

# Python

## Introducción

Python es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible.

### Se trata de un lenguaje:

- ✓ Interpretado o de script.
- ✓ Tipado dinámico.
- ✓ Fuertemente tipado.
- ✓ Multiplataforma.
- ✓ Orientado a objetos.

### Lenguaje interpretado o de script

Un lenguaje interpretado o de script es aquel que se ejecuta utilizando un programa intermedio llamado intérprete, en lugar de compilar el código a lenguaje máquina que pueda comprender y ejecutar directamente una computadora (lenguajes compilados). La ventaja de los lenguajes compilados es que su ejecución es más rápida. Sin embargo los lenguajes interpretados son más flexibles y más portables.

Python tiene, no obstante, muchas de las características de los lenguajes compilados, por lo que se podría decir que es semi interpretado. En Python, como en Java y muchos otros lenguajes, el código fuente se traduce a un pseudo código máquina intermedio llamado bytecode la primera vez que se ejecuta, generando archivos .pyc o .pyo (bytecode optimizado), que son los que se ejecutarán en sucesivas ocasiones.

### Tipado dinámico

La característica de tipado dinámico se refiere a que no es necesario declarar el tipo de dato que va a contener una determinada variable sino que su tipo se determinará en tiempo de ejecución según el tipo del valor al que se asigne, y el tipo de esta variable puede cambiar si se le asigna un valor de otro tipo.

### Fuertemente tipado

No se permite tratar a una variable como si fuera de un tipo distinto al que tiene, es necesario convertir de forma explícita dicha variable al nuevo tipo previamente. Por ejemplo, si tenemos una variable que contiene un texto (variable de tipo cadena o string) no podremos tratarla como un número (sumar la cadena “9” y el número 8). En otros lenguajes el tipo de la variable cambiaría para adaptarse al comportamiento esperado, aunque esto es más propenso a errores.

### Multiplataforma

El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios.

## Orientado a objetos

La orientación a objetos es un paradigma de programación en el que los conceptos del mundo real relevantes para nuestro problema se trasladan a clases y objetos en nuestro programa. La ejecución del programa consiste en una serie de interacciones entre los objetos. Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.

## Casos de éxito de python

Algunos casos de éxito en el uso de Python son Google, Yahoo, la NASA, Industrias Light & Magic, y todas las distribuciones Linux, en las que Python cada vez representa un tanto por ciento mayor de los programas disponibles.

## Instalación de Python

Existen varias implementaciones distintas de Python: CPython, Jython, IronPython, PyPy, etc. CPython es la más utilizada, la más rápida y la más madura. Cuando la gente habla de Python normalmente se refiere a esta implementación. En este caso tanto el intérprete como los módulos están escritos en C.

Jython es la implementación en Java de Python, mientras que IronPython es su contrapartida en C# (.NET). Su interés estriba en que utilizando estas implementaciones se pueden utilizar todas las librerías disponibles para los programadores de Java y .NET.

PyPy, por último, como habréis adivinado por el nombre, se trata de una implementación en Python de Python.

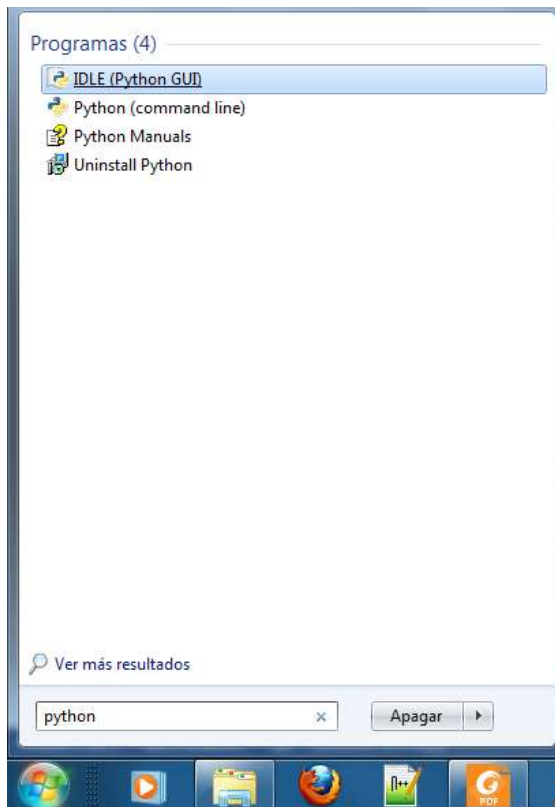
CPython está instalado por defecto en la mayor parte de las distribuciones Linux y en las últimas versiones de Mac OS. Para comprobar si está instalado abre una terminal y escribe `python`. Si está instalado se iniciará la consola interactiva de Python y obtendremos parecido a lo siguiente:

```
Python 2.5.1 (r251:54863, May 2 2007, 16:56:35)
[GCC 4.1.2 (Ubuntu 4.1.2-0ubuntu4)] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

## Primer programa en python

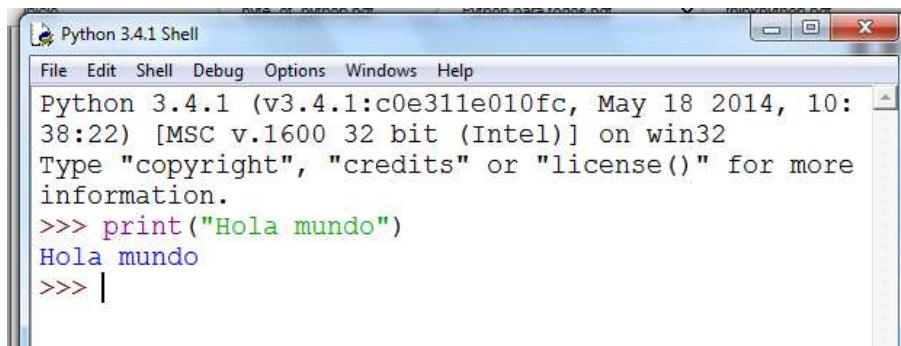
### En este caso usaremos Windows

- 1) ingresa en el buscador de inicio de windows python y selecciona IDLE(Python GUI), para que se abra la consola de comandos de python

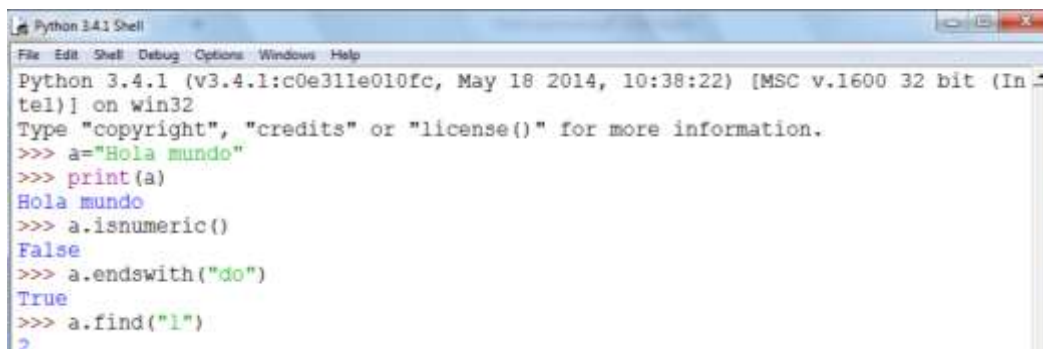


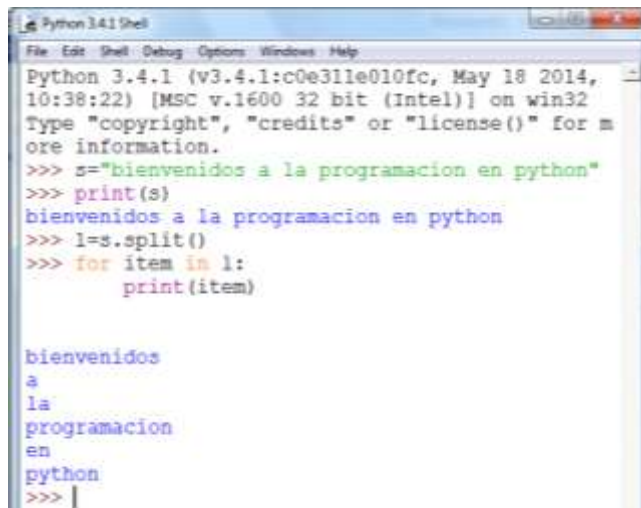
2) digita

`print("Hola mundo")`



## Digita las siguientes líneas de código



A screenshot of a Python 3.4.1 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Windows', and 'Help'. The main text area shows the following code and output:

```
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> s="bienvenidos a la programacion en python"
>>> print(s)
bienvenidos a la programacion en python
>>> l=s.split()
>>> for item in l:
>>>     print(item)

bienvenidos
a
la
programacion
en
python
>>> |
```

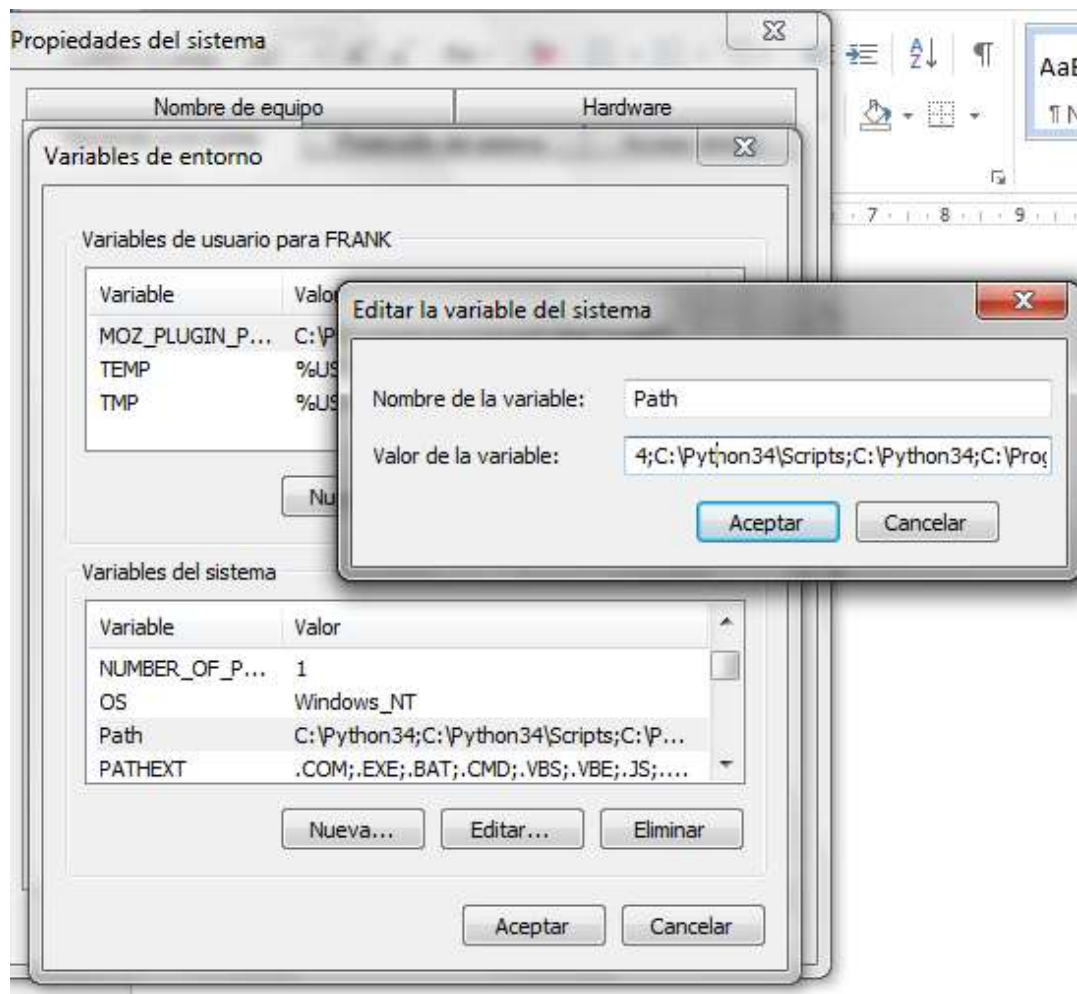
Instalar python en Windows

Python setup.py install

Comprobar que Django esta instalado,

- 1) Abrir la consola de python
- 2) Import Django
- 3) Django.get\_version()

Agregar django a las variables del entorno



Crear un proyecto

```
C:\Windows\system32\cmd.exe
>>> import Django
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named 'Django'
>>> import django
>>> django.get_version()
'1.7.4'
>>> ^Z

C:\Users\FRANK>python
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import django
>>> django.get_version()
'1.7.4'
>>> ^Z

C:\Users\FRANK>cd desktop
C:\Users\FRANK\Desktop>mkdir pythonProject
C:\Users\FRANK\Desktop>cd pythonProject
C:\Users\FRANK\Desktop\pythonProject>django-admin startproject myfirstproject
C:\Users\FRANK\Desktop\pythonProject>cd myfirstproject
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>dir
El volumen de la unidad C no tiene etiqueta.
El número de serie del volumen es: 8491-BEDE

Directorio de C:\Users\FRANK\Desktop\pythonProject\myfirstproject
16/04/2016  22:00    <DIR>          .
16/04/2016  22:00    <DIR>          ..
16/04/2016  22:00                257  manage.py
16/04/2016  22:00    <DIR>          myfirstproject
                        1 archivos          257 bytes
                        3 dirs  10.822.828.032 bytes libres
```

¿Qué es el wsgi.py?

Web Server Gateway Interface, se utiliza para poner en producción o desplegar la aplicación en un servidor

Correr la aplicación

Manage.py runserver

```
[sessions]
  clearsessions

[staticfiles]
  collectstatic
  findstatic
  runserver

C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until they are applied.
Run 'python manage.py migrate' to apply them.
April 16, 2016 - 22:17:08
Django version 1.7.4, using settings 'myfirstproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Para detener el servidor CTRL+C

Interfaz de Administracion

manage.py makemigration

manage.py migrate

comprobar

<http://127.0.0.1:8000/admin>

crear usuario

manage.py createsuperuser

```
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py createsuperuser
Username (leave blank to use 'frank'): admin
Email address: micorreo@yahoo.es
Password:
Password (again):
Superuser created successfully.
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>
```

Crear la primera app

django-admin startapp preguntasyrespuestas

```
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>django-admin startapp preguntasyrespuestas
```

ingresar a settings.py y registrar la aplicación

```
29
30     # Application definition
31
32     INSTALLED_APPS = (
33         'django.contrib.admin',
34         'django.contrib.auth',
35         'django.contrib.contenttypes',
36         'django.contrib.sessions',
37         'django.contrib.messages',
38         'django.contrib.staticfiles',
39         'preguntasyrespuestas',
40     )
```

Crear los modelos

```
from django.db import models

# Create your models here.

class Pregunta(models.Model):
    #Es obligatorio especificarle el parametro max_length
    asunto= models.CharField(max_length=200)
    descripcion = models.TextField()
    fecha_publicacion = models.DateTimeField(auto_now_add=True)

class Respuesta(models.Model):
    #Definir la relaciones
    Pregunta = models.ForeignKey(Pregunta)
    contenido = models.TextField()
    mejor_respuesta = models.BooleanField("Respuesta preferida", default=False)
```



Mapear la base de datos

```
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py makemigrations
Migrations for 'preguntasypuestas':
  0001_initial.py:
    - Create model Pregunta
    - Create model Respuesta
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py validate
System check identified no issues (0 silenced).
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py migrate
Operations to perform:
  Apply all migrations: contenttypes, admin, sessions, auth, preguntasypuestas
Running migrations:
  Applying preguntasypuestas.0001_initial... OK
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>_
```

Usando la api database de django

Manage.py shell

From preguntasypuestas import Pregunta,Respuesta

```
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py shell
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from preguntasypuestas.models import Pregunta,Respuesta
>>>
```

Probamos en la base de datos no existe ningún objeto con:

NombreClase.objects.all(), Que nos devuelve una lista con todos los objetos de dicha clase

```
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py shell
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from preguntasypuestas.models import Pregunta,Respuesta
>>> Pregunta.objects.all()
[]
>>> Respuesta.objects.all()
[]
>>> _
```

Para crear una pregunta de nuestro modelo y aceptar valores por defecto se debe importar la biblioteca timezone, sino esperara que se e ingrese un valor en la fecha.

From django.utils import timezone

```
[]
>>> from django.utils import timezone
>>> p=Pregunta(asunto="Quien es tu escritor favorito",descripcion="",fecha_publicacion=timezone.now())
>>> _
```

Comprobamos nuevamente que el objeto ha sido persistido en el modelo



Para persistirlo sobre el objeto instanciando se debe llamar la método save() de la siguiente forma:

nombreObjeto.save()

```
>>> p.save()
>>> Pregunta.objects.all()
[<Pregunta: Pregunta object>]
>>> _
```

Por ultimo podemos acceder a los valores del objeto

```
>>> p.id
1
>>> p.asunto
'Quié es tu escritor favorito'
>>> p.descripcion
''
>>> p.fecha_publicacion
datetime.datetime(2016, 4, 17, 6, 53, 36, 446200, tzinfo=<UTC>)
>>> _
```

Si fijamos en la llamada del método que nos devuelve la lista de objetos, se muestra como el objeto ya existe en la base de datos, sin embargo la instancia únicamente dice object y así aparecerán los demás, si quisiéramos cambiar, para que lo mostrara con algún nombre que lo identifique, entonces debemos de cambiar el archivo models.py agregándole el método `__str__(self)` de la siguiente manera

```
>>> p.save()
>>> Pregunta.objects.all()
[<Pregunta: Pregunta object>]
>>> _
```

```

from django.db import models

# Create your models here.

class Pregunta(models.Model):
    #Es obligatorio especificarle el parametro max_length
    asunto= models.CharField(max_length=200)
    descripcion = models.TextField()
    fecha_publicacion = models.DateTimeField(auto_now_add=True)

    #especificando la representacion de identificacion
    #de cada objeto
    def __str__(self):
        return self.asunto

class Respuesta(models.Model):
    #Definir la relaciones
    Pregunta = models.ForeignKey(Pregunta)
    contenido = models.TextField()
    mejor_respuesta = models.BooleanField("Respuesta preferida", default=False)

    #identificando el objeto
    def __str__(self):
        return self.contenido

```

Probamos nuevamente y veremos como ha cambiado la identificación del objeto de object al contenido de la pregunta

```

C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py shell
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from preguntasyrespuestas.models import Pregunta,Respuesta
>>> Pregunta.objects.all()
[<Pregunta: Quies es tu escritor favorito>]
>>>

```

Agregamos un nuevo método para comprobar que si se ha agregado ahora la fecha de publicación, para ello importamos **from django.utils import timezone**

```

from django.db import models
from django.utils import timezone

# Create your models here.

class Pregunta(models.Model):
    #Es obligatorio especificarle el parametro max_length
    asunto= models.CharField(max_length=200)
    descripcion = models.TextField()
    fecha_publicacion = models.DateTimeField(auto_now_add=True)

    #especificando la representacion de identificacion
    #de cada objeto
    def __str__(self):
        return self.asunto

    #agregamos un metodo para comprobar si se ha publicado hoy
    def publicadoHoy(self):
        return self.fecha_publicacion.date()==timezone.now().date()

```

Reiniciamos la consola para que muestre los cambios y probamos y vemos que nos devolverá **True** si publicamos ahora la pregunta o **False** en caso contrario

```
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py shell
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
(InteractiveConsole)
>>> from preguntasyrespuestas.models import Pregunta, Respuesta
>>> from django.utils import timezone
>>> p=Pregunta.objects.get(pk=1)
>>> p.id
1
>>> p.asunto
'Quié es tu escritor favorito'
>>> p.descripcion
''
>>> p.fecha_publicacion
datetime.datetime(2016, 4, 17, 6, 53, 36, 446200, tzinfo=<UTC>)
>>> p.publicadoHoy()
True
>>>
```

Ahora veremos como crear respuestas

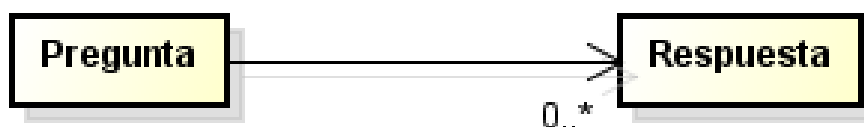
En el modelo tenemos

```
from django.db import models

# Create your models here.

class Pregunta(models.Model):
    #Es obligatorio especificarle el parametro max_length
    asunto= models.CharField(max_length=200)
    descripcion = models.TextField()
    fecha_publicacion = models.DateTimeField(auto_now_add=True)

class Respuesta(models.Model):
    #Definir la relaciones
    Pregunta = models.ForeignKey(Pregunta)
    contenido = models.TextField()
    mejor_respuesta = models.BooleanField("Respuesta preferida", default=False)
```



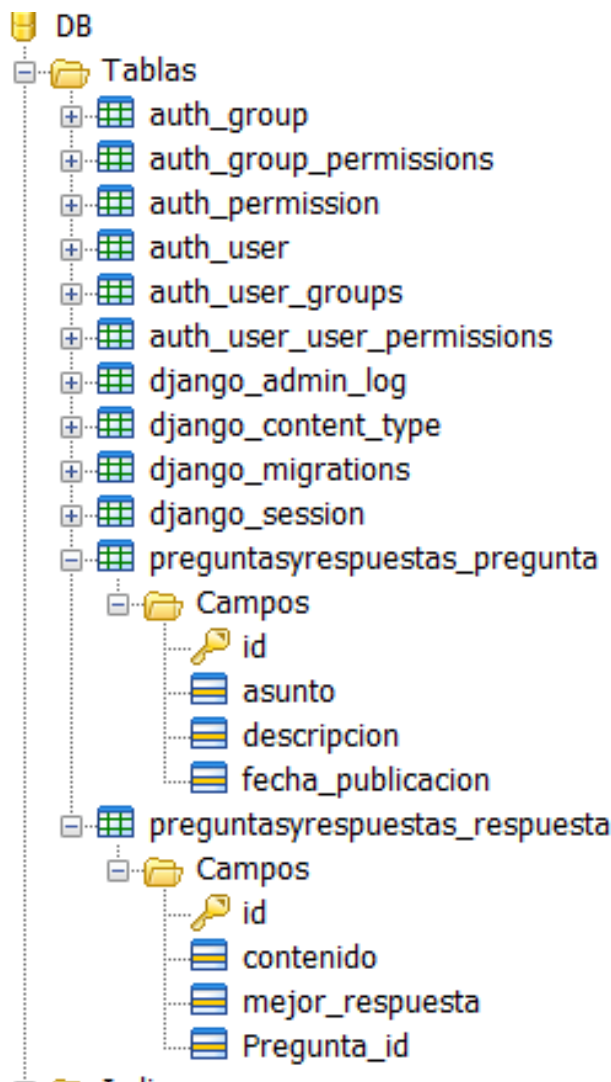
Este modelo quiere decir que una Pregunta puede tener varias respuestas

En la base de datos tenemos

**preguntasyrespuestas\_pregunta(id,asunto,descripción,fecha\_publicacion)**

**preguntasyrespuestas\_respuesta(id,contenido,mejor\_respuesta,Pregunta\_id)**

Esta es la forma que django mapea la base de datos, nombre de paquete\_nombre de clase todo en miniscula. Observa que la primary key lo crea django automáticamente si no se lo especificamos y que se ha creado un foreign key que se lo hemos especificado explícitamente. Si usamos un gestor visual de sqlite podemos ver lo siguiente



Para crear respuesta la sintaxis es la siguiente:

`nombreObjeto.nombreclase_set.create(atributo1=valor,...,atributon=valorn)`

en este caso **`p.respuesta_set.create(contenido="valor")`** y luego guardamos con **`p.save()`** y comprobamos

```
>>> p.respuesta_set.create(contenido="Mi preferido es Borges")
<Respuesta: Mi preferido es Borges>
>>> p.respuesta_set.create(contenido="Mi preferido es Garcia")
<Respuesta: Mi preferido es Garcia>
>>> p.respuesta_set.create(contenido="Mi preferido es Dalton")
<Respuesta: Mi preferido es Dalton>
>>> p.save()
>>> Pregunta.objects.all()
[<Pregunta: Quies es tu escritor favorito>]
>>> Respuesta.objects.all()
[<Respuesta: Mi preferido es Borges>, <Respuesta: Mi preferido es Garcia>, <Respuesta: Mi preferido es Dalton>]
>>>
```

Si lo vemos en un gestor visual veriamos algo como lo siguiente

| id | contenido              | mejor_respuesta          | Pregunta_id |
|----|------------------------|--------------------------|-------------|
| 1  | Mi preferido es Borges | <input type="checkbox"/> | 1           |
| 2  | Mi preferido es Garcia | <input type="checkbox"/> | 1           |
| 3  | Mi preferido es Dalton | <input type="checkbox"/> | 1           |

observamos como es que automaticamente se persisten los datos en la base de datos y se asocian los registros, se ve claramente la relación de 1 a muchos donde una pregunta puede tener de 0 a varias respuestas.

### Aministracion condjango

Asta ahora hemos visto, como la persistencia de datos desde la consola, ahora veremos como hacerlo desde el administrador que nos ofrece django

crea un super usuario, esto esta explicado en las primeras partes de este tutorial

```
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py createsuperuser
Username (leave blank to use 'frank'): admin
Email address: micorreo@yahoo.es
Password:
Password (again):
Superuser created successfully.
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>
```

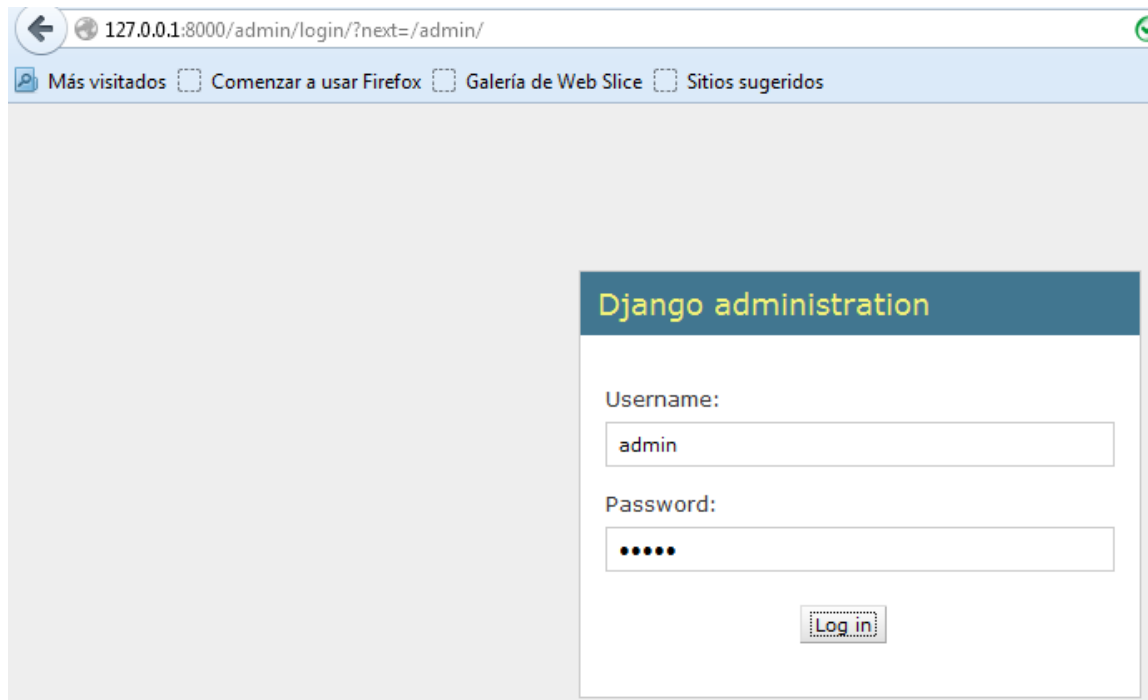
Luego corre la aplicación

```
C:\Users\FRANK\Desktop\pythonProject\myfirstproject>manage.py runserver
Performing system checks...
System check identified no issues (0 silenced).
April 17, 2016 - 10:23:59
Django version 1.7.4, using settings 'myfirstproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

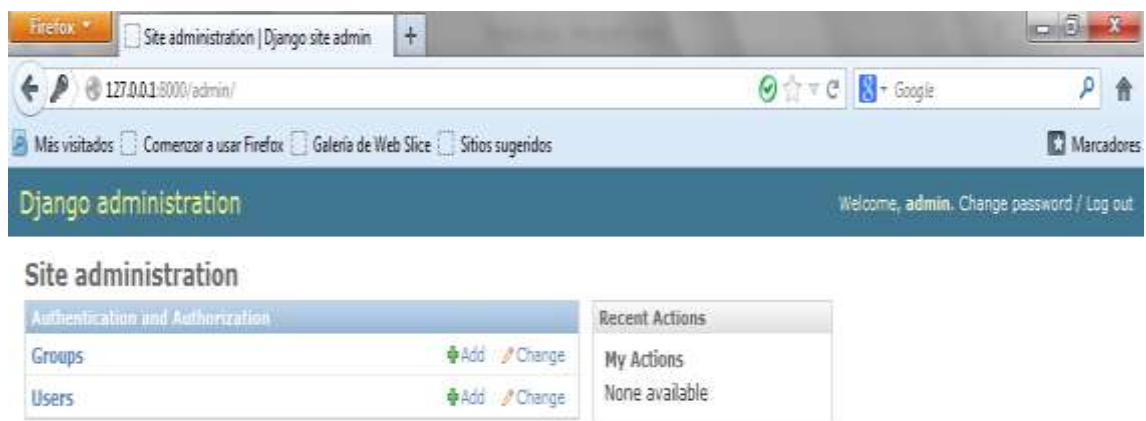
Ingresa a la siguiente dirección

<http://127.0.0.1:8000/admin>

Te aparecerá una ventana de login, ingresa tu usuario y contraseña presiona Log In



Te aparecerá la ventana de administración

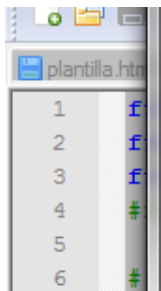


Lo que haremos es indicarle a django que nos incluya los modelos en esta ventana de administración y poder realizar el CRUD desde aca si necesidad de usar la consola, para ello haremos lo siguiente:

Abrimos el archivo models.py e importamos

```
from django.contrib import admin
```

luego al final registramos los modelos con **admin.site.register(Modelo)**



```
from django.db import models
from django.utils import timezone
from django.contrib import admin

# Create your models here.

class Pregunta(models.Model):
```

```
from django.contrib import admin

# Create your models here.

class Pregunta(models.Model):
    #Es obligatorio especificarle el parametro max_length
    asunto= models.CharField(max_length=200)
    descripcion = models.TextField()
    fecha_publicacion = models.DateTimeField(auto_now_add=True)

    #especificando la representacion de identificacion
    #de cada objeto
    def __str__(self):
        return self.asunto

    #agregamos un metodo para comprobar si se ha publicado hoy
    def publicadoHoy(self):
        return self.fecha_publicacion.date()==timezone.now().date()

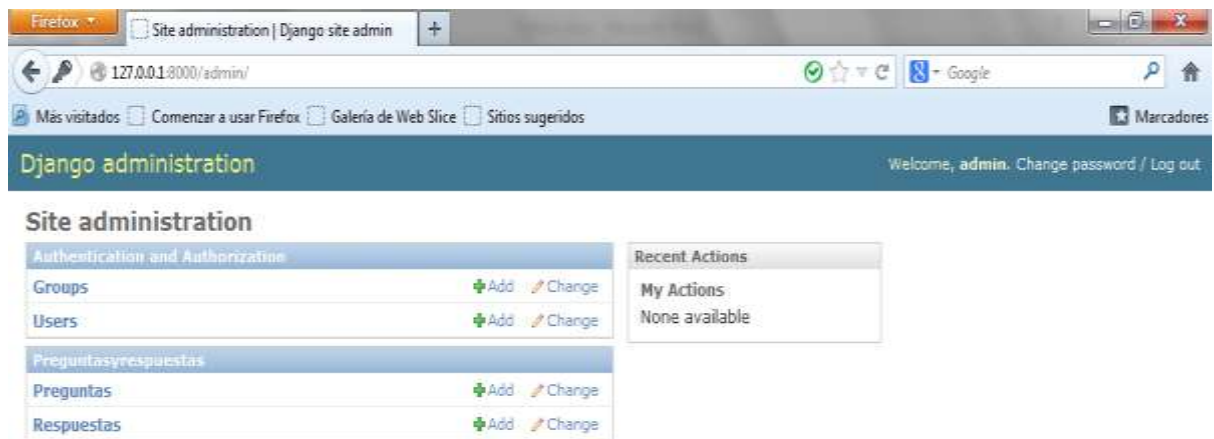
class Respuesta(models.Model):
    #Definir la relaciones
    Pregunta = models.ForeignKey(Pregunta)
    contenido = models.TextField()
    mejor_respuesta = models.BooleanField("Respuesta preferida", default=False)

    #identificando el objeto
    def __str__(self):
        return self.contenido

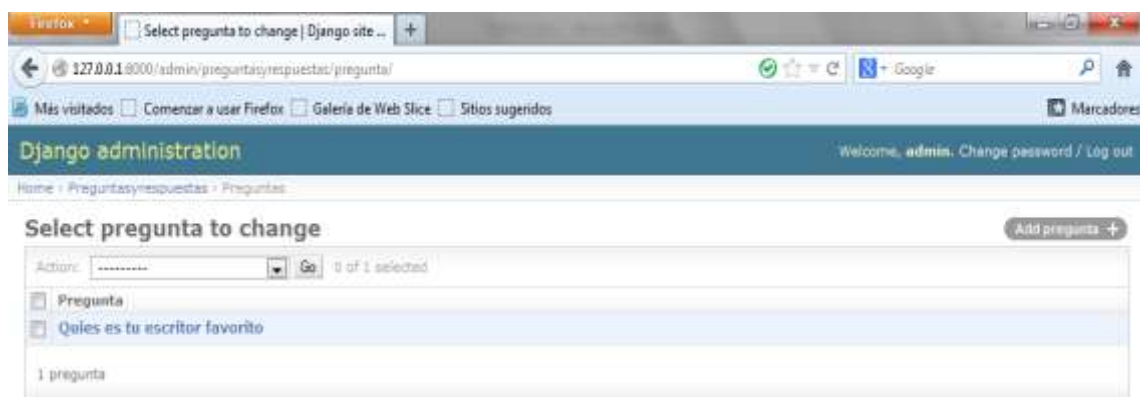
admin.site.register(Pregunta)
admin.site.register(Respuesta)
```

Guarda los cambios en el archivo y actualiza la ventana de administración, ahora veras como se ha incluido el paquete de **preguntas y respuestas** con los modelos que le hemos indicado **Pregunta y Respuesta**





Da click en Change de Preguntas y veras como te aparece una ventana con la pregunta realizada anteriormente, donde puedes agregar otra pregunta, eliminarla y editarla si presionas en el link del asunto de la pregunta.



Si ingresar al link de Respuestas veras como también existen las respuestas que guardamos en los pasos anteriores.

## Preguntas y respuestas administration

| Preguntas y respuestas |  |
|------------------------|--|
| Preguntas              | <a href="#">+ Add</a> <a href="#">Change</a> |
| Respuestas             | <a href="#">+ Add</a> <a href="#">Change</a> |



Si agregamos una respuesta veras claramente como están relacionados los objetos de Pregunta con esto mediante un select(lista desplegable), también podemos crear una pregunta si no existe al presionar el icono de +(color verde)

The screenshot shows a web browser window with the Django administration interface. The page title is 'Add respuesta | Django site admin'. The URL bar shows '127.0.0.1:8000/admin/preguntas/respuestas/respuesta/add/'. The page header includes 'Django administration' and a welcome message for 'admin'. The breadcrumb trail is 'Home > Preguntas y respuestas > Respuestas > Add respuesta'. The main form is titled 'Add respuesta' and contains two fields: 'Pregunta:' and 'Contenido:'. The 'Pregunta:' field is a dropdown menu with a green '+' icon to its right, indicating a 'create new' option. The dropdown menu is open, showing a list of questions, with '¿Quié es tu escritor favorito?' selected. The 'Contenido:' field is a large text area. At the bottom of the form, there is a checkbox labeled 'Respuesta preferida'. The bottom of the page features three buttons: 'Save and add another', 'Save and continue editing', and 'Save'.

Firefox Add respuesta | Django site admin

127.0.0.1:8000/admin/preguntas/respuestas/respuesta/add/

Más visitados Comenzar a usar Firefox Galería de Web Slice Sitios sugeridos Marcadores

Django administration Welcome, admin. Change password / Log out

Home > Preguntas y respuestas > Respuestas > Add respuesta

### Add respuesta

Pregunta:  +

Contenido:

☐ Respuesta preferida

Save and add another Save and continue editing Save