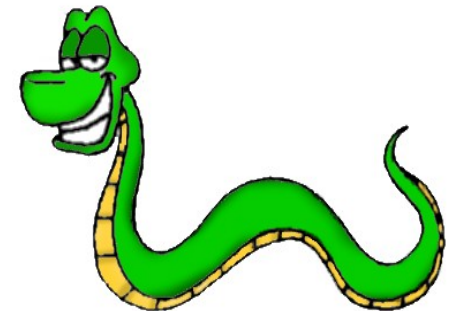


Introducción a Python

LIFE IS SHORT – YOU NEED PYTHON!



- ¿Quién soy?
- ¿Qué es APSL?
- ¿Qué es Python?
- Preguntas
- ¿Cómo seguir?

Sobre mi

- Gerente de APSL
- Antes jefe de proyecto web en TUI España
- Blog: <http://trespams.com>
- Twitter: aaloy
- Correo: aaloy@apsl.net

Aviso de RANT!

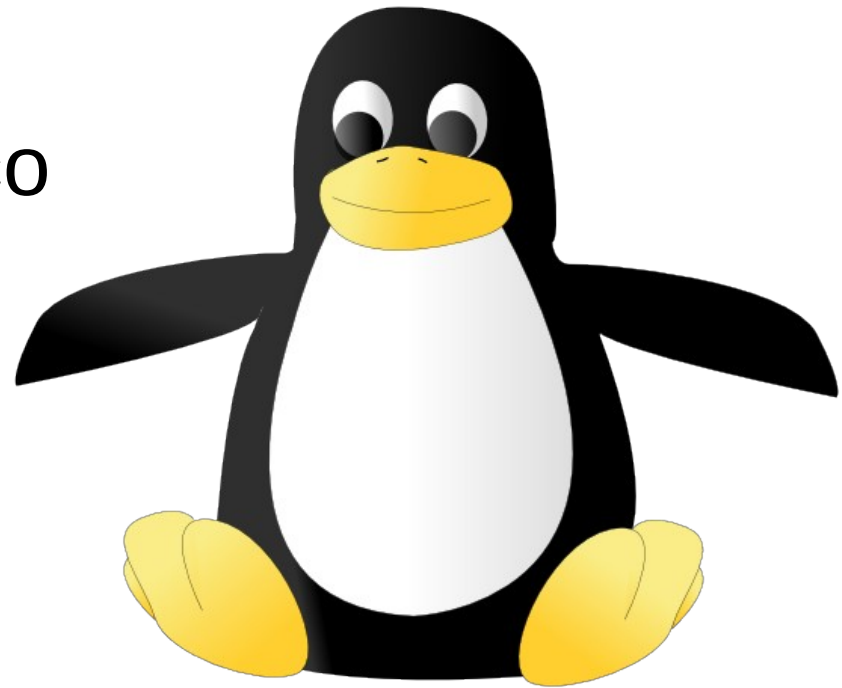


¿Qué es Python?

- Un lenguaje interpretado orientado a objetos
- Una herramienta de desarrollo rápido
- Con una sintaxis lógica, clara y legible
- Libre y multiplataforma
- Creado por Guido Van Rossum al 1991
- Que debe su nombre a los *Monty Python Flying Circus*
- Que viene ya con las baterías incluidas

¿Para qué me puede servir?

- Administrar sistemas
- Crear aplicaciones de escritorio (wx, qt, gtk, tcl/tk)
- Crear webs
- Cálculo numérico y simbólico
- Acceso a base de datos
- Trabajo con Big Data
- Prototipado
- Gráficos



Puntos flacos

- Sistemas en tiempo real
- Sistemas con tiempo de procesamiento crítico
- Todavía hay lenguajes más rápidos (PyPy!)
- El GIL puede ser un problema



Pero hablamos de Python, siempre hay una manera!!!

¿Quién lo utiliza?

- Google
- Dropbox
- Instagram
- Mozilla
- Pinterest
- Disqus
- Zope, Plone, Django
- Distribuciones Linux (RH, Ubuntu,)
- <http://wiki.python.org/moin/OrganizationsUsingPython>

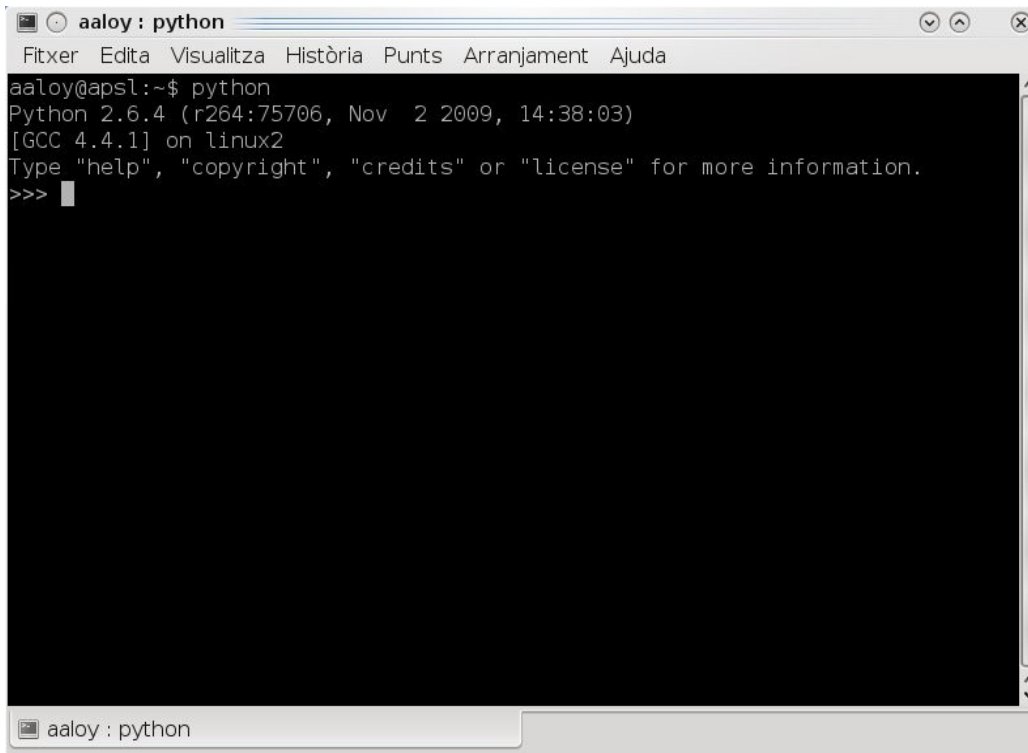


Instalación de Python

- Descarga: <http://python.org>
 - Para Windows
 - En Linux está instalado se serie
- Herramientas y utilidades:
 - ipython
 - easy_install
 - pip
 - ipdb

Empecemos!

>> python

A screenshot of a terminal window titled 'aaloy : python'. The window has a menu bar with options: Fitxer, Edita, Visualitza, Història, Punts, Arranjament, Ajuda. The terminal output shows the command 'python' being executed, resulting in 'Python 2.6.4 (r264:75706, Nov 2 2009, 14:38:03) [GCC 4.4.1] on linux2'. It then displays the prompt '>>>' with a cursor. The window's title bar and a single tab labeled 'aaloy : python' are visible at the bottom.

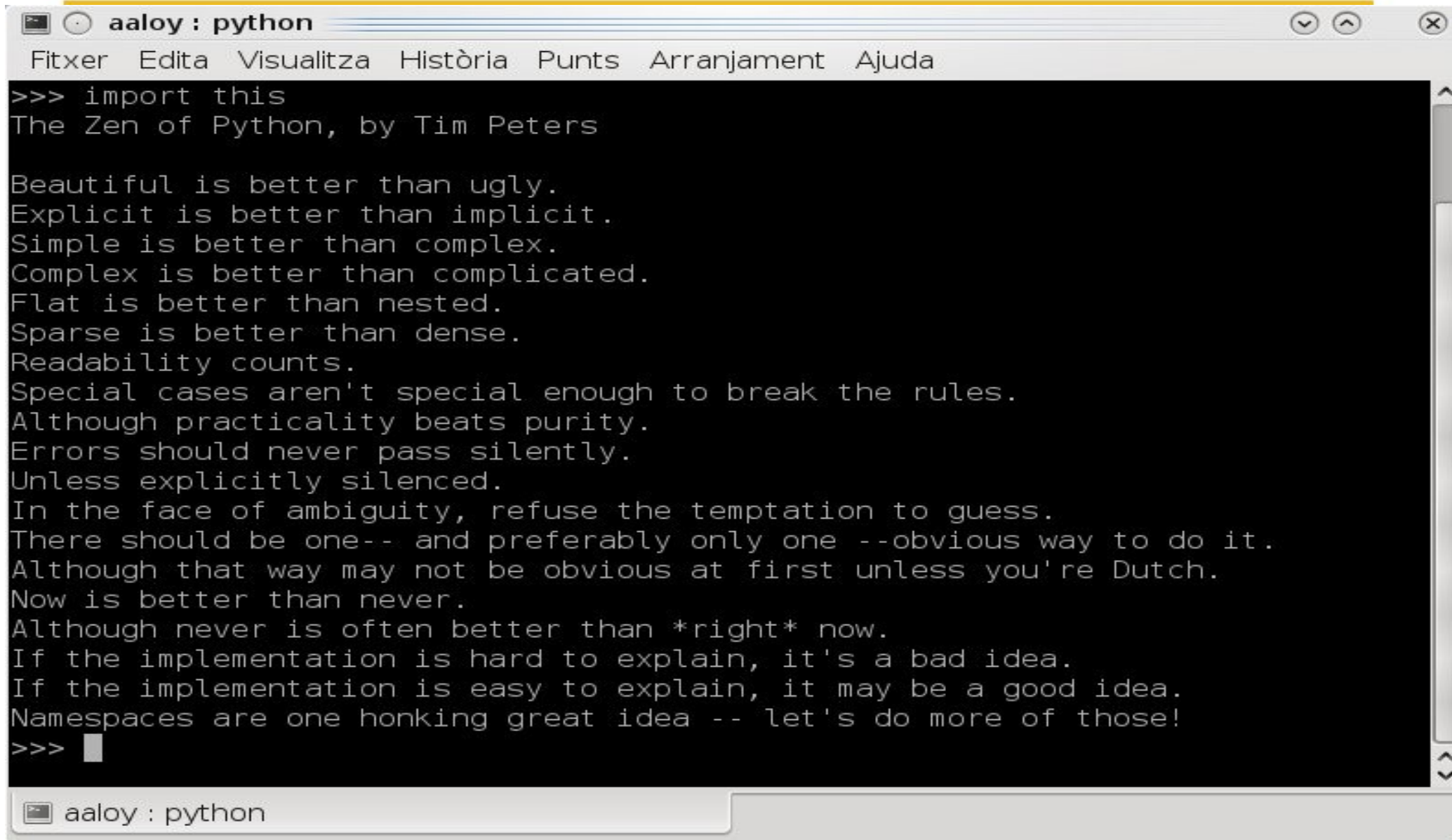
Editores:

- Ninja-Ide
- Eclipse + PyDev
- Vim, Gvim
- Emacs
- Ulipad
- Notepad++
- pyCharm

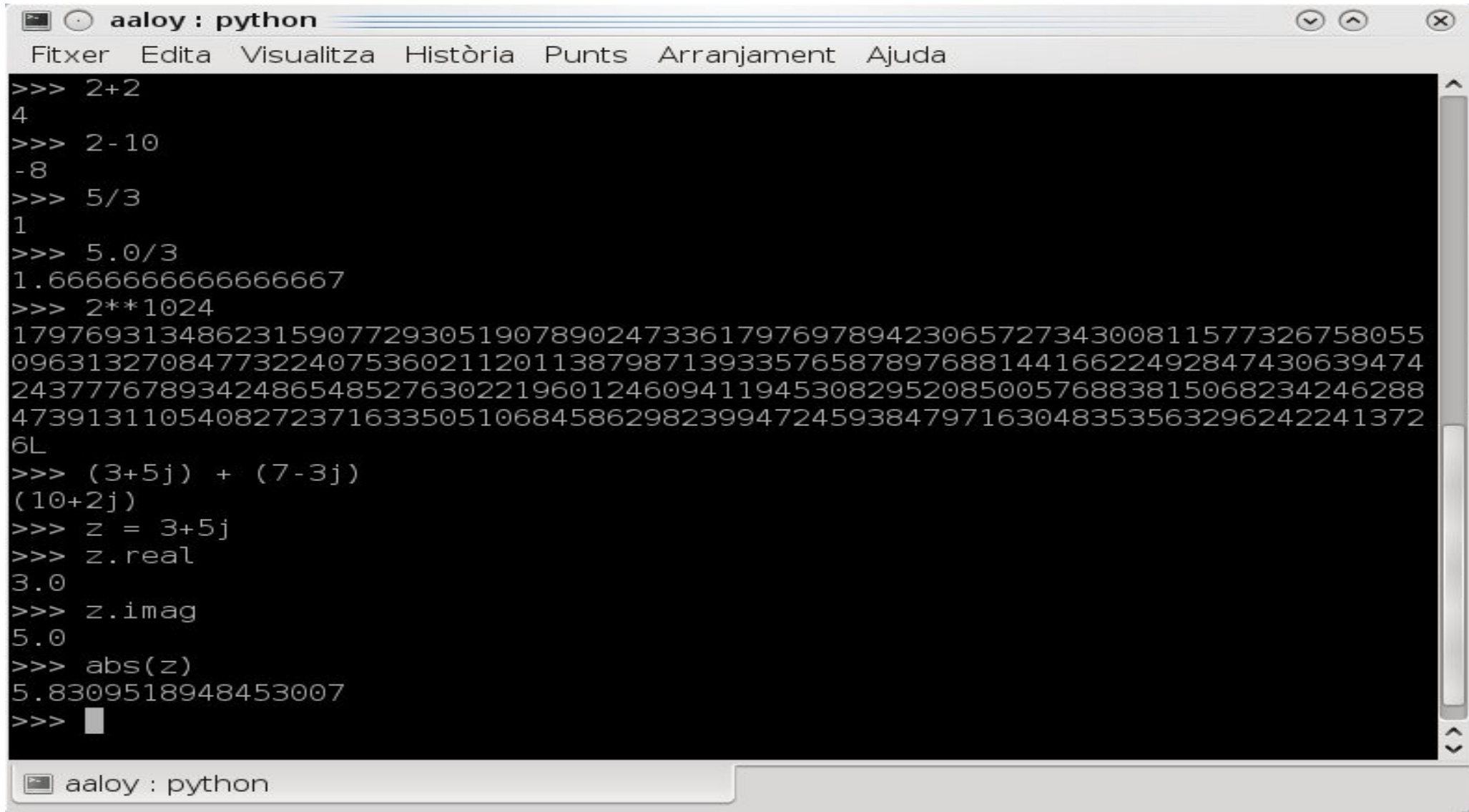


CUALQUIER EDITOR DE TEXTO PLANO SIRVE!!!!

Un poco de Zen

A screenshot of a Python interpreter window titled 'aaloy : python'. The window has a menu bar with options: Fitxer, Edita, Visualitza, Història, Punts, Arranjament, and Ajuda. The main area is a black terminal with white text. It shows the command '>>> import this' followed by the output 'The Zen of Python, by Tim Peters'. Below this is a list of 19 Zen of Python principles. The window has standard macOS window controls (red, yellow, green buttons) in the top right corner. At the bottom, there is a tab labeled 'aaloy : python'.

Como calculadora

A screenshot of a Python shell window titled 'aaloy : python'. The window has a menu bar with options: Fitxer, Edita, Visualitza, Història, Punts, Arranjament, and Ajuda. The main area is a black terminal with white text showing various Python calculations. The calculations include basic arithmetic, floating-point division, large integer exponentiation, and complex number operations. The window has standard OS controls (minimize, maximize, close) in the top right corner and a tab labeled 'aaloy : python' at the bottom left.

```
>>> 2+2
4
>>> 2-10
-8
>>> 5/3
1
>>> 5.0/3
1.6666666666666667
>>> 2**1024
1797693134862315907729305190789024733617976978942306572734300811577326758055
0963132708477322407536021120113879871393357658789768814416622492847430639474
2437776789342486548527630221960124609411945308295208500576883815068234246288
4739131105408272371633505106845862982399472459384797163048353563296242241372
6L
>>> (3+5j) + (7-3j)
(10+2j)
>>> z = 3+5j
>>> z.real
3.0
>>> z.imag
5.0
>>> abs(z)
5.8309518948453007
>>> █
```

- Administración de sistemas
- Cálculo científico
- Consola más amigable
- Con autocompletado
- Multitud de atajos

Demasiado bueno para no usarlo!

```
IPython 0.13.1 -- An enhanced Interactive Python.
?          -> Introduction and overview of IPython's features.
%quickref  -> Quick reference.
help       -> Python's own help system.
object?    -> Details about 'object', use 'object??' for extra details.
```

```
In [1]: import math
```

```
In [2]: dir(math)
```

```
Out[2]:
```

```
['__doc__',
 '__name__',
 '__package__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'copysign',
 'cos',
 'cosh',
 'erf',
 'erfc',
 'exp',
 'exp2',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
 'hypot',
 'isinf',
 'isnan',
 'ldexp',
 'lgamma',
 'log',
 'log10',
 'log2',
 'loggamma',
 'logspace',
 'modf',
 'nan',
 'nextafter',
 'pow',
 'radians',
 'sin',
 'sinh',
 'sqrt',
 'tan',
 'tanh',
 'trunc',
 'zeta']
```

```
In [4]: math.acos?
```

```
Type: builtin_function_or_method
```

```
String Form:<built-in function acos>
```

```
Docstring:
```

```
acos(x)
```

```
Return the arc cosine (measured in radians) of x.
```

```
In [5]: dir(math.acos)
```

```
Out[5]:
```

```
['__call__',
 '__class__',
 '__cmp__',
 '__delattr__',
 '__doc__',
 '__getattribute__',
 '__hash__',
 '__init__',
 '__int__',
 '__long__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__']
```

Los comandos *help* y *dir*

Hablemos de la indentación

- Forma parte del language
- Refuerza la legibilidad
- Sólo debemos configurar nuestro editor
 - tabs a espacios
 - 4 espacios por tab
 - y de paso lo configuraremos a **utf-8**
 - y retorno de carro tipo unix



Conozcamos el lenguaje

Cadenas

```
In [1]: "una cadena"
```

```
Out[1]: 'una cadena'
```

```
In [2]: "podemos utilizar 'las comillas' de manera consistente"
```

```
Out[2]: "podemos utilizar 'las comillas' de manera consistente"
```

```
In [3]: 'o ' + 'concatenar' + 'cadenas' + "sin problemas"
```

```
Out[3]: 'o concatenarcadenassin problemas'
```

```
In [6]: u"fácil" u'muy fácil'
```

```
Out[6]: u'f\xe1cilmuy f\xe1cil'
```

```
In [7]: """incluso podemos tener cadenas de más  
de una línea de texto y no pasa nada. Para eso  
están las triples comillas"""
```

```
Out[7]: 'incluso podemos tener cadenas de m\xe1s  
de una l\xednea de texto y no pasa nada. Para eso\nest\xe1n las triples  
comillas'
```

Números

```
In [1]: 2+2
```

```
Out[1]: 4
```

```
In [2]: 2**22
```

```
Out[2]: 4194304
```

```
In [3]: 2.0*28838383838383+3**6
```

```
Out[3]: 57676767677495.0
```

```
In [4]: 1/3
```

```
Out[4]: 0
```

```
In [5]: 1/3.0
```

```
Out[5]: 0.3333333333333333
```

```
In [6]: (3+3j) * (4-1j)
```

```
In [7]: (5+2.5j).real
```

```
Out[7]: 5.0
```

```
In [9]: import math  
math.sin(3)
```

```
Out[9]: 0.1411200080598672
```

```
In [10]: math.pi
```

```
Out[10]: 3.141592653589793
```

Todo es un objeto

- cadenas
- números
- Funciones
- Módulos
- ...



Utilizaremos `dir(objeto)` para ver sus propiedades

dir y help

```
In [1]: i

In [1]: import math

In [2]: dir(math)
Out[2]:
['_doc_',
 '_name_',
 '_package_',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'clear_type_cache',
 'current_frames',
 'getframe',
 'mercurial',
 'multiarch',
 'api_version',
 'argv',
 'builtin_module_names',
 'byteorder',
 'call_tracing',
 'callstats']

In [9]: from os import sys

In [10]: dir(sys)
Out[10]:
['_displayhook_',
 '_doc_',
 '_egginsert_',
 '_excepthook_',
 '_name_',
 '_package_',
 '_plen_',
 '_stderr_',
 '_stdin_',
 '_stdout_',
 '_clear_type_cache',
 '_current_frames',
 '_getframe',
 '_mercurial',
 '_multiarch',
 '_api_version',
 '_argv',
 '_builtin_module_names',
 '_byteorder',
 '_call_tracing',
 '_callstats']
```

Help on built-in function fsum in module math:

```
fsum(...)
    fsum(iterable)
```

Return an accurate floating point sum of values in the iterable.
Assumes IEEE-754 floating point arithmetic.

Help on built-in function copysign in module math:

```
copysign(...)
    copysign(x, y)
```

Return x with the sign of y.

(END)

Slice de cadenas

- Les cadenes són inmutables
- Podemos acceder a sus elementos tratándolas como vectores
- El *slicing* nos permite acceder a un elemento o a varios de ellos
 - cadena[0]
 - cadena[1:3]
 - cadena[-1]

Ejemplos

```
In [1]: x = "soy una cadena"
```

```
In [2]: x[0]
```

```
Out[2]: 's'
```

```
In [4]: x[0:3]
```

```
Out[4]: 'soy'
```

```
In [6]: x[1:3]
```

```
Out[6]: 'oy'
```

```
In [7]: x[4:]
```

```
Out[7]: 'una cadena'
```

```
In [8]: x[: -1]
```

```
Out[8]: 'soy una caden'
```

```
In [13]: x[-3:]
```

```
Out[13]: 'ena'
```

```
In [11]: x[::-1]
```

```
Out[11]: 'anedac anu yos'
```

```
In [16]: x[2:7:2]
```

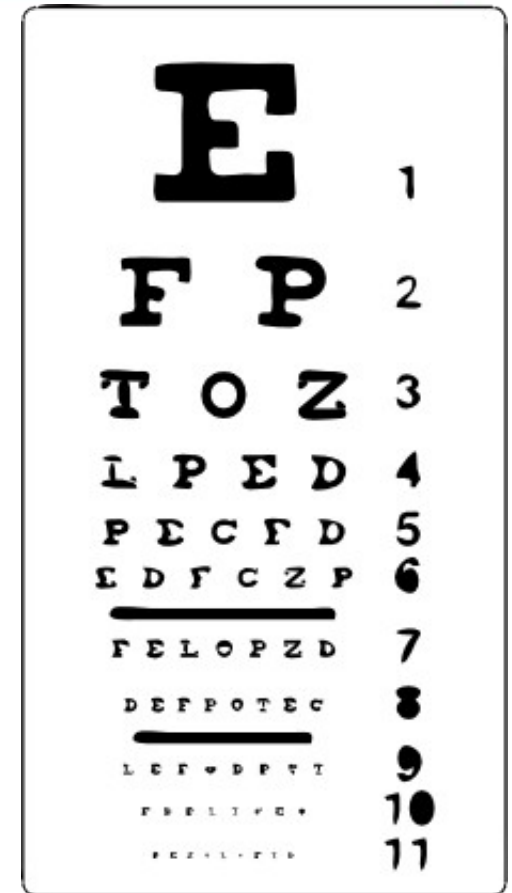
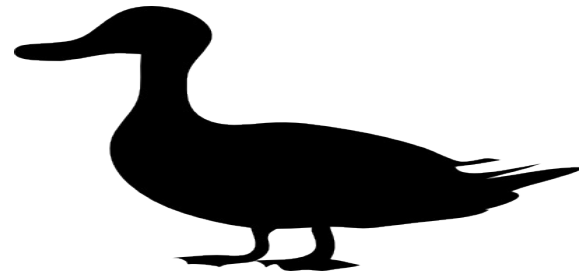
```
Out[16]: 'yua'
```

```
In [17]: len(x)
```

```
Out[17]: 14
```

Listas y tuplas

- Las listas son mutables
- Las tuplas inmutables
- Pueden contener cualquier objeto
- Y permiten slicing



Ejemplo

```
In [1]: hola=list()
```

```
In [2]: hola
```

```
Out[2]: []
```

```
In [3]: hola.append("primer elemento")
```

```
In [4]: hola.append(2)
```

```
In [5]: hola
```

```
Out[5]: ['primer elemento', 2]
```

```
In [6]: ['esto', 2, (3+2.0j), lambda x: x + 1, ]
```

```
Out[6]: ['esto', 2, (3+2j), <function __main__.<lambda>>]
```

```
In [8]: milista = ['esto', 2, (3+2.0j), lambda x: x + 1, ]  
milista[3](3)
```

```
Out[8]: 4
```


Ejemplo de listas y tuplas

```
In [8]: milista = ['esto', 2, (3+2.0j), lambda x: x + 1, ]  
milista[3](3)
```

```
Out[8]: 4
```

```
In [9]: x = (1, 3, 'hola')  
print x  
  
(1, 3, 'hola')
```

```
In [10]: milista[0]=x  
print milista  
  
[(1, 3, 'hola'), 2, (3+2j), <function <lambda> at 0x27ce230>]
```

```
In [11]: x[0]=3
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-11-f6c8a3109560> in <module>()  
----> 1 x[0]=3
```

```
TypeError: 'tuple' object does not support item assignment
```

Podemos operar

- Sumar (+)
- Hacer slice
- Añadir elementos *lista.append(objeto)*
- Borrar elementos (*del llista[x]*)
- ordenarlos *llista.sort()*
- Saber su longitud *len(lista)*

- Se definen con dict
- O creándolos directamente
- Permiten almacenar cualquier objeto utilizando un índice (hastable)

`dir(diccionario)`

Ejemplos

```
In [2]: x = dict()
        print x

        {}
```

```
In [3]: z = {}
        print z

        {}
```

```
In [4]: x = {"un": 1, 2: "hola", 'p': [1, 2, 3,] }
        print x

        {'p': [1, 2, 3], 2: 'hola', 'un': 1}
```

```
In [5]: x['p']
```

```
Out[5]: [1, 2, 3]
```

```
In [6]: x.items()
```

```
Out[6]: [('p', [1, 2, 3]), (2, 'hola'), ('un', 1)]
```

```
In [7]: x.keys()
```

```
Out[7]: ['p', 2, 'un']
```

```
In [8]: x.values()
```

```
Out[8]: [[1, 2, 3], 'hola', 1]
```

Ejemplos

```
In [2]: x = dict()  
        print x  
  
        {}
```

```
In [3]: z = {}  
        print z  
  
        {}
```

```
In [4]: x = {"un": 1, 2: "hola", 'p': [1, 2, 3,] }  
        print x  
  
        {'p': [1, 2, 3], 2: 'hola', 'un': 1}
```

```
In [5]: x['p']
```

```
Out[5]: [1, 2, 3]
```

```
In [6]: x.items()
```

```
Out[6]: [('p', [1, 2, 3]), (2, 'hola'), ('un', 1)]
```

```
In [7]: x.keys()
```

```
Out[7]: ['p', 2, 'un']
```

```
In [8]: x.values()
```

```
Out[8]: [[1, 2, 3], 'hola', 1]
```

I todavía hay más ...

- En las librerías estándar
- Tipos de datos optimizados
- `import collections`

<code>namedtuple()</code>	factory function for creating tuple subclasses with named fields	<i>New in version 2.6.</i>
<code>deque</code>	list-like container with fast appends and pops on either end	<i>New in version 2.4.</i>
<code>Counter</code>	dict subclass for counting hashable objects	<i>New in version 2.7.</i>
<code>OrderedDict</code>	dict subclass that remembers the order entries were added	<i>New in version 2.7.</i>
<code>defaultdict</code>	dict subclass that calls a factory function to supply missing values	<i>New in version 2.5.</i>

Estucturas de control: IF

if condicion:

#

elif condicion:

#

elif condicion:

#

else:

#

```
In [3]: x = 4
        if x > 3:
            print "mayor que 3"
        else:
            print "otros casos"
```

mayor que 3

```
In [2]: print "mayor que 3" if x > 3 else "no es mayor que 3"
```

mayor que 3

```
In [4]: if x == 1:
        print "es uno"
        elif x == 2:
            print "es dos"
        elif x == 3:
            print "es tree"
        else:
            print "muchos"
```

muchos

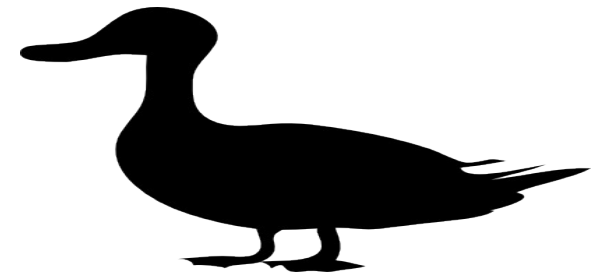
A efectos de comparación se considera falso: None, False, cero o vacío y se considera verdadero True, un conenedor no vacío o distinto de cero.

for

- Nos permite recorrer cualquier cosa que permita la iteración, i.e. Cualquier iterable.

```
for elemento in lista_iterable:  
    codigo  
else:  
    codi
```

break i continue nos permite modificar el flujo



Ejemplos FOR

```
In [23]: import random
z = random.randint(1, 100)
for x in range(1, random.randint(1, 50)):
    if x % 11 == 0:
        print z
        break
    else:
        print "no encontrado"
```

60

```
In [24]: lista = ['un', 3, 'siete', 1.3]
```

```
In [26]: for elemento in lista:
        if type(elemento) == int:
            print u"%s es un número entero" % elemento
```

3 es un número entero

```
In [27]: x = {1:"uno", 2: "dos"}
for k,v in x.items():
    print "%s: %s" % (k, v)
```

1: uno
2: dos

While

while condicion:

codigo

else:

codigo

```
In [7]: lista = [30, 12, 3, 4, 4, 7, 1, 3]
while len(lista) > 3:
    z = lista.pop()
    print len(lista)
    print z
    print lista
    if z > 10:
        print z
        break
else:
    print "No encontrado"
```

```
7
3
[30, 12, 3, 4, 4, 7, 1]
6
1
[30, 12, 3, 4, 4, 7]
5
7
[30, 12, 3, 4, 4]
4
4
[30, 12, 3, 4]
3
4
[30, 12, 3]
No encontrado
```

Funciones

Definición de la función

```
def fibo(n):  
    """Calcula elemento fibonaci  
    An = An-1 + An-2  
    Parámetros: n entero  
    """  
    if n == 1:  
        return 1  
    elif n==0:  
        return 0  
    else:  
        return fibo(n-1)+fibo(n-2)
```

documentación

retorno

- La funciones son objetos
- Podemos tener parámetros con valores por defecto

Ejemplos de funciones

```
def random_unicode_words(num):
    """Deuevle el número de palabras especificado en unicode"""
    return _random_words(num).decode('utf8')

def random_slug(num=2):
    slug = random_words(num).replace(" ", "-")
    return ''.join(ch for ch in slug if ch.isalnum() or ch == '-')

def random_slugs(num):
    return [random_slug() for x in range(num)]

def random_url(sufix=".com", params=0):
    data = (random_slug(), sufix, random_slug())
    simple_url = "http://%s%s/%s/" % data
    if params>0:
        param_url = "&".join(["%s=%s" % (random_slug(1),
            random_words(1)) for x in range(params)])
        return "%s?%s" % (simple_url, param_url)
    else:
        return simple_url
```

```
In [22]: def funcion(x, y=20, *args, **kwargs):
....:     print x
....:     print y
....:     print args
....:     print kwargs
....:
```

Funciones II

```
In [25]: funcion(10)
10
20
()
{}
```

```
In [22]: def funcion(x, y=20, *args, **kwargs):
.....:     print x
.....:     print y
.....:     print args
.....:     print kwargs
.....:
```

```
In [26]: funcion(y=10)
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-26-11c90cd20b73> in <module>()
----> 1 funcion(y=10)

TypeError: funcion() takes at least 1 argument (1 given)
```

```
In [30]: funcion(10, 30)
10
30
()
{}
```

```
In [32]: funcion(y=5, x=10)
10
5
()
{}
```

```
In [34]: funcion(5, 10, 20, 30, 60, 80)
5
10
(20, 30, 60, 80)
{}
```

```
In [35]: funcion(20, z=3, p=4)
20
20
()
{'p': 4, 'z': 3}
```

```
In [36]: funcion(1,2,3,4,5,6,7,8, k=18, z=66)
1
2
(3, 4, 5, 6, 7, 8)
{'k': 18, 'z': 66}
```

- Podemos crear programas y guardarlos en un archivo

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
# autor:
# licence:

if __name__ == '__main__':
    pass
```

- Configuramos el editor: tabs a espacios
- Tabs a 4 espacios
- UTF-8 i salto de línea unix

PОО: clases

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
```

```
class Test(object):
    "Clase de test"
    def __init__(self, a, b):
        self.a = a
        self.b = b
```

```
    def suma(self):
        return self.a+self.b
```

```
if __name__ == '__main__':
    p = Test(2, 5)
    print p.suma()
    p = Prova('hola', ' como estamos')
    print p.suma()
```

constructor

atributos

método



POO (2)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

class Test(object):
    "Clase de Test"
    VALOR = 10
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def suma(self):
        "método ligado a la instancia"
        return self.a+self.b

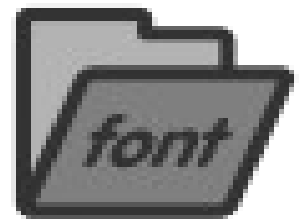
    @classmethod
    def where_am_i(cls):
        "Método asociado a la clase"
        return __name__

    @property
    def major(self):
        if self.a >= self.b:
            return self.a
        else:
            return self.b
```


Módulos

- `from math import *`
- `from math import sin`
- `import math`

`sin(10)` en los dos primers casos
`math.sin(10)` en el segundo



Qué nos falta?

- Herencia y polimorfismo
- Decoradores, generadores, iteradores
- List Comprehension
- Manipulación de archivos
- Creación de módulos
- Excepciones
- Librerías básicas
- Depuración ...



Listos para aprender

- En 1 hora ya podemos leer código
- Empezar a crear scripts
- Leer la documentación

#11900 You cannot just paste code with no understanding of what is going on and expect it to work.

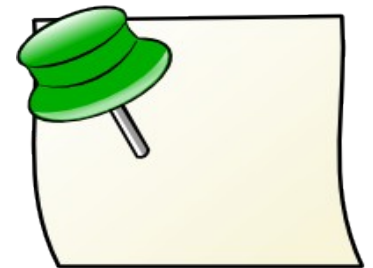
#11917 Read. Learn.
Evolve.

Codis d'error de Mark Jason Dominus



Recapitulemos!

- Configurar el editor antes de empezar
- help i dir son nuestros amigos
- Python NO es Java, no es PHP, tiene su propia manera de programarse, su propio idioma.
- En caso de duda consulta el Zen
- En lo posible sigamos las normas PEP-8
- Y tengamos siempre presentes:
 - DRY
 - KISS



Gracias!

¿Alguna pregunta?



Y mañana ...

- Empezamos con Django
- Crearemos una aplicación web
- Desde cero

django

The Web framework for perfectionists with deadlines.

Django es un framework de desarrollo de
aplicaciones web

Maduro, completo y bien documentado

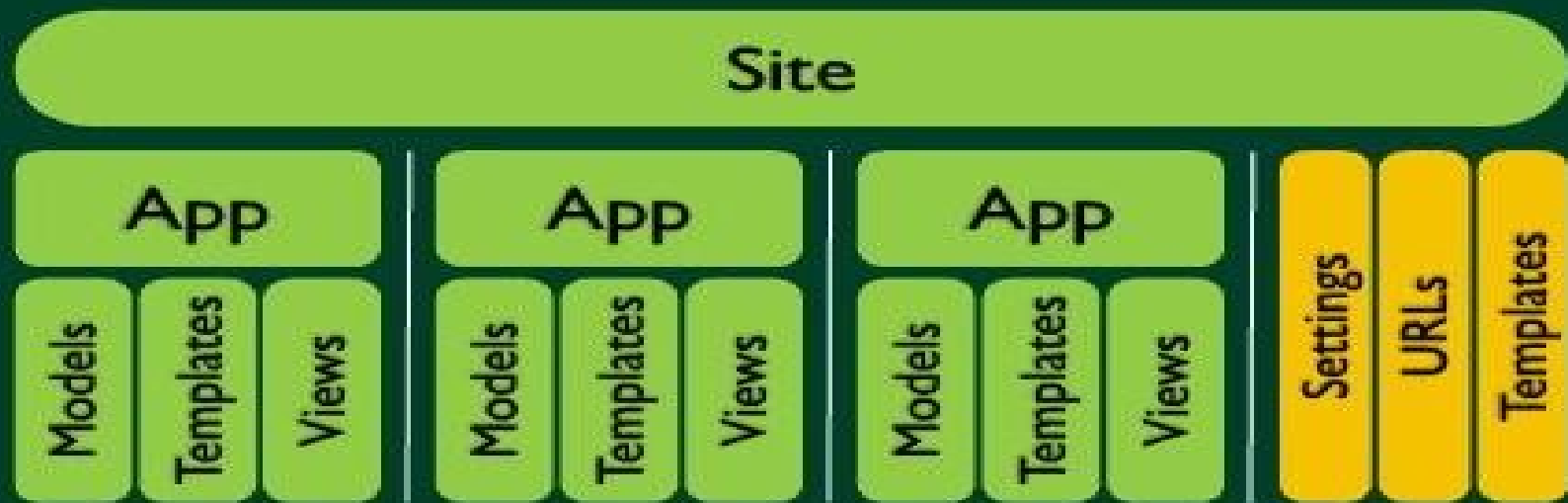
Con una gran comunidad de
desarrolladores y usuarios

Orientado a crear aplicaciones robustas
y salir online rápidamente



apsl Estructura de una aplicación Django

👉 Django App Structure



Imatge: Rob Yates

Architecture



Imagen: Rob Yates

¿Cómo empezar?

- Es importante saber Python
- Utilizar pip y virtualenv
- Y tener conocimientos sobre lo que es una aplicación web
- ... y una base de datos
- Conviene hacer el tutorial completo

¿Qué debemos saber?

- Django utiliza el patrón MVT, una adaptación del patrón MVT
- No hay "magia", pero es común seguir unas convenciones establecidas:
 - Modelo de datos: `models.py`
 - Formularios: `forms.py`
 - Controlador: `urls.py`
 - Lógica de visualización: `views.py`
 - Plantillas en templates
- Y recordar que es Python

- Define el modelo de datos

```
class Localidad(models.Model):
    """Mantiene la lista de municipios"""
    nombre = models.CharField(_("Nombre"),
                              max_length=40, unique=True)
    slug = models.SlugField(unique=True)
    url = models.URLField(blank=True, null=True)
    isla = models.ForeignKey(Isla)
    created = models.DateTimeField(editable=False, auto_now_add=True)
    updated = models.DateTimeField(editable=False, auto_now=True)

    def save(self, *args, **kwargs):
        if not self.slug:
            self.slug = slugify(self.nombre)
        super(Localidad, self).save(*args, **kwargs)

    def __unicode__(self):
        return self.nombre

    class Meta:
        verbose_name = _(u'Localidad')
        verbose_name_plural = _(u'Localidades')
        ordering = ('nombre', )
```

A partir del modelo podremos crear y acceder a la información mediante el ORM

Ej: Localidad.objects.all()

- Contiene la configuración que nos permitirá gestionar los datos desde un administrador autogenerado.

Administración de Django Bienvenido/a

Sitio administrativo

Auth	
Grupos	+ Añadir ✎ Modificar
Usuarios	+ Añadir ✎ Modificar
Authority	
Permisos	+ Añadir ✎ Modificar
Constance	
Config	✎ Modificar
Contacto	
Contactos	+ Añadir ✎ Modificar
Django_Mailer	
Blacklisted e-mail addresses	+ Añadir ✎ Modificar
Logs	+ Añadir ✎ Modificar
Messages	+ Añadir ✎ Modificar
Queued messages	+ Añadir ✎ Modificar
Flatbloks	
Flat blocks	+ Añadir ✎ Modificar

Administración de Django Bienvenido/a, **apsl**. [Cambiar contraseña](#) / [Terminar sesión](#)

[Inicio](#) > [Auth](#) > [Usuarios](#)

Escoja usuario a modificar

[Añadir usuario](#) [+](#)

Acción: seleccionados 0 de 2

<input type="checkbox"/>	Nombre de usuario	Dirección de correo electrónico	Nombre propio	Apellidos	Es staff
<input type="checkbox"/>			✓
<input type="checkbox"/>	...				✓

2 usuarios

Filtro
Por es staff
[Todo](#)
[Sí](#)
[No](#)
Por es superusuario
[Todo](#)
[Sí](#)
[No](#)
Por activo
[Todo](#)
[Sí](#)
[No](#)

- Define quién responderá a una petición http
- Está formada por expresiones regulares

```
# -*- coding: utf-8 -*-  
from django.conf.urls.defaults import *  
from contacto.views import ContactoView  
from django.views.generic import TemplateView  
  
urlpatterns = patterns('',  
    url(r'^$', ContactoView.as_view(), name='ct-form'),  
)  
  
urlpatterns += patterns('',  
    url(r'^thanks/$', TemplateView.as_view(  
        template_name='contacto/thanks.html'), name='ct-thanks'),  
)
```

- Los formularios son una de las utilidades más potentes de Django
- Validan la información
- Controlan los errores
- Permiten lógica de negocio
- Pudiendo estar asociados a modelos o no

Ejemplo

```
class EnviarAmigoForm(forms.Form):
    """Formulario para enviar la oferta a un amigo"""

    remitente = forms.CharField()
    email = forms.CharField()
    destinatario = forms.CharField()
    email_destinatario = forms.EmailField()
    comentarios = forms.CharField(widget=forms.Textarea)
    newsletter = forms.BooleanField(required = False)

    def enviar(self, request, oferta, plantilla="mail/enviar.html"):
        log.info('Enviamos a un amigo')
        if oferta.canal == 'P':
            plantilla = 'mail/texto.html'
        data = self.cleaned_data
        from_email = "%s <%s>" % (data['remitente'], data['email'])
        to = "%s <%s>" % (data['destinatario'], data['email_destinatario'])
        subject = "Mail amigo"
        html_content = render_to_string(plantilla,
            {'oferta':oferta, 'data':data},
            context_instance=RequestContext(request))
        msg = EmailMessage(subject, html_content, from_email, to = [to])
        msg.content_subtype = "html"
        msg.send()
```

```
class GuestInfoForm(forms.Form):
    """Datos de contacto del huésped"""
    nombre = forms.CharField()
    apellidos = forms.CharField()
    fecha_nacimiento = forms.DateField()
    email = forms.EmailField()
    telefono = forms.CharField()

    seleccion = forms.CharField(widget=forms.HiddenInput)
    acepto = forms.BooleanField()

    def clean_fecha_nacimiento(self):
        data = self.cleaned_data['fecha_nacimiento']
        actual_year = datetime.date.today().year
        guest_year = data.year
        dif = actual_year - guest_year
        if (dif < 0) or (dif < 16) or (dif > 120):
            raise forms.ValidationError(
                _(u'El rango de edad va de los 16 a los 120 años'))
        return data
```

```
class LocalidadForm(forms.ModelForm):
    "Formulario ligado al modelo localidad"
    class Meta:
        model = Localidad
```

- En un mundo ideal deben tener muy poca lógica, limitarse a obtener datos y enviarlos a visualización
- Son las funciones o clases referenciadas en `urls.py`
- Dos maneras:
 - Como funciones
 - Como classes (CBV)

Ejemplos

```
def ficha_evento(request, slug):  
    """Información sobre un evento concreto"""  
    evento = get_object_or_404(Evento, slug = slug)  
    return render(request, 'evento/ficha.html',  
                  {'evento': evento})
```

```
class FichaEventoView(TemplateView):  
    template_name = 'evento/ficha.html'  
  
    def get_context_data(self, **kwargs):  
        context = super(FichaEventoView, self).get_context_data(**kwargs)  
        evento = get_object_or_404(Evento, slug = kwargs['slug'])  
        context['evento'] = evento  
        return context
```

Plantillas

- Django tiene un lenguaje de plantillas propio
- Amigable con los diseñadores
- Que ha servido de base a otros lenguajes de plantillas
- Con una capacidad muy limitada por diseño de aplicar lógica de negocio
- Que permite trabajar con herencia y bloques

Ejemplo

```
<!DOCTYPE html>
<html lang="en">
<head>
  <link rel="stylesheet" href="style.css" />
  <title>{% block title %}My amazing site{% endblock %}</title>
</head>

<body>
  <div id="sidebar">
    {% block sidebar %}
    <ul>
      <li><a href="/">Home</a></li>
      <li><a href="/blog/">Blog</a></li>
    </ul>
    {% endblock %}
  </div>

  <div id="content">
    {% block content %}{% endblock %}
  </div>
</body>
</html>
```

```
{% extends "base.html" %}
```

```
{% block title %}My amazing blog{% endblock %}
```

```
{% block content %}
```

```
{% for entry in blog_entries %}
```

```
  <h2>{{ entry.title }}</h2>
```

```
  <p>{{ entry.body }}</p>
```

```
{% endfor %}
```

```
{% endblock %}
```

Y todavía hay más!

- Podemos controlar tanto la entrada como la salida a través de middlewares que se programa en Python
- Podemos pasar variables a las plantillas a través de los context processors
- Recibir mensajes de error cuando la aplicación falla de manera imprevista
- Extender el sistema de plantillas
- Utilizar diferentes cachés
- Reutilizar módulos propios o de terceros

El ecosistema Django

- <http://www.djangoproject.com>
- La lista de correo de usuario, con más de 24.000 suscritos
- PyPi
- Proyectos Github
- Proyectos Bitbucket
- Conferencias (Djangocon)
- Y una comunidad activa y amigable

- **fabric**
- django-debug-toolbar
- django-extensions
- django-reversion
- django-compressor
- django-redis
- django-cms
- django-rest-framework / Tastypie

Potencia y fiabilidad

- Tenemos el control de nuestro proyecto a todos los niveles.
- Una documentación magnífica y todo el código fuente.
- Una comunidad activa, que ha demostrado que Django escala hacia arriba y hacia abajo
- Con una filosofía de trabajo que prima la compatibilidad hacia atrás
- Con "seguridad por defecto"

¿Debo utilizar Django?

- No silver bullet
- Necesita una fase de adaptación
- Pero esta fase es mucho más corta que con cualquier otro framework
- Los proyectos se vuelven mucho más mantenibles
- Con un coste de desarrollo significativamente menor
- Con unos desarrolladores más contentos

HANDS ON!