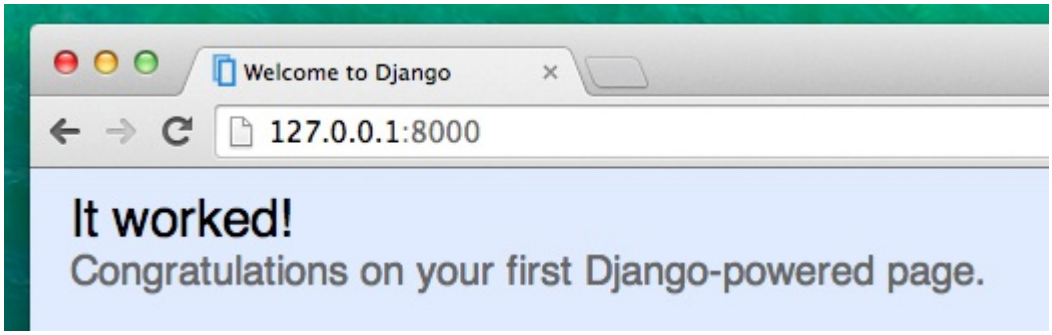
Práctica 9.1: Django urls

Vamos a construir nuestra primera página web – una página de inicio para nuestro blog. Pero primero, vamos a aprender un poco sobre URLs en Django.

¿Qué es una URL?

Una URL en este caso particular es simplemente una dirección web, puedes ver una URL cada vez que visitas cualquier sitio web - es visible en la barra de direcciones de tu navegador (Sí, `http://127.0.0.1:8000` es una URL. Y `http://uabc.mx` es también una URL).



Cada página en Internet necesita su propia URL. De esta manera tu aplicación sabe lo que debe mostrar a un usuario que abre una URL. En Django se usa algo llamado `URLconf` (configuración de URL), un conjunto de patrones que Django intentará hacer coincidir con la dirección URL recibida para encontrar la vista correcta.

¿Cómo funcionan las URLs en Django?

Vamos a abrir el archivo `mysite/urls.py` y ver cómo es:

```
"""mysite URL Configuration

[...]
"""
from django.conf.urls import url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
]
```

Como puedes ver, Django ya puso algo aquí para nosotros.

Las líneas que comienzan con triples comillas (`'''` o `"""`) se les llama *docstrings* – puedes escribirlas al inicio de un script, clase o método para describir que hace. Estas líneas no se ejecutaran en Python.

Ya está aquí la URL de admin, que se visitó en una práctica anterior:

```
url(r'^admin/', admin.site.urls),
```

Esto significa que para cada URL que empieza con `admin/` Django encontrará su correspondiente *view*. En este caso estamos incluyendo en una sola línea muchas URLs de admin, así no está todo empaquetado en este pequeño archivo - es más limpio y legible.

Regex

¿Te preguntas cómo Django hace juego de las direcciones URL con las vistas? Bueno, esta parte es difícil. Django utiliza `regex` – también llamadas expresiones regulares. Regex tiene muchas (muchas!) normas que forman un patrón de búsqueda. Dado que las expresiones regulares son un tema avanzado, no entraremos en detalles sobre su funcionamiento.

Si te interesa entender cómo creamos esos patrones, aquí hay un ejemplo del proceso – solamente necesitaremos un limitado subconjunto de reglas para expresar el patrón que estamos buscando:

- `^` denota el principio del texto
- `$` denota el final del texto
- `\d` representa un dígito
- `+` indica que el ítem anterior debería ser repetido por lo menos una vez
- `()` para encerrar una parte del patrón

Cualquier otra cosa en la definición del URL será tomada literalmente.

Ahora imagina que tienes un sitio web con una dirección como `http://www.mysite.com/post/12345/`, donde `12345` es el número de post.

Escribir vistas separadas para todos los números de post sería realmente molesto. Con las expresiones regulares podemos crear un patrón que coincidirá la URL y extraerá el número para nosotros: `^post/(\d+)/$`. Analicemos esta expresión parte por parte para entender qué es lo que estamos haciendo aquí:

- **^post/** le está diciendo a Django que tome cualquier cosa que tenga `post/` al principio del URL (justo antes de `^`)
- **(\d+)** significa que habrá un número (de uno o más dígitos) y que queremos que ese número sea capturado y extraído
- **/** le dice a Django que otro carácter `/` debería venir a continuación
- **$** indica el final del URL, lo que significa que sólo cadenas finalizando con `/` harán juego con este patrón

Tu primer URL de Django

Es hora de crear nuestro primer URL. Queremos que 'http://127.0.0.1:8000/' sea la página de inicio de nuestro blog y que muestre una lista de posts.

También queremos mantener el archivo `mysite/urls.py` limpio, así que importaremos URLs de nuestro `blog` al archivo principal `mysite/urls.py`.

Agrega una línea que importará `blog.urls`. Nota que estamos usamos la función aquí `include` **así que deberás** agregar agregar eso a la primera linea de import en el script.

Tu archivo `mysite/urls.py` debería verse algo así:

```
from django.conf.urls import include, url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'', include('blog.urls')),
]
```

Django ahora redirigirá todo lo que vaya hacia 'http://127.0.0.1:8000/' a `blog.urls` y buscará por más instrucciones allí.

Cuando escribes expresiones regulares en Python acostúmbrate a poner `r` al principio de la cadena de caracteres. Esto es una indicación para que Python entienda que la cadena contendrá caracteres especiales que no son para ser interpretados por Python, sino que son parte de la expresión regular.

blog.urls

Crea un nuevo archivo vacío `blog/urls.py`. Agrega estas primeras dos líneas:

```
from django.conf.urls import url
from . import views
```

Aquí solo estamos importando la función `url` de Django y todas nuestras `views` de la aplicación `blog` (todavía no tenemos ninguna, pero lo haremos en un momento).

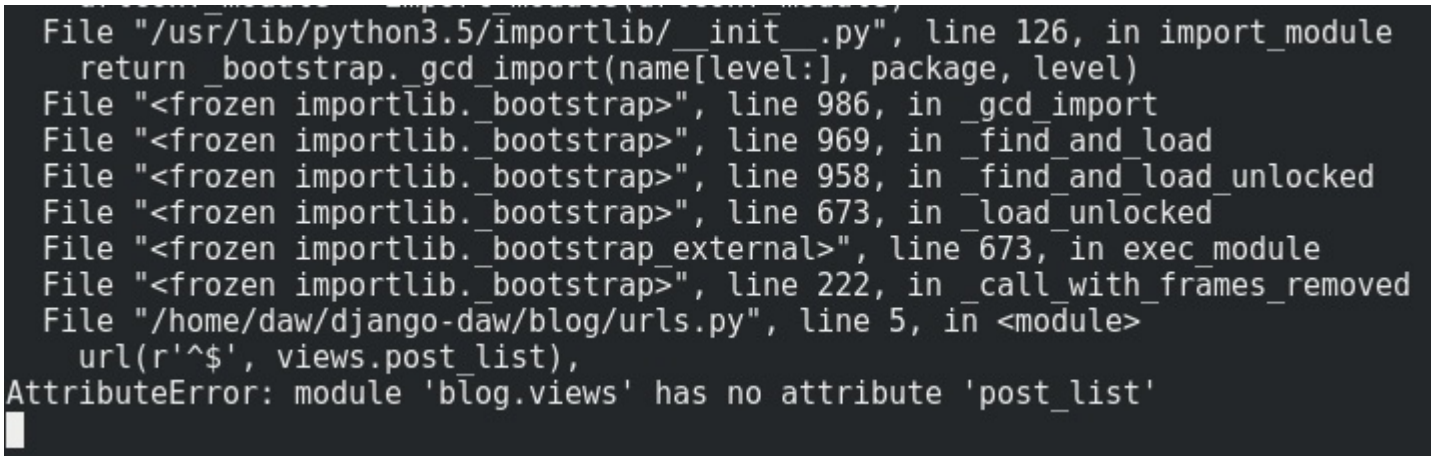
Luego de esto, podemos agregar nuestro primer patrón URL:

```
urlpatterns = [
    url(r'^$', views.post_list, name='post_list'),
]
```

Como puedes ver, ahora estamos asignando una `view` llamada `post_list` al URL `^$`. Esta expresión regular hará juego con `^` (un inicio) seguido de `$` (un final) - por lo tanto, sólo una cadena vacía coincidirá. Y esto es correcto, ya que en los URL resolvers de Django 'http://127.0.0.1:8000/' no es parte del URL. Este patrón mostrará a Django qué `views.post_list` es el lugar correcto al que ir si alguien ingresa a tu sitio web con la dirección 'http://127.0.0.1:8000/'.

La ultima parte, `name='post_list'`, es el nombre de la URL que se usara para identificar la vista. Este puede ser igual al nombre de la vista o puede ser algo completamente diferente. Nosotros estaremos usando los nombres de las URLs más adelante en el proyecto, así que es importante nombrar cada URL en la aplicación. También debemos de mantener los nombres de las URLs únicos y fáciles de recordar.

Si intentas visitar http://127.0.0.1:8000/ en este momento, encontrarás un mensaje del tipo 'web page not available'. Esto es porque el servidor (recordaste ejecutar `runserver` ?) no esta corriendo. Hecha un vistazo a la ventana de tu servidor para ver el porque.



Tu consola muestra un error, pero no es para preocuparse - es de hecho muy util. En este caso esta diciendo que **no attribute 'post_list'**. Ese es el nombre de la *vista* que Django está tratando de encontrar y usar, pero no la hemos creado todavia. En este punto tu `/admin/` tampoco funcionará. No pasa nada, esto lo solucionaremos en la siguiente parte.

> Si deseas saber mas sobre URLconfs en Django, mira la documentación oficial URLs.

Práctica 9.2 - Vistas de Django

Es hora de deshacerse del error que hemos creado en la parte anterior.

Un *view* es un lugar donde ponemos la "lógica" de nuestra aplicación. Se solicitará información del `model` que creaste anteriormente y se pasará a un `template` que crearás en la próxima práctica. Las vistas son sólo funciones de Python que son un poco más complicados que lo que hicimos en la práctica Introducción a Python.

Las vistas se colocan en el archivo `views.py`. Agregaremos nuestras vistas al archivo `blog/views.py`.

blog/views.py

Bien, vamos abrir este archivo y ver lo que contiene:

```
from django.shortcuts import render

## Create your views here.
```

No demasiadas cosas aquí todavía.

Recuerda que las líneas que comienzan con `#` son comentarios - esto significa que esas líneas no serán ejecutadas por Python.

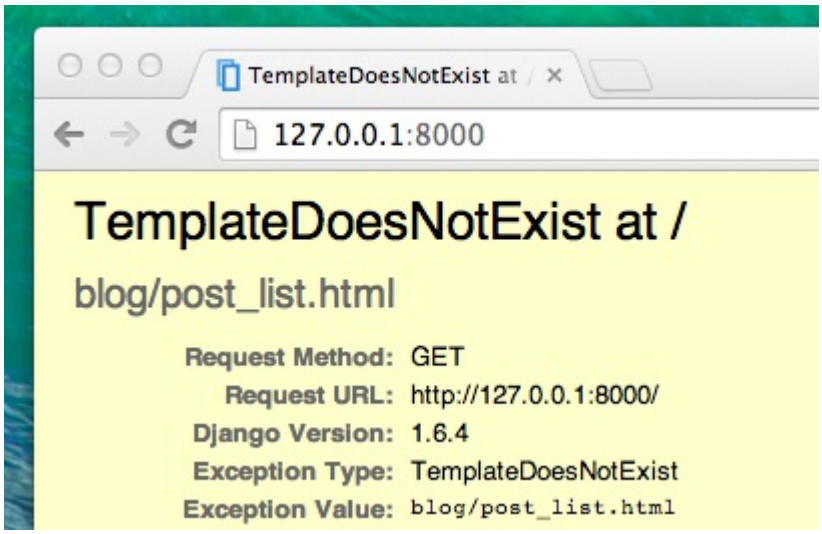
La vista más simple puede ser como esto:

```
def post_list(request):
    return render(request, 'blog/post_list.html', {})
```

Como puedes ver, hemos creado un método (`def`) llamado `post_list` que toma un `request` y hace un `return` de un método `render` que dibujará nuestra plantilla `blog/post_list.html`.

Guarda el archivo, dirígete a http://127.0.0.1:8000/ y veamos lo que tenemos ahora.

¡Otro error! Leamos lo que está pasando ahora:



Esto indica que el servidor esta corriendo de nuevo, al menos, pero todavía no se ve bien. Esta es una pagina de error creada por el servidor, nada de que asustarnos - justo como los errores en la consola estos de hecho nos pueden ser muy útiles. Puedes leer que el `TemplateDoesNotExist`. Corrijamos este error y creemos un *template* en la siguiente práctica.

> Aprende más acerca de las views de Django leyendo la documentación oficial Views