CURSOS polotic misiones

PROGRAMACIÓN WEB

CON PYTHON Y JAVASCRIPT

CLASE 5

CLASE 5

INTRODUCCIÓN AL LENGUAJE PYTHON

CURSOS polotic misiones

python™

CLASE 5

HOY

CURSOS polotic misiones

- Programación Funcional
- Tratamiento de Excepciones
- Introducción a los Frameworks
- Instalación de Django

CLASE 5

CURSOS polotic MISIONES

Programación Funcional

CLASE 5
PROGRAMACIÓN FUNCIONAL

CURSOS polotic misiones

- Además de admitir la Programación Orientada a Objetos, Python también admite el paradigma de **programación funcional**, en el que las funciones se tratan como valores como cualquier otra variable.
- En Programación Funcional las funciones son tratadas como ciudadanos de primera (first-class citizens), lo que significa que se le pueden asociar nombres, pueden ser pasadas como argumentos o retornadas desde otras funciones, como cualquier otro tipo de datos.
- Lo que permite que los programas sean escritos en estilos declarativos y compuestos, donde las pequeñas funciones son combinadas en manera modular.
- Surgió del ámbito científico universitario gracias al calculo lambda,

que era un sistema formal de computo basado en solamente funciones.

CLASE 5
PROGRAMACION FUNCIONAL

CURSOS polotic misiones

- **DECORATORS**

Una cosa que la programación funcional hace posible es la idea de un decorador, que es una función de orden superior que puede modificar otra función.

Por ejemplo, escriba un decorador que anuncie cuándo una función está a punto de comenzar y cuándo termina. Luego podemos aplicar este decorador usando un símbolo @.

```
def mi_mensaje(f):
        def wrapper():
                print("A punto de correr la función")
                f()
                print("Finalizada la función")
        return wrapper

@mi_mensaje
def hola():
        print("¡Hola, mundo!")

hola()
```

```
""" Salida:
A punto de correr la función
¡Hola, mundo!
Finalizada la función
"""
```

CLASE 5
PROGRAMACION FUNCIONAL

CURSOS polotic misiones

- **FUNCIONES LAMBDA**

Las funciones Lambda proporcionan otra forma de crear funciones en Python. Por ejemplo, si queremos definir la misma función `cuadrado` que hicimos en la clase anterior, podemos escribir:

```
cuadrado = lambda x: x * x
```

Donde la entrada está a la izquierda de `:` y la salida está a la derecha.

Esto puede ser útil cuando no queremos escribir una función completamente separada para un solo uso pequeño. Por ejemplo, si queremos ordenar algunos objetos en los que al principio no está claro cómo ordenarlos. Imagina que tenemos una lista de personas, pero con nombres y casas en lugar de solo nombres que deseamos ordenar:

```
personas = [
        {"nombre": "Harry", "casa": "Gryffindor"},
        {"nombre": "Cho", "casa": "Ravenclaw"},
        {"nombre": "Draco", "casa": "Slytherin"} ]

personas.sort()

print(personas)
```

- **FUNCIONES LAMBDA**

El codigo de nuestro diccionario produce un error:

```
Traceback (most recent call last):
  File "p:/Cursos Polo/Python con Javascript/Codigo Fuente/progfunc.py", line 5, in <module>
    personas.sort()
TypeError: '<' not supported between instances of 'dict' and 'dict'
```

Esto ocurre porque Python no sabe cómo comparar dos diccionarios para comprobar si uno es menor que el otro.

Podemos resolver este problema incluyendo un argumento `key` para la función `sort`, que especifica qué parte del diccionario deseamos usar para ordenar.

¡Veamos!

CLASE 5
PROGRAMACION FUNCIONAL

- **FUNCIONES LAMBDA**

```
personas = [
        {"nombre": "Harry", "casa": "Gryffindor"},
        {"nombre": "Cho", "casa": "Ravenclaw"},
        {"nombre": "Draco", "casa": "Slytherin"}
]

def f(unaPersona):
        return unaPersona["nombre"]

personas.sort(key=f)

print(personas)
```

Si bien esto funciona, hemos tenido que escribir una función completa que solo usamos una vez, podemos hacer que **nuestro código sea más legible mediante el uso de una función lambda**

CLASE 5
PROGRAMACION FUNCIONAL

- **FUNCIONES LAMBDA**

```
personas = [
        {"nombre": "Harry", "casa": "Gryffindor"},
        {"nombre": "Cho", "casa": "Ravenclaw"},
        {"nombre": "Draco", "casa": "Slytherin"}
]

personas.sort(key=lambda unaPersona: unaPersona["nombre"])

print(personas)
```

CLASE 5

CURSOS polotic MISIONES

Excepciones

CLASE 5
EXCEPCIONES

CURSOS polotic misiones

- Durante esta conferencia, nos encontramos con algunas excepciones diferentes, por lo que ahora analizaremos algunas formas nuevas de abordarlas.

- En el siguiente fragmento de código, tomaremos dos números enteros del usuario e intentaremos dividirlos:

```
numero1 = int(input("Ingrese primer numero: "))
numero2 = int(input("Ingrese segundo numero: "))

resultado = numero1 / numero2

print(f"{numero1} / {numero2} = {resultado}")
```

CLASE 5
EXCEPCIONES

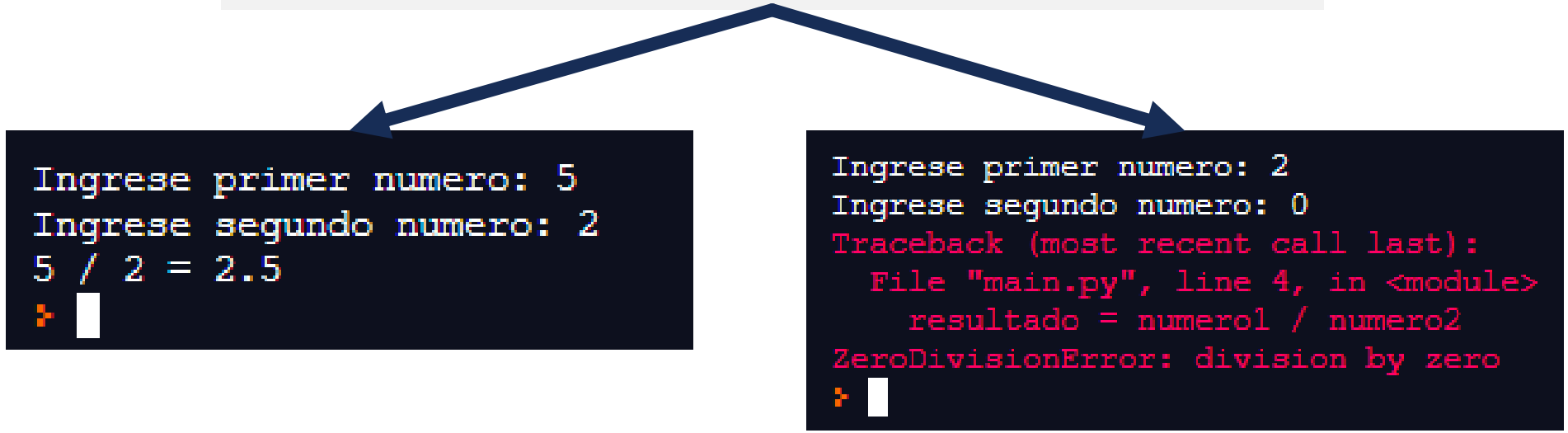
CURSOS polotic misiones

- Este código puede ofrecer dos respuestas:

```
numero1 = int(input("Ingrese primer numero: "))
numero2 = int(input("Ingrese segundo numero: "))

resultado = numero1 / numero2

print(f"{numero1} / {numero2} = {resultado}")
```



CLASE 5
EXCEPCIONES

- Podemos lidiar con este desordenado error usando Manejo de Excepciones.
- En el siguiente bloque de código, intentaremos (`try`) dividir los dos números, excepto (`except`) cuando obtengamos un `ZeroDivisionError`:

```
import sys

numero1 = int(input("Ingrese primer numero: "))
numero2 = int(input("Ingre
try:
    resultado = numero1 /
except ZeroDivisionError:
    print("Error: No se puede dividir por 0.")
    #Salir del programa
    sys.exit(1)

print(f"{numero1} / {numero2} = {resultado}")
```

```
Ingrese primer numero: 5
Ingrese segundo numero: 0
Error: No se puede dividir por 0.
repl process died unexpectedly: exit status 1
```

CLASE 5
EXCEPCIONES

CURSOS polotic misiones

- Sin embargo aún tenemos errores si el usuario ingresa caracteres que no son números:

```
Ingrese primer numero: c
Traceback (most recent call last):
  File "main.py", line 3, in <module>
    numero1 = int(input("Ingrese primer numero: "))
ValueError: invalid literal for int() with base 10: 'c'
>
```

- Tendremos que controlar también el ingreso de datos al sistema con otro `try`

CLASE 5
EXCEPCIONES

- Tendremos que controlar también el ingreso de datos al sistema con otro `try`

```python
import sys

try:
    numero1 = int(input("Ingrese primer numero: "))
    numero2 = int(input("Ingrese segundo numero: "))
except ValueError:
    print("Error: Valor no valido. ")
    sys.exit(1)

try:
    resultado = numero1 / numero2
except ZeroDivisionError:
    print("Error: No se puede dividir por 0.")
    #Salir del programa
    sys.exit(1)

print(f"{numero1} / {numero2} = {resultado}")
```

```
Ingrese primer numero: c
Error: Valor no valido.
repl process died unexpectedly: exit status 1
```

ELSE

- Puedes usar la palabra clave `else` para definir un bloque de código que se ejecutará si no se produjeron errores:

```
try:
    print("Hola")
except:
    print("Algo salió mal")
else:
    print("Nada salió mal")
```

FINALLY

- El bloque `finally`, si se especifica, se ejecutará independientemente de si el bloque `try` genera un error o no.

```
try:
    print("Hola")
except:
    print("Algo salió mal")
finally:
    print("El try y except finalizó")
```

FINALLY

- Ejemplo intentando abrir y escribir a un archivo que es de solo lectura:

```
try:
  f = open("arhivoejemplo.txt")
  f.write("Linea de prueba dentro del archivo.")
except:
  print("Algo pasó al intentar abrir el archivo.")
finally:
  f.close()
```

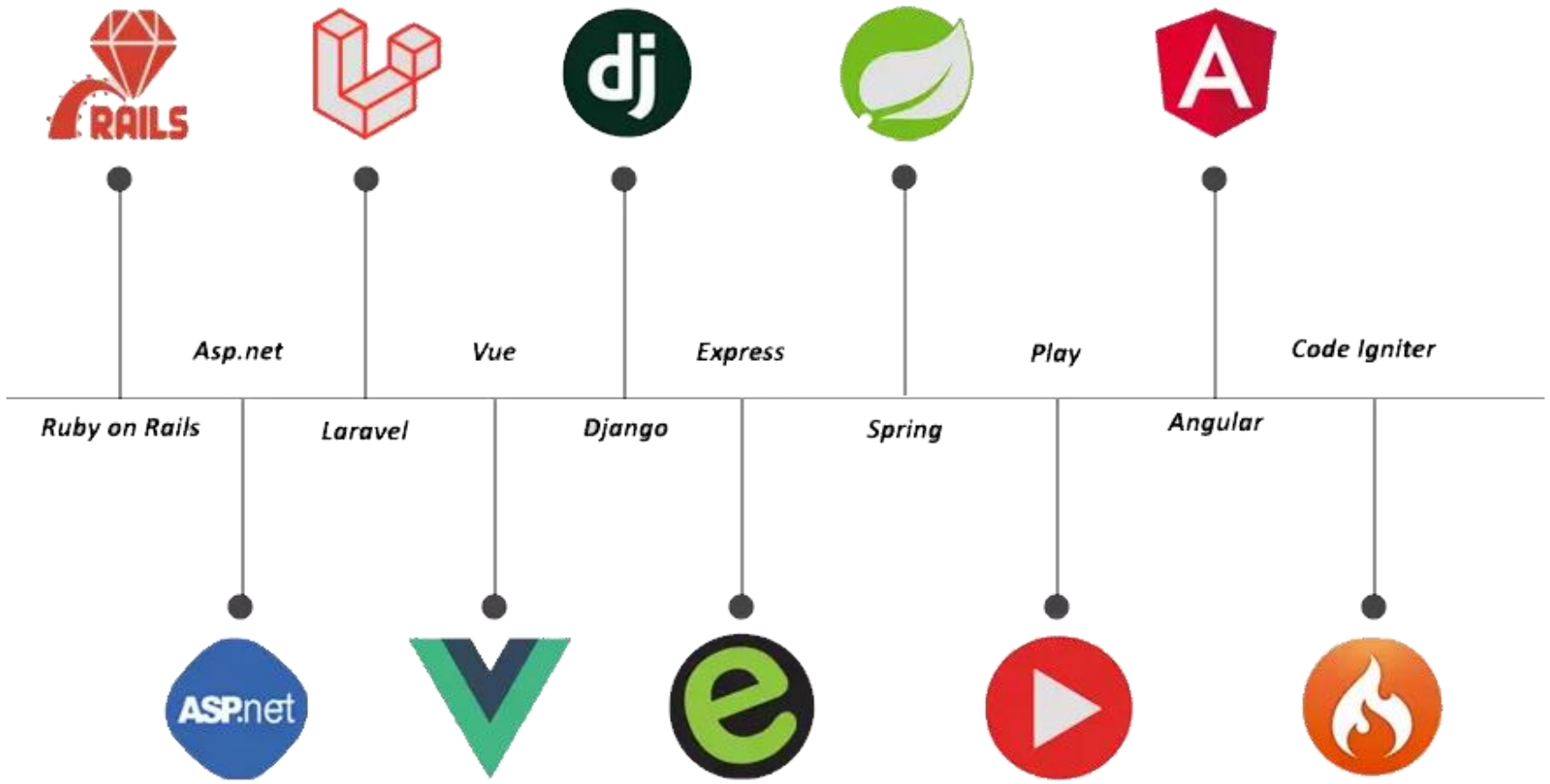
CLASE 5

CURSOS polotic MISIONES

Frameworks

CLASE 5

CURSOS polotic misiones



CLASE 5

CURSOS polotic MISIONES

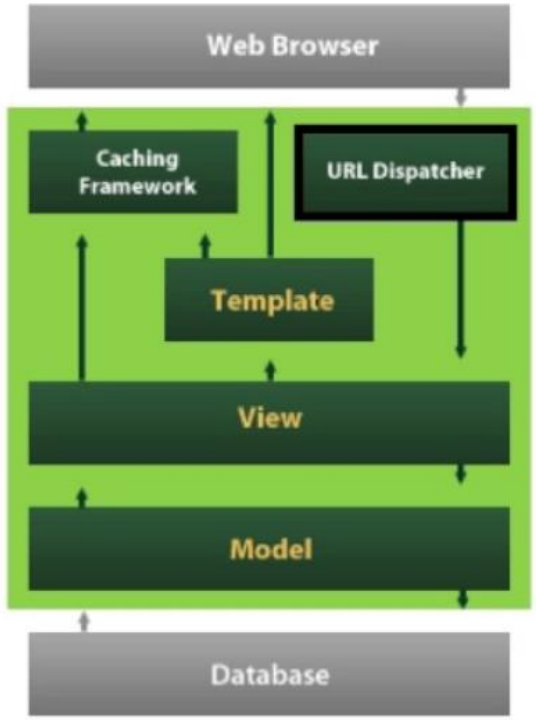
CLASE 5
DJANGO

CURSOS polotic misiones

- Django es un marco web Python de alto nivel que fomenta un desarrollo rápido y un diseño limpio y pragmático.
- Nos permitirá escribir código Python que genere HTML y CSS de forma dinámica.
- La ventaja de usar un framework como Django es que ya hay mucho código escrito para nosotros que podemos aprovechar.

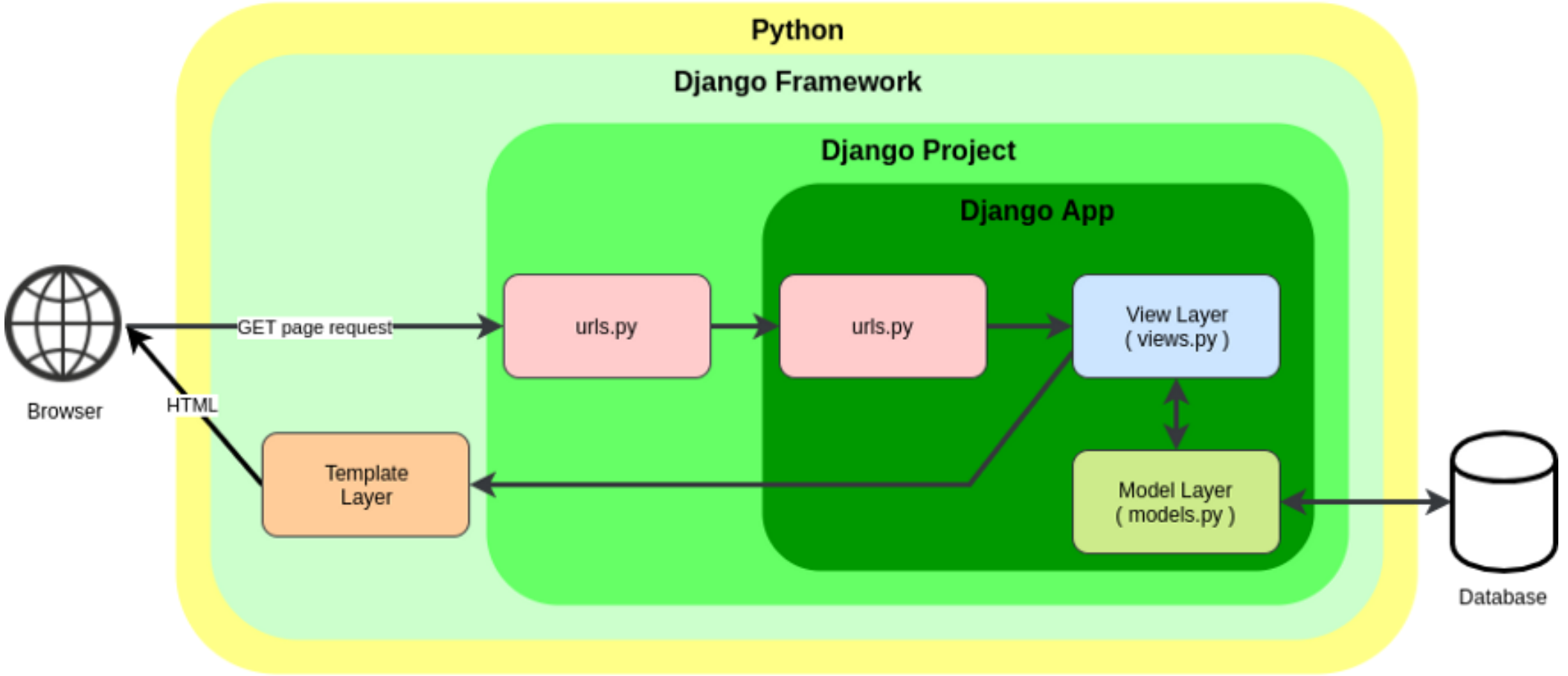
CLASE 5

CURSOS polotic misiones



CLASE 5

CURSOS polotic misiones



CLASE 5
DJANGO

CURSOS polotic misiones

- Mapeador relacional de objetos
- Interfaz de administración automática
- Diseño de URL elegante
- Sistema de plantillas
- Sistema de caché
- Internacionalización

CURSOS polotic misiones

MAPEADOR RELACIONAL DE OBJETOS

- Cada atributo del modelo representa un campo de base de datos.
- Los metadatos del modelo van en una clase interna denominada Meta.
- Los metadatos utilizados para el sitio de administración entran en una clase interna llamada Admin.
- Django le brinda una API de acceso a la base de datos generada automáticamente.

```
from django.db import models

class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

- Posibilidad de integrar con diversas bases de datos:
  - postgresql_psycopg2
  - postgresql
  - mysql
  - sqlite3
  - ado_mssql

CURSOS polotic misiones

INTERFAZ DE ADMINISTRACIÓN AUTOMÁTICA

- El tedioso problema de proporcionar una interfaz de administración ya está resuelto.
- Ayuda a proporcionar una distinción clara entre los editores de contenido y el público.

CURSOS polotic misiones

DISEÑO DE URL ELEGANTE

```
from django.conf.urls.defaults import *

urlpatterns = patterns('',
    (r'^articles/2003/$', 'news.views.special_case_2003'),
    (r'^articles/(\d{4})/$', 'news.views.year_archive'),
    (r'^articles/(\d{4})/(\d{2})/$', 'news.views.month_archive'),
    (r'^articles/(\d{4})/(\d{2})/(\d+)/$', 'news.views.article_detail'),
)
```

- "URL sin cruft"
- Mapeos realizados con RegEx para funciones de devolución de llamada.
- Al coincidir, pasa un objeto de solicitud y captura los parámetros de URL.
- Fácil referencia en plantillas para URL perfectas

CURSOS polotic misiones

SISTEMA DE PLANTILLAS

- Utiliza su propio sistema de plantillas
- La plantilla es simplemente un archivo de texto, no limitado a XML, HTML.
- Se puede utilizar para cualquier formato de texto (por ejemplo, correos electrónicos, CSV)

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

{% block title %}{{ section.title }}{% endblock %}

{% block content %}
<h1>{{ section.title }}</h1>

{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

CURSOS polotic misiones

- **EN CONCLUSIÓN…**

  - Django se adhiere al principio DRY:

    "**D**on't **R**epeat **Y**ourself" (No te repitas)

  - Sus puntos fuertes:
    - "Aplicaciones" prediseñadas
    - Plantillas de gran alcance
    - Abstracción de base de datos flexible

- **EN CONCLUSIÓN…**

  - Sus puntos débiles:
    - Aún no hay soporte *oficial* para AJAX.
    - Diseñado para integrarse estrechamente con el sistema de plantillas de Django.
    - Es la web.

CURSOS polotic MISIONES

- **INSTALANDO DJANGO…**

  - Para comenzar, tendremos que instalar Django, lo que significa que también tendrá que instalar `pip` si aún no lo ha hecho.
  - Una vez que tenga Pip instalado, puede ejecutar

    `pip3 install Django` en su terminal para instalarlo.

CURSOS polotic MISIONES

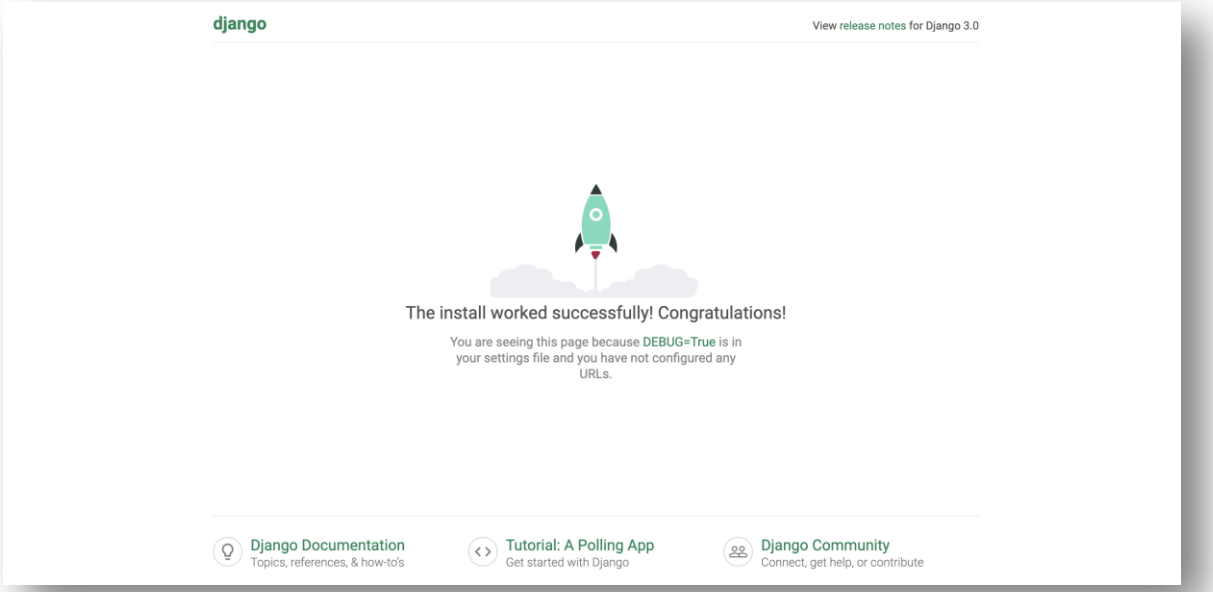
- Después de instalar Django, podemos seguir los pasos para crear un nuevo proyecto Django:

1. Ejecutar `django-admin startproject NOMBRE_PROYECTO` para crear varios archivos de inicio para nuestro proyecto.
2. Ejecutar `cd NOMBRE_PROYECTO` para navegar al directorio del nuevo proyecto.

3. Abre el directorio en el editor de texto que quieras. Notarás que se han creado algunos archivos automáticamente. No es necesario que analicemos la mayoría de ellos por ahora, pero hay tres que serán muy importantes desde el principio:
   - `manage.py` es lo que usamos para ejecutar comandos en nuestro terminal. No tendremos que editarlo, pero lo usaremos con frecuencia.
   - `settings.py` contiene algunos ajustes de configuración importantes para nuestro nuevo proyecto. Hay algunas configuraciones predeterminadas, pero es posible que deseemos cambiar algunas de ellas de vez en cuando.
   - `urls.py` contiene instrucciones sobre a dónde se debe dirigir a los usuarios después de navegar a una determinada URL. Ejecutar `cd NOMBRE_PROYECTO` para navegar al directorio del nuevo proyecto.

4. Iniciar el proyecto ejecutando `python manage.py runserver`. Esto abrirá un servidor de desarrollo, al que puedes acceder visitando la URL proporcionada.

- Este servidor de desarrollo se ejecuta localmente en tu máquina, lo que significa que otras personas no podrán acceder a tu sitio web. Esto debería llevarte a una página de destino predeterminada:



PROGRAMACIÓN WEB

CON PYTHON Y JAVASCRIPT

CURSOS polotic MISIONES

GRACIAS

programacionpolotic@gmail.com

CLASE 5