1.0,>=3.0.2 (from requests)
  Using cached chardet-3.0.4-py2.py3-none-any.whl
Collecting certifi>=2017.4.17 (from requests)
  Downloading certifi-2017.7.27.1-py2.py3-none-any.whl (349kB)
    100% |#####| 358kB 142kB/s
Installing collected packages: urllib3, idna, chardet, certifi, requests
Successfully installed certifi-2017.7.27.1 chardet-3.0.4 idna-2.6 requests-2.18.4 urllib3-1.22

(env) C:\Users\Llanos\Documents\Python\proyecto>
```

EXPRESIONES REGULARES

- Las expresiones regulares son mecanismos que nos permiten localizar patrones dentro de textos.
- Dentro de Python utilizaremos el paquete `re`.
 - Una búsqueda dentro de Python generalmente sigue la forma:
`coincidencia=re.search(patrón, cadena).`
 - Las coincidencias encontradas almacenadas en el objeto `coincidencia` pueden accederse mediante el método `group()`.

PATRONES BÁSICOS

Patrón	Descripción
.	Periodo, indica cualquier carácter excepto un salto de línea
\w	Carácter alfanumérico en minúsculas incluyendo _
\W	Es cualquier carácter que no sea alfanumérico incluyendo _
\s	Cualquier carácter que represente un espacio en blanco: espacio, salto de línea, tabulador, intro,...
\S	Cualquier carácter que no sea reconocido por \s
\d	Digito desde 0 al 9.
^	Inicio de la cadena
\$	Fin de la cadena
\b	Carácter comprendido entre un carácter \w y uno \W

REPETICIONES Y OPCIONES

- Para indicar que el patrón se repite usamos:
 - + indica que el patrón se repite una o más veces.
 - * indica que el patrón se repite cero o más veces.
- Usaremos [] para indicar que puede aparecer cualquiera de las expresiones regulares indicadas entre corchetes.
 - [ab] indica que puede aparecer una a ó una b.
 - [\d\s] indica que puede aparecer o un dígito o un carácter blanco.
- Re siempre busca el patrón más largo y más a la izquierda.

EJEMPLOS

Patron	Cadena	Salida
r'iii'	"Riiing"	lii
r'pi'	"Riiing"	No encontrado
r'\w\w\w'	I23-ABC-2356aSD	['ABC', 'aSD']
r'\d\w'	I23-ABC-2356aSD	[6a]

GRUPOS

- Dentro de un patrón nos puede interesar extraer varias partes de la expresión. Cada parte la podemos definir como un grupo usando paréntesis.
- Por ejemplo: en el patrón `(\w+[\.\w]+)@(\w[\w.]++)` intenta modelar las direcciones de correo electrónico tendríamos el nombre de usuario como primer grupo y el dominio de correo como segundo grupo.

```
correo=re.search(r' (\w+[\.\w]+)@(\w[\w.]++) ','correo@servidor.es')
```

```
print correo.group() #muestra correo@servidor.es
```

```
print correo.group(1) #muestra correo
```

```
print correo.group(2) #muestra servidor.es
```


FINDALL

- Mientras search busca la cadena más larga y hacia la izquierda, si queremos ver si el patrón se repite más en el documento debemos usar el método findall.

Ejemplo:

```
telefonos=['967 21 34 56 671 78 88 19']  
tlfnos=re.findall(r'\d\d\d \d\d \d\d', Telefonos)  
for t in Tlfnos:  
    print t
```

OTRAS OPCIONES DE RE

- Además podemos usar tanto search como findall con los siguientes modificadores:

Modificador	Descripción
IGNORECASE	No distingue entre mayúsculas o minúsculas en un patrón.
MULTILINE	Busca el patrón en un texto que es más largo de una línea, de forma que podemos usar ^ y \$ para indicar principio y fin de línea.
DOTALL	Permite que el patrón . pueda usarse también con una línea nueva.

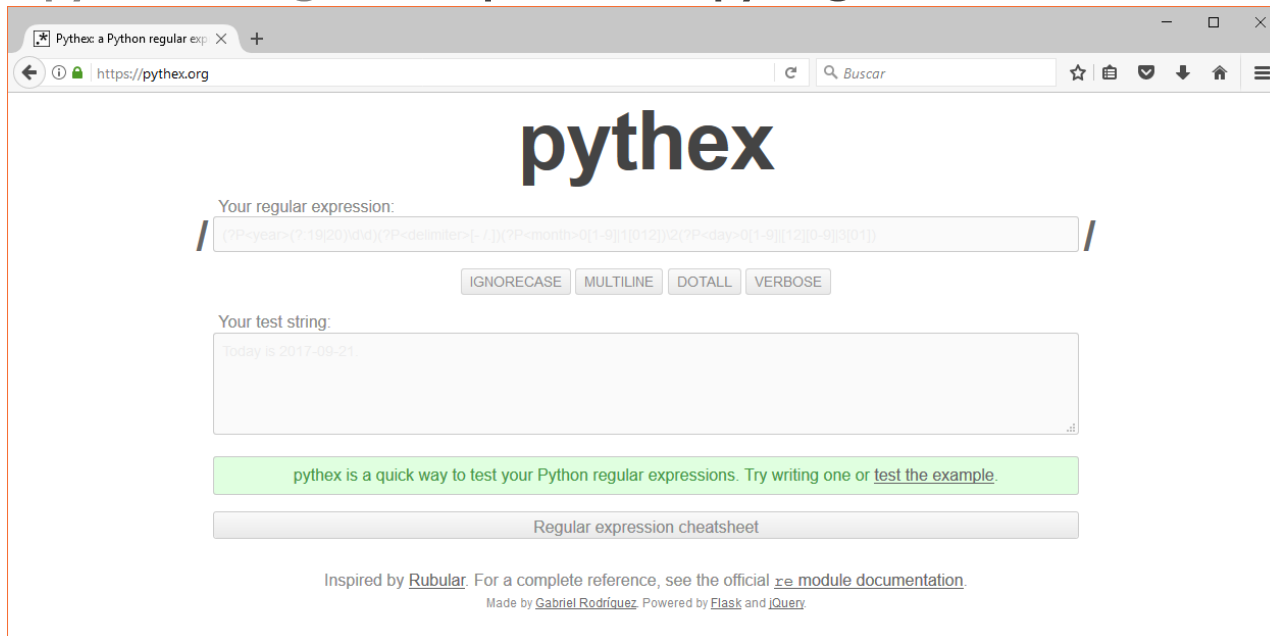
REEMPLAZAR

- La función `re.sub(patrón, reemplazo, cadena)` nos permite localizar patrones en un texto y sustituirlos por otra cadena de reemplazo.
- Dentro de la cadena de reemplazo podemos utilizar `\1`, `\2...` como referencia a los grupos localizados en el patrón.
- Ejemplo:

```
cadena=re.sub(r'<p>(.*?)</p>','<div> \1</div>','<p> Cambiamos de etiqueta</p>')
```

EDITORES DE EXPRESIONES REGULARES

- En ocasiones no es fácil diseñar una expresión regular que concuerde.
- Para practicar y ayudarnos tenemos algunos editores online:
<https://pythex.org/> ó <http://www.pyregex.com/>



ENTORNOS VIRTUALES: VIRTUALENV

- Cuando programamos una aplicación queremos poder desplegarla fácilmente en cualquier equipo.
- Eso implica no sólo instalar Python sino los paquetes que usamos en esa aplicación.
- En otras ocasiones queremos trabajar con distintas versiones de Python o incluso de un mismo paquete.
- Para eso debemos crear entornos virtuales de Python.

CREAR UN ENTORNO VIRTUAL

- Necesitamos tener instalado el paquete virtualenv.
- Creamos un directorio para el proyecto.
- Creamos el entorno virtual con la orden virtualenv seguido de un directorio para el entorno.
- Activamos el entorno virtual.

```
C:\Users\Llanos\Documents\Python>mkdir proyecto  
  
C:\Users\Llanos\Documents\Python>cd proyecto  
  
C:\Users\Llanos\Documents\Python\proyecto>virtualenv env  
New python executable in C:\Users\Llanos\Documents\Python\proyecto\env\Scripts\python.exe  
Installing setuptools, pip, wheel...done.
```

ACTIVAR UN ENTORNO VIRTUAL

- Bajo Windows: env/Scripts/actíivate
- Bajo Linux:
 - CentOS: source env/bin/actíivate
 - Otros: ./env/bin/actíivate
- Bajo MaC: env/bin/activate

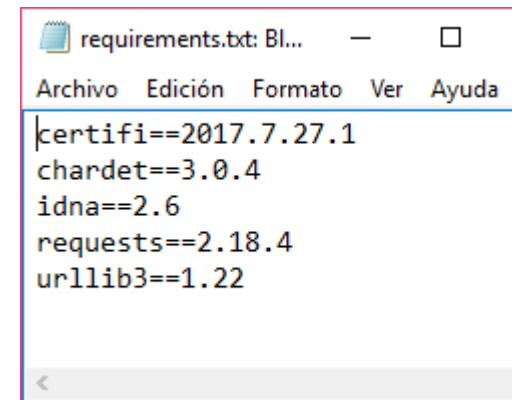
```
C:\Users\Llanos\Documents\Python\proyecto>cd env
C:\Users\Llanos\Documents\Python\proyecto\env>cd Scripts
C:\Users\Llanos\Documents\Python\proyecto\env\Scripts>activate.bat
(env) C:\Users\Llanos\Documents\Python\proyecto\env\Scripts>
```

GESTIÓN DE ENTORNO VIRTUALES

- Dentro del entorno virtual podemos instalar los paquetes como si lo hiciéramos en la versión global de Python.
- Para guardar la configuración del sistema usamos la orden:

`pip freeze > requirements.txt`

```
(env) C:\Users\Llanos\Documents\Python\proyecto>pip freeze > requirements.txt  
(env) C:\Users\Llanos\Documents\Python\proyecto>
```



A screenshot of a text editor window titled 'requirements.txt: Bl...'. The window has a menu bar with 'Archivo', 'Edición', 'Formato', 'Ver', and 'Ayuda'. The text content is as follows:

```
certifi==2017.7.27.1  
chardet==3.0.4  
idna==2.6  
requests==2.18.4  
urllib3==1.22
```

- Para instalar un proyecto en un nuevo equipo:
 - Creamos un entorno virtual con normalidad.
 - Instalamos todas las dependencias con la orden: `pip install -r requirements.txt`

DOCUMENTACIÓN

- Cómo ya comentamos en otro momento todos los objetos de Python cuentan con una variable especial llamada `__doc__` que contiene una cadena de texto que se muestra cuando llamamos a la ayuda.
- También podemos generar la documentación de un módulo con la herramienta `pydoc`.
 - Podemos exportarla a una pagina web, de forma similar a Java, con la siguiente orden:

```
pydoc.-w módulo módulo ...
```

EJEMPLO

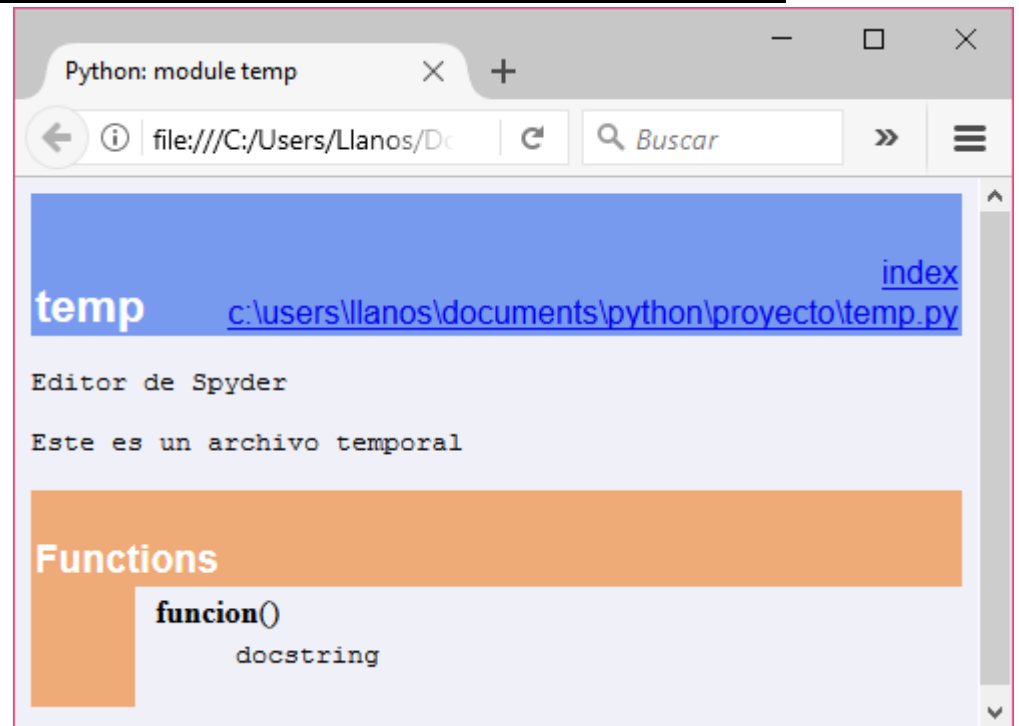
```
(env) C:\Users\Llanos\Documents\Python\proyecto>pydoc -w temp  
wrote temp.html
```

```
# -*- coding: utf-8 -*-  
"""
```

Editor de Spyder

Este es un archivo
temporal
"""

```
def funcion():  
    """ docstring """  
    return 1
```



SERIALIZACIÓN DE OBJETOS

- En ocasiones necesitamos guardar parte de los datos almacenamos en las variables de nuestro programa en un fichero para continuar más tarde.
 - Guardar el estado de un programa para continuar su ejecución más tarde.
 - Guardar un modelo predictivo una vez entrenado para utilizarlo de diversos sistemas.
 - Convertir cualquier objeto de Python en una cadena para poder acceder como si fuera un diccionario
 - ...
- Este proceso se denomina serialización.

SERIALIZACIÓN: PICKLE

- Para utilizarlo tenemos que invocar la librería Pickle.

```
import pickle
```

- Una vez tenemos listo el objeto a serializar podemos volcarlo a un fichero:

```
fichero=open("path",'wb')  
pickle.dump(objeto,fichero)  
fichero.close()
```

- Para recuperar un objeto desde fichero usamos la orden:

```
fichero=open("path",'rb')  
objeto=pickle.load(fichero)  
fichero.close()
```

- Cuidado con las excepciones relacionadas con la codificación.

RECURSOS PARA APRENDER PYTHON

- Tutorial Python por Google (inglés): <https://developers.google.com/edu/python/>
- CodeAcademy para Python: <https://www.codecademy.com/es/tracks/python-traduccion-al-espanol-america-latina-clone-1>
- Intro to Python for Data Scientists (DataCamp - Inglés): <https://www.datacamp.com/courses/intro-to-python-for-data-science>
- Libros:
 - Python para todos: <http://mundogeek.net/tutorial-python/>
 - Oficial de Python argentina: <http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf>
 - Python para Informáticos: <http://do1.dr-chuck.net/py4inf/ES-es/book.pdf>
 - Curso de Python para principiantes: <http://www.iaa.es/python/curso-python-para-principiantes.pdf>