# Crea tu propio Blog

# Programar no es tan difícil como aparenta y queremos mostrarte cuán divertido puede llegar a ser.

https://tutorial.djangogirls.org/es/

# Cómo Funciona Internet.

- Internet es una gran telaraña que interconecta muchos computadores en el mundo.
- Internet se conforma de muchas páginas web creadas en algún lenguaje de programación y subidas a  un dominio para hacerlas públicas mundialmente.
- Para ingresar a alguna de las tantas páginas que tiene internet debes saber la direccion de esta como por ejemplo www.google.com.
- Hoy en dia en internet puedes crear páginas que son tiendas virtuales , blog , mapas , domicilios online y muchas cosas más...
- **https://tutorial.djangogirls.org/es/how_the_internet_works/**
-

# Línea de Comandos

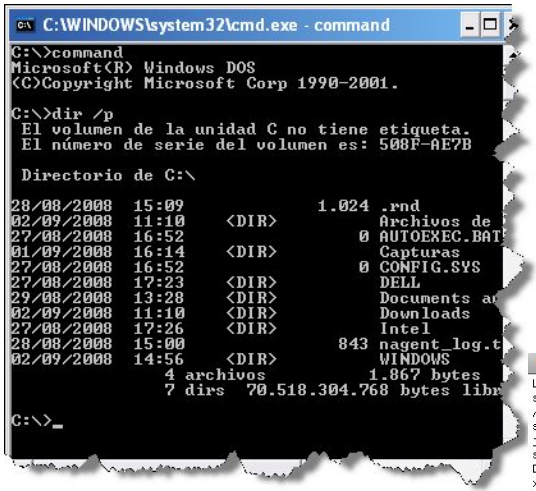
La línea de comandos es una aplicación que todos los computadores poseen precisamente para manipular tu computador por medio de texto como por ejemplo , mover archivos , crear archivos , crear carpetas etc.

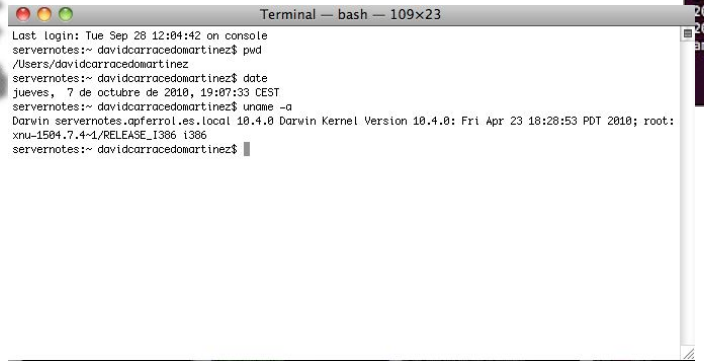
Como Ingresar a la Línea de comandos?

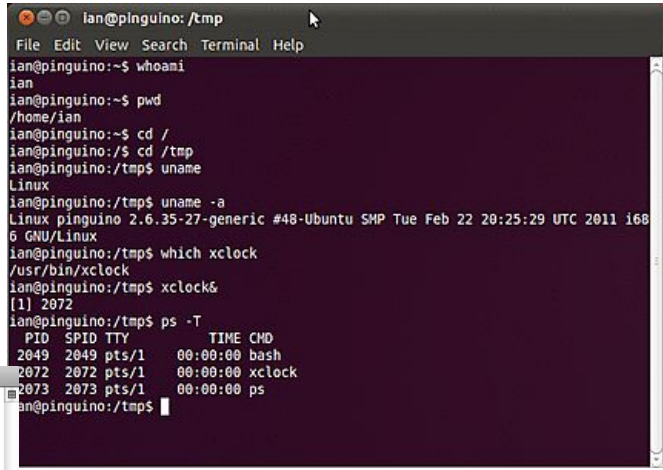
**Windows**
Ir al menú Inicio ➜ Todos los programas ➜ Accesorios ➜ Command Prompt
**Mac OS X**
Aplicaciones ➜ Servicios ➜ Terminal
**Linux**
Está probablemente en Aplicaciones ➜ Accesorios ➜ Terminal

**https://tutorial.djangogirls.org/es/intro_to_command_line/**

# Como debe verse la Línea de Comandos



# Instalación de Python

Python es un lenguaje de programación muy popular que puede utilizarse para la creación de sitios web, juegos, software académico, gráficos y mucho, mucho más.

https://tutorial.djangogirls.org/es/python_installation/

## Windows

Puedes descargar Python para Windows desde el sitio web,  la version a utilizar para este taller es 3.5 **https://www.python.org/downloads/** . Después de descargar el archivo *.msi, debes ejecutarlo (haz doble click en el archivo) y sigue las instrucciones. Es importante recordar la ruta (el directorio) donde se ha instalado Python. ¡Será necesario más adelante!

Algo para tener en cuenta: en la segunda pantalla del asistente de instalación, llamada "Customize", asegúrate de ir hacia abajo y elegir la opción "Add python.exe to the Path", como en

**Linux**

Es muy posible que ya tengas Python instalado. Para verificar que ya lo tienes instalado (y qué versión es), abre una consola y tipea el siguiente comando:

$ python3 --version
Python 3.4.2
Si no tienes Python instalado o si quieres una versión diferente, puedes instalarlo como sigue:
Ubuntu
Tipea este comando en tu consola:
sudo apt-get install python3.4

# Editor de Código

Sublime Text 3

Sublime Text es un editor muy popular con un periodo de prueba gratis. Es fácil de instalar y está disponible para todos los sistemas operativos.

**Descárgalo aquí**

**https://tutorial.djangogirls.org/es/code_editor/**

# Python prompt

Para empezar a jugar con Python, tenemos que abrir una línea de comandos en nuestra computadora. Ya sabes cómo hacerlo, lo aprendiste en el capítulo de Introducción a la línea de comandos.

Una vez que estés lista, sigue las siguientes instrucciones.

Queremos abrir una consola de Python, así que escribe python3 y pulsa Enter.

```
$ python3
Python 3.4.2 (...)
Type "copyright", "credits" or "license" for more information.
>>>
```

la terminal - teclear comandos (código) dentro de la terminal de Python resulta en respuestas de Python

números y strings - en Python los números son usados para matemáticas y strings para objetos de texto

operadores - como + y *, combina valores para producir uno nuevo
funciones - como upper() y len(), realizan opciones sobre los objetos.

**https://tutorial.djangogirls.org/es/python_introduction/**

# Qué es Django?

Django es un framework para aplicaciones web gratuito y open source, escrito en Python. Es un WEB framework - un conjunto de componentes que te ayudan a desarrollar sitios web más fácil y rápidamente.

**https://tutorial.djangogirls.org/es/django/**

# Instalar Django

Puedes instalar Django usando pip. En la consola, ejecuta pip install django==1.10 (fíjate que utilizamos un doble signo igual: ==).

La versión de django a utilizar en este taller es 1.10

**pip install django==1.10**
Downloading/unpacking django==1.10
Installing collected packages: django
Successfully installed django
Cleaning up...

**https://tutorial.djangogirls.org/es/django_installation/**

# Primer Proyecto en Django

Para crear un proyecto debes utilizar el siguiente comando :

**django-admin.py startproject mysite**

Después de ejecutar el comando anterior tendrás los siguientes archivos:

```
djangogirls
├──── manage.py
└──── mysite
        settings.py
        urls.py
        wsgi.py
        __init__.py
```

manage.py es un script que ayuda con la administración del sitio. Con ello podremos iniciar un servidor web en nuestro ordenador sin necesidad de instalar nada más, entre otras cosas.

El archivo settings.py contiene la configuración de tu sitio web.

¿Recuerdas cuando hablamos de un cartero que debía comprobar donde entregar una carta? El archivo urls.py contiene una lista de los patrones utilizados por urlresolver.

Ahora debemos configurar settings.py :

Configurar una base de datos
Hay una gran variedad de opciones de bases de datos para almacenar los datos de tu sitio. Utilizaremos el que viene por defecto, sqlite3.
Esto ya está configurado en esta parte de tu archivo mysite/settings.py:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

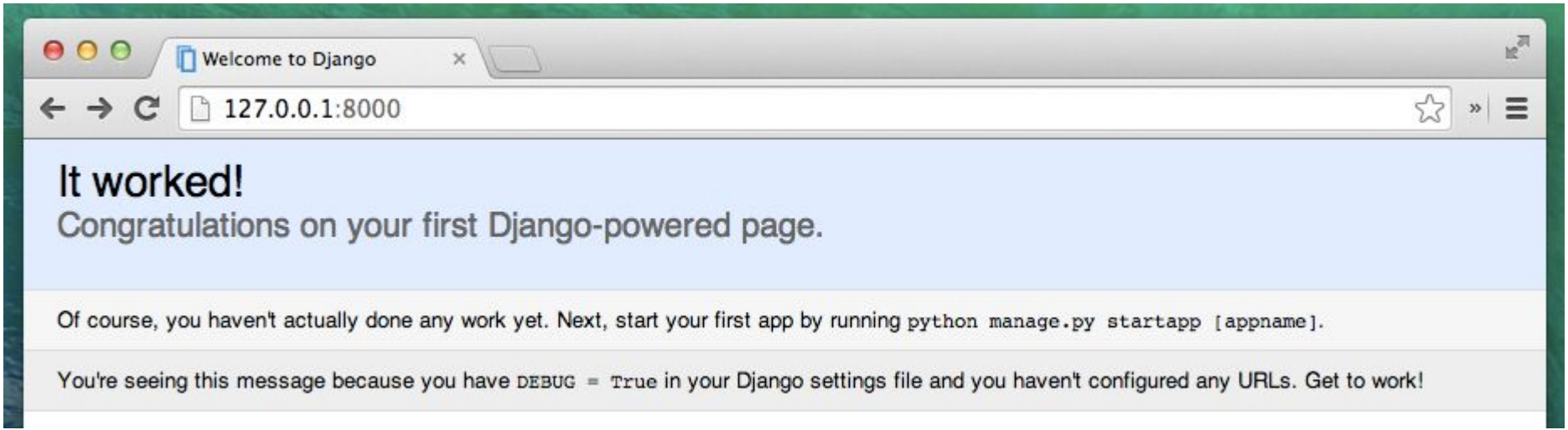
Para crear una base de datos para nuestro blog, ejecutemos lo siguiente en la consola: python manage.py migrate (necesitamos estar en el directorio de djangogirls que contiene el archivo manage.py).

¡Es hora de iniciar el servidor web y ver si nuestro sitio web está funcionando!

Debes estar en el directorio que contiene el archivo manage.py (en la carpeta djangogirls). En la consola, podemos iniciar el servidor web ejecutando python manage.py runserver

Ahora todo lo que debes hacer es verificar que tu sitio esté corriendo - abre tu navegador (Firefox, Chrome, Safari, Internet Explorer o el que utilices) e ingresa la dirección: http://127.0.0.1:8000/

**https://tutorial.djangogirls.org/es/django_start_project/**


Welcome to Django
127.0.0.1:8000
It worked!
Congratulations on your first Django-powered page.
Of course, you haven't actually done any work yet. Next, start your first app by running python manage.py startapp [appname].
You're seeing this message because you have DEBUG = True in your Django settings file and you haven't configured any URLs. Get to work!

# Creando una aplicación

Ahora para crear una aplicación debemos usar el siguiente comando:

**manage.py startapp blog**

Vas a notar que se crea un nuevo directorio llamado blog y contiene una serie de archivos.

**https://tutorial.djangogirls.org/es/django_models/**

Después de crear una aplicación también necesitamos decirle a Django que debe utilizarla. Lo hacemos en el archivo mysite/settings.py. Tenemos que encontrar INSTALLED_APPS y añadir una línea que contenga 'blog', justo por encima de ). El producto final debe tener este aspecto:

```
INSTALLED_APPS = (
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'blog',
)
```

En el archivo blog/models.py definimos todos los objetos llamados Models - este es un lugar en el cual definiremos nuestro modelo post.

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

    class Post(models.Model):
        author = models.ForeignKey('auth.User')
        title = models.CharField(max_length=200)
        text = models.TextField()
        created_date = models.DateTimeField(
                default=timezone.now)
        published_date = models.DateTimeField(
                blank=True, null=True)

        def publish(self):
            self.published_date = timezone.now()
            self.save()

        def __str__(self):
            return self.title
```

El último paso es añadir nuestro nuevo modelo a nuestra base de datos. Primero tenemos que hacer que Django sepa que tenemos algunos cambios en nuestro modelo (acabamos de crearlo), escribe python manage.py makemigrations blog. Se verá así:

```
(myvenv) ~/djangogirls$ python manage.py makemigrations blog
Migrations for 'blog':
  0001_initial.py:
  - Create model Post
```

Django preparará un archivo de migración que tenemos que aplicar ahora a nuestra base de datos escribiendo python manage.py migrate blog.

# Admin de Django

Para agregar, editar y borrar los posts que hemos modelado, utilizaremos el administrador de Django.

Vamos a abrir el archivo blog/admin.py y reemplazar su contenido con esto:

```
from django.contrib import admin
from .models import Post


admin.site.register(Post)
```

Como puedes ver, importamos (incluimos) el modelo Post definido en el capítulo anterior. Para hacer nuestro modelo visible en la página del administrador, tenemos que registrar el modelo con admin.site.register(Post).

Para poder ingresar al admin  deberás crear un superusuario - un usuario que tiene control sobre todo lo que hay en el sitio. Vuelve hacia atrás a tu línea de comandos y tipea python manage.py createsuperuser, presiona enter y tipea tu nombre de usuario (en minúsculas, sin espacios), dirección de email y contraseña cuando sean requeridos. No te preocupes que no puedes ver tu contraseña mientras la tipeas - así es como debe ser. Simplemente tipéala y presiona 'Enter' para continuar.

Si quieres saber más sobre el administrador de Django, puedes visitar la documentación de Django:
https://docs.djangoproject.com/en/1.8/ref/contrib/admin/

**https://tutorial.djangogirls.org/es/django_admin/**

# Despliegue

Es hora de desplegar tu página web sigue los pasos de la siguiente guia:

https://tutorial.djangogirls.org/es/deploy/

# Django Urls

Una URL 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! 127.0.0.1:8000 es una URL. Y http://djangogirls.org 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.

**https://tutorial.djangogirls.org/es/django_urls/**

# ¿Cómo funcionan las URLs en Django?

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

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

urlpatterns = [
    # Examples:
    # url(r'^$', 'mysite.views.home', name='home'),
    # url(r'^blog/', include('blog.urls')),

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

# ¡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 mysite/urls.py principal.

Elimina las líneas comentadas (líneas comenzando con #) y agrega una línea que importará blog.urls en el url principal ('').

Tu archivo mysite/urls.py debería verse como este:

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

    urlpatterns = [
        url(r'^admin/', include(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 más instrucciones allí.

# blog.urls

Crea un nuevo archivo vacío blog/urls.py. ¡Muy bien! Agrega estas primeras dos líneas:

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

Aquí solo estamos importando los métodos de Django y todas nuestras views del blog (todavía no tenemos ninguna, pero lo haremos en un minuto)

Luego de esto, podemos agregar nuestro primer patrón URL:

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

# Vistas de Django

Una View es un lugar donde ponemos la "lógica" de nuestra aplicación. Se solicitará información del model que creaste anteriormente y se pasará a una view que crearás en el próximo capítulo. Las vistas son sólo métodos de Python que son un poco más complicados que lo que hicimos en el capítulo de Introducción a Python.

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

https://tutorial.djangogirls.org/es/django_views/

# 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. La view 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 renderizará (construirá) nuestra plantilla blog/post_list.html.

# Introduccion html

La siguiente Guía contiene los conceptos básicos para empezar con html:

https://tutorial.djangogirls.org/es/html/

# ORM de Django

Un QuerySet es, en esencia, una lista de objetos de un modelo determinado. Un QuerySet te permite leer los datos de una base de datos, filtrarlos y ordenarlos.

Abre la consola y escribe este comando:

(myvenv) ~/djangogirls$ python manage.py shell
El resultado debería ser:

(InteractiveConsole)
>>>
Ahora estás en la consola interactiva de Django. Es como la consola de Python, pero con un toque de magia Django :)..

Vamos a mostrar todos nuestros posts primero. Puedes hacerlo con el siguiente comando:

```
>>> Post.objects.all()
Traceback (most recent call last):
      File "<console>", line 1, in <module>
NameError: name 'Post' is not defined
```

¡Uy! Apareció un error. Nos dice que no hay ningún objeto Post. Esto es correcto, ¡nos olvidamos de importarlo primero!

```
>>> from blog.models import Post
```

https://tutorial.djangogirls.org/es/django_orm/

Esto es simple: importamos el modelo Post de blog.models. Vamos a intentar mostrar todos los posts nuevamente:

```
>>> Post.objects.all()
[<Post: my post title>, <Post: another post title>]
```

Esta es una lista de las posts creadas anteriormente. Hemos creado estos posts usando la interfaz del administrador de Django. Sin embargo, ahora queremos crear nuevos posts usando Python, ¿cómo lo hacemos?

Esta es la forma de crear un nuevo objeto Post en la base de datos:

```
>>> Post.objects.create(author=me, title='Sample title', text='Test')
```

Pero hay un ingrediente faltante: me. Necesitamos pasar una instancia del modelo User como autor. ¿Cómo hacemos eso?

Primero importemos el modelo User:

```
>>> from django.contrib.auth.models import User
```

Ahora finalmente podemos crear nuestro primer post:

>>> Post.objects.create(author=me, title='Sample title', text='Test')
¡Hurra! ¿Quieres probar si funcionó?

>>> Post.objects.all()
[<Post: my post title>, <Post: another post title>, <Post: Sample title>]

# Datos Dinámicos en Plantillas

enemos diferentes piezas en su lugar: el modelo Post está definido en models.py, tenemos a post_list en views.py y la plantilla agregada. ¿Pero cómo haremos realmente para que nuestros posts aparezcan en nuestra plantilla HTML? Porque eso es lo que queremos hacer: tomar algún contenido (modelos guardados en la base de datos) y mostrarlo adecuadamente en nuestra plantilla, ¿no?

Esto es exactamente lo que las views se supone que hacen: conectar modelos con plantillas. En nuestra view post_list necesitaremos tomar los modelos que deseamos mostrar y pasarlos a una plantilla. Así que básicamente en una view decidimos qué (modelo) se mostrará en una plantilla.

https://tutorial.djangogirls.org/es/dynamic_data_in_templates/

# Plantillas

Verás, en HTML no puedes realmente poner código Python, porque los navegadores no lo entienden. Ellos sólo saben HTML. Sabemos que HTML es algo estático, mientras que Python es mucho más dinámico.

Django template tags nos permiten transferir cosas de Python como cosas en HTML, así que tu puedes construir sitios web dinámicos más rápido y fácil.

**Mostrar la plantilla post list**

En el capítulo anterior dimos a nuestra plantilla una lista de posts en la variable posts. Ahora lo mostraremos en HTML.

https://tutorial.djangogirls.org/es/django_templates/

# CSS

¿Qué es CSS?

CSS (Cascading Style Sheets, que significa 'hojas de estilo en cascada') es un lenguaje utilizado para describir el aspecto y el formato de un sitio web escrito en lenguaje de marcado (como HTML). Trátalo como maquillaje para nuestra página web ;).

Pero no queremos empezar de cero otra vez, ¿verdad? Una vez más, usaremos algo que ya ha sido realizado por programadores y publicado en Internet de forma gratuita. Ya sabes, reinventar la rueda no es divertido.

¡Vamos a usar Bootstrap!

Bootstrap es uno de los frameworks HTML y CSS más populares para desarrollar webs bonitas: https://getbootstrap.com/

Lo escribieron programadores que trabajaban para Twitter y ahora lo desarrollan voluntarios de todo el mundo.

**https://tutorial.djangogirls.org/es/css/**

# Extendiendo Plantillas

Otra cosa buena que Django tiene para tí es la extensión de plantillas. ¿Qué significa esto? Significa que puedes usar las mismas partes de tu HTML para diferentes páginas de tu sitio web.

De esta forma no tienes que repetir el código en cada uno de los archivos cuando quieres usar una misma información o un mismo esquema. Y si quieres cambiar algo, no necesitas hacerlo en cada plantilla.

**https://tutorial.djangogirls.org/es/template_extending/**

# Amplia tu Aplicación

Ya hemos completado todos los pasos necesarios para la creación de nuestro sitio web: sabemos cómo escribir un model, url, view y template. También sabemos cómo hacer que nuestro sitio web se vea lindo.

¡Hora de practicar!

Lo primero que necesitamos en nuestro blog es, obviamente, una página para mostrar un post, ¿cierto?

Ya tenemos un modelo Post, así que no necesitamos añadir nada a models.py.

https://tutorial.djangogirls.org/es/extend_your_application/

# Formularios en Django

Lo último que haremos en nuestro website es crear un apartado para agregar y editar posts en el blog. Django admin está bien, pero es bastante difícil de personalizar y hacerlo bonito. Con forms tendremos un poder absoluto sobre nuestra interfaz - ¡podemos hacer casi cualquier cosa que podamos imaginar!

Lo bueno de Django forms es que podemos definirlo desde cero o creando un ModelForm y se guardará el resultado del formulario en el modelo.

https://tutorial.djangogirls.org/es/django_forms/

# Dominio

PythonAnywhere te ha dado un dominio gratuito, pero tal vez no quieras tener ".pythonanywhere.com" al final de la URL de tu blog. Quizás quieras que tu blog viva en "www.infinite-kitten-pictures.org" o "www.3d-printed-steam-engine-parts.com" o "www.antique-buttons.com" o "www.mutant-unicornz.net", o lo que quieras que sea.

Aquí hablaremos brevemente sobre cómo obtener un dominio y veremos cómo vincularlo a tu aplicación web en PythonAnywhere. Sin embargo, deberías saber que la mayoría de los dominios son de pago, y PythonAnywhere te cobrará un valor adicional para usar tu propio nombre de dominio .

https://tutorial.djangogirls.org/es/domain/

# Enlaces de Interes

- Django's official tutorial
- New Coder tutorials
- Code Academy Python course
- Code Academy HTML & CSS course
- Django Carrots tutorial
- Learn Python The Hard Way book
- Getting Started With Django video lessons
- Two Scoops of Django: Best Practices for Django book

# Tips para Coach

https://coach.djangogirls.org/tips/