



Tema 0 Introducción

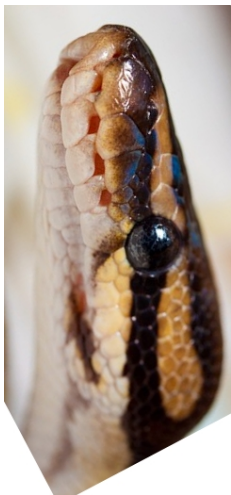
Curso de Python Avanzado

Juan Pedro Bolívar Puente

Instituto de Astrofísica de Andalucía

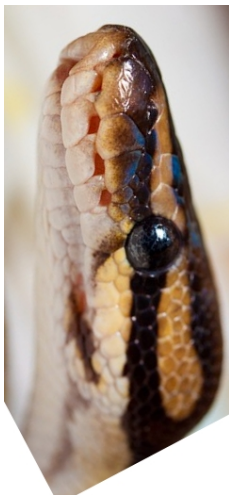
Mayo de 2011

Índice



- 1 Introducción
- 2 Herramientas
- 3 Repaso
- 4 Módulos
- 5 Documentación
- 6 Ejemplos

Índice



- 1 **Introducción**
- 2 Herramientas
- 3 Repaso
- 4 Módulos
- 5 Documentacion
- 6 Ejemplos

Página web

`http://www.sinusoid.es/python-avanzado`

Tutorías

`raskolnikov@gnu.org`

¿Dónde?

Sala de Juntas del Instituto de Astrofísica de Andalucía

¿Cuándo?

9 a 13 de Mayo de 2011, de 9:00 a 14:30 (25 horas)

Objetivos

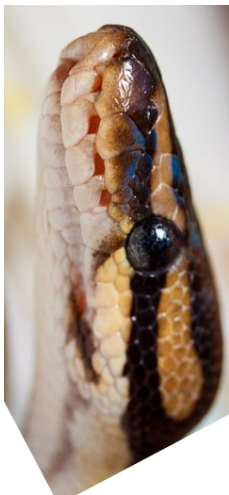
- Aprender el lenguaje con **profundidad**
- Aprender a hacer **aplicaciones grandes**
'Idioms', modularidad, organización
- Aprender a hacer **aplicaciones modernas**
Interfaces gráficas, web
- Aprender a hacer **aplicaciones buenas**
Pruebas, eficiencia, ...

¡En Python!

¡Volvemos **renacentistas!**



Índice



- 1 Introducción
- 2 Herramientas**
- 3 Repaso
- 4 Módulos
- 5 Documentación
- 6 Ejemplos

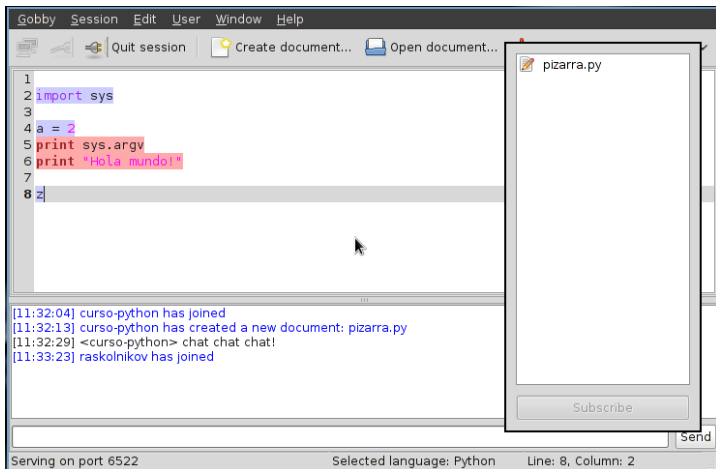
IPython es tu amigo

Se recomienda usar `ipython`

```
In [0]: %quickref
In [1]: %alias
In [2]: ls
In [3]: !top
In [4]: print _i, __i, _ih[1:2]
In [5]: dir list
In [6]: %who
In [7]: open ('t<TAB>
In [8]: %timeit
```


Pizarra virtual

`gobby` nos servirá para editar colaborativamente



The screenshot shows the Gobby text editor window. The menu bar includes 'Gobby', 'Session', 'Edit', 'User', 'Window', and 'Help'. The toolbar contains icons for 'Quit session', 'Create document...', and 'Open document...'. The main text area contains a Python script:

```
1
2 import sys
3
4 a = 2
5 print sys.argv
6 print "Hola mundo!"
7
8 z
```

Below the text area is a chat log with the following messages:

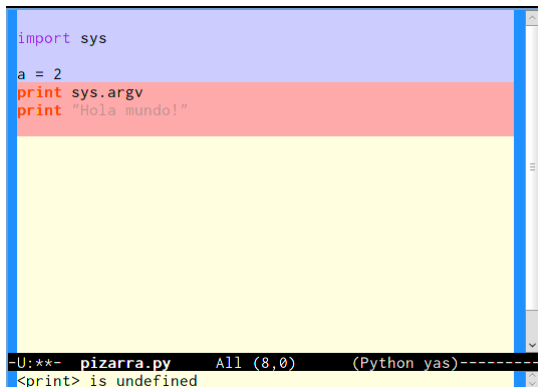
```
[11:32:04] curso-python has joined
[11:32:13] curso-python has created a new document: pizarra.py
[11:32:29] <curso-python> chat chat chat!
[11:33:23] raskolnikov has joined
```

A floating window titled 'pizarra.py' is open over the text area, containing a 'Subscribe' button. At the bottom of the Gobby window, there is a status bar with the text 'Serving on port 6522', 'Selected language: Python', and 'Line: 8, Column: 2'. A 'Send' button is located at the bottom right of the chat log area.

El mejor editor del mundo ...

Es delicioso usar todo esto en `emacs`

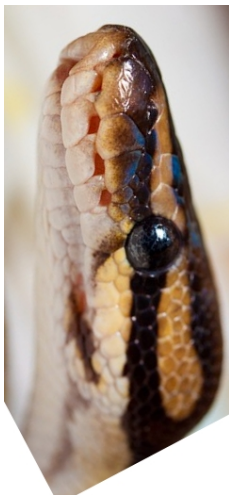
- Integra `ipython`
- Ejecuta código directamente
- Resalta errores
- Se conecta con `gobby`



```
import sys
a = 2
print sys.argv
print "Hola mundo!"
```

-U:***- pizarra.py All (8,0) (Python yas)-----
<print> is undefined

Índice



- 1 Introducción
- 2 Herramientas
- 3 Repaso**
- 4 Módulos
- 5 Documentacion
- 6 Ejemplos

Valores y operadores

```
1 == 1
1 is 1
[1] == [1]
[1] is [1]
type (1.2)
a = 2
a += 1
print a
```

Listas y cadenas

```
s = 'abc'
s.replace ('b', 'B')
print s
'hola' is 'hola'
s2 = list (s)
s2[1] = 'B'
print s2
print ''.join (s2)
range (2, 22, 2)
[3, 'hola', 5.2]
[ x**2 for x in range(10) if x%2==0]
if []:
    print "No se ejecuto"
```

Diccionarios

```
a = { 'java' : 'inquisicion',  
      'python' : 'renacimiento',  
      'lisp' : 1958,  
      'cobol' : None }  
a['python'] == 'renacimiento'  
a['lisp'] += 1  
del a['java']  
b = dict([(1, 9), (1, 10), (2, 20)])  
n = 'Juan_Pedro_Bolivar_Puente'  
dict(zip(range(len(n)), n))
```

Bucles e IO

```
for idx in range (5):  
    print idx
```

```
for key, val in b.iteritems ():  
    print key, '->', val
```

```
while True:  
    s = raw_input ('Dime _guapo:_')  
    if s.lower () == 'guapo':  
        break
```

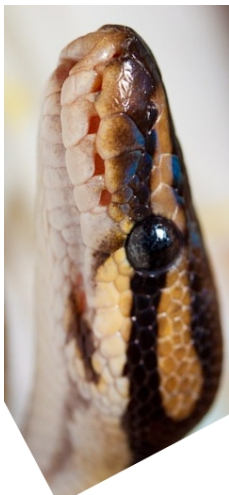
Funciones e IO

```
import sys
def acrostico (fname):
    f = open (fname)
    for x in f.readlines ():
        sys.stdout.write (
            x[0].replace ('\n', '□'))
```


Funciones

```
def saludar (despedida = False):  
    if despedida:  
        print "Adios_mundo!"  
    else:  
        print "hola_mundo!"  
  
saludar ()  
saludar (True)  
saludar (despedida = True)
```

Índice



- 1 Introducción
- 2 Herramientas
- 3 Repaso
- 4 Módulos**
- 5 Documentacion
- 6 Ejemplos

Módulos

Modulo = Espacio de nombres

Agrupar funciones, clases, constantes, globales ...

Ejemplo: Importar un módulo

```
import os
print os.name
```

Ejemplo: Importar un nombre

```
from os import name
print name
```

Modulos

Ejemplo: Importar un submódulo

```
import os.path
from os import path
print path.join ('una', 'ruta')
```

Ejemplo: Renombrado

```
import os.path as p
print p.exists ('/')
```

Modulos

Terminologia

Modulo Módulo *hoja*.

Es un **fichero** `.py`

Paquete Módulo con *submódulos*.

Es un directorio con **fichero** `__init__.py`

¡Atención!

A veces usaremos indistintamente el termino **módulo**

Búsqueda de módulos

Busca en todos los directorios de `sys.path`

- 1 Para los `modulos`, busca un fichero `.py` con su nombre en el directorio actual.
- 2 Para los `paquetes`, busca un subdirectorio con un fichero `__init__.py` en el directorio actual.

Si no encuentra tira `ImportError`

```
sys.path = [ os.getcwd () ] + \  
            PYTHONPATH + \  
            installation dependent
```

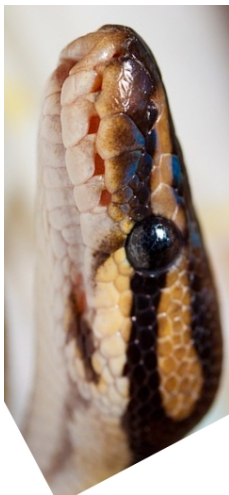
Carga de un módulo

- 1 El módulo `se ejecuta!`
- 2 En un paquete, se `ejecuta __init__.py`.
- 3 Los nombres creados \Rightarrow `objeto módulo`.
- 4 `__name__` contiene el nombre de nuestro módulo.
- 5 Módulo raíz se llama `__main__`

Patrón “funcion main”

```
if __name__ == '__main__':  
    ejecutar_funcion_main ()
```

Índice



- 1 Introducción
- 2 Herramientas
- 3 Repaso
- 4 Módulos
- 5 Documentacion**
- 6 Ejemplos

Documentación

Docstring

La primera cadena de cada **módulo, clase o función** es considerada de documentación.

- Acceso
 - `help (objeto)` ⇒ interactivo.
 - `objeto.__doc__` ⇒ programático.
- Usar cadenas triples “ “ “

Recomendaciones

Usar marcadores estilo *JavaDoc*

`@param x`: Un parametro 'x'

`@return` Valor de retorno.

`@raise e` Tira tal tipo de excepción.

`@author` Autor de un modulo, clase, funcion...

`@date` Fecha de creacion.

Recomendaciones

Usar marcadores estilo *Sphinx*

`:param x:` Un parametro 'x'

`:return:` Valor de retorno.

`:raise e:` Tira tal tipo de excepción.

`:author:` Autor de un modulo, clase, funcion...

`:date:` Fecha de creacion.

Recomendaciones

Usar estilos reStructuredText

```
Titulo
```

```
=====
```

```
Subtitulo
```

```
-----
```

```
* Elemento 1
```

```
* Elemento 2
```

```
1. Ordenado 1
```

```
2. Ordenado 2
```

```
*cursiva*, **negrita**, \*, ...
```

Manejándose con la documentación

`pydoc` permite ver la documentación de cualquier cosa desde `la consola` o `el navegador`

```
$ pydoc modulo
```

```
$ pydoc -w fichero ...
```

```
$ pydoc -g
```

```
$ pydoc -p <port>
```

ReStructuredText

`doctutils` permite trabajar con ficheros *ReST*
Es una biblioteca y además trae `utilidades`

```
$ rst2html fich.rst > fich.html
```

```
$ rst2latex fich.rst > fich.tex
```

```
$ rst2odt fich.rst > fich.tex
```

... etc ...

Otras herramientas ...

Epydoc

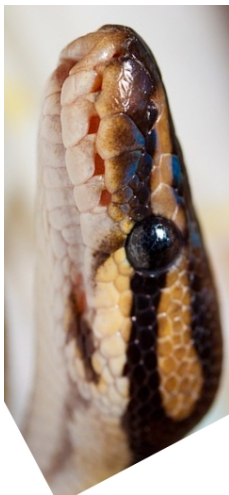
- Automática, sencilla de usar
- Permite sintáxis JavaDoc
- Ejemplo

<http://epydoc.sourceforge.net/stdlib/>

Sphinx

- Menos automática
- Produce documentación de más calidad
- Ejemplo <http://docs.python.org/library/>

Índice



- 1 Introducción
- 2 Herramientas
- 3 Repaso
- 4 Módulos
- 5 Documentación
- 6 Ejemplos**

Ejemplo

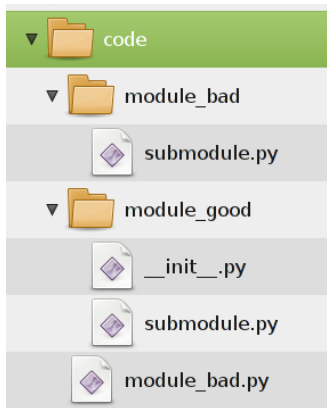


Figura: Modulos de prueba

module_bad

- Usa UTF-8 sin directiva
- No documenta.
- Submodulos sin `__init__.py`
- Función de prueba sin “patrón main”.

module_good

¡Todo lo contrario!

Ejemplo

module_good/__init__.py

```
# -*- coding: utf-8 -*-  
"""  
Este es un modulo muy bueno!  
"""  
  
def fun ():  
    """  
    Esta es una funcion muy util!  
    """  
    print "Hola_mundo!"
```

Ejemplo

```
module_good/__init__.py
```

```
class MyClass (object):  
    """  
    Una clase chula.  
    """  
  
    def method (self, param):  
        """  
        Un metodo fascinante.  
        :param param: Esto mola
```

Ejemplo

```
module_good/__init__.py
```

```
if __name__ == '__main__':  
    print "**_Probando_fun_"  
    fun ()
```

Realizando pruebas ...

```
import module_bad
import module_good

try:
    import module_bad.submodule
except ImportError:
    print "No encuentra el modulo"

import module_good.submodule
module_good.submodule.good_func ()
```





Realizando pruebas ...

```
help (module_good)
help (module_good.fun)
help (module_good.MyClass)

print module_good.__doc__
print module_good.fun.__doc__
print module_good.MyClass.__doc__

print module_good.__name__
print module_good.fun.__name__
print module_good.MyClass.__name__
```

Recursos adicionales

-  [A Primer on Scientific Programming With Python](#)
Hans Petter Langtangen
Springer, 1st edition (September 2009)
-  [Pro Python](#)
Marty Alchin
APRESS, 1st edition (Jun 2010)
-  [IPython](#)
<http://ipython.scipy.org/>
-  [Docutils](#)
<http://docutils.sourceforge.net>

¿Preguntas?

Muchas gracias por su atención.

