PYTHON DJANGO:

ULTIMATE BEGINNERS COURSE

By Arno Pretorius

NOTICE: THESE SLIDES ARE COPYRIGHTED AND ARE ONLY FOR PERSONAL USE.

- These slides are only for those that are currently enrolled in the **Python Django - Ultimate Beginners Course**
- Good luck with your learning in this course!

© Arno Pretorius

SECTION:

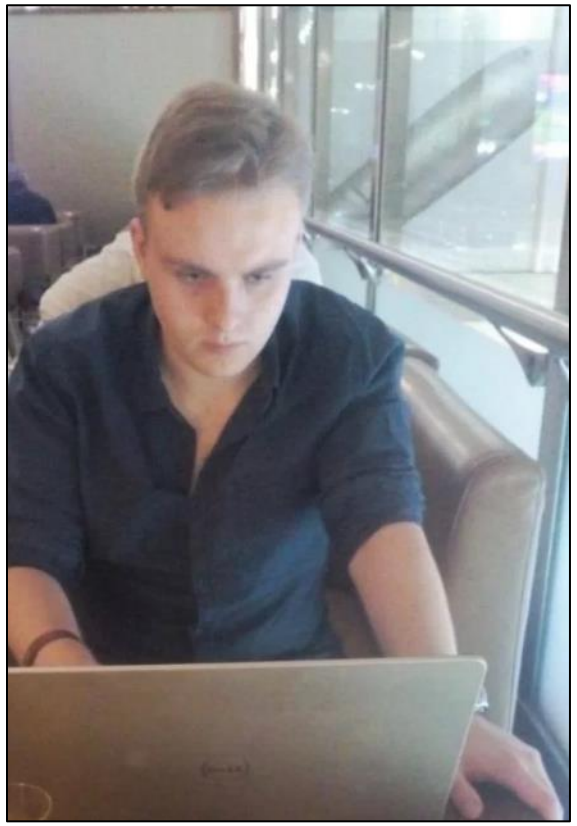
COURSE INTRODUCTION

© Arno Pretorius

PYTHON DJANGO:

ULTIMATE BEGINNERS COURSE

By Arno Pretorius



Arno Pretorius

I'm a qualified IT teacher who has taught programming both in-person and online.

Over the years, I have created and deployed many real-world Django based applications, including a job portal for university graduates and an exclusive social network.

Trust me you are in good hands!


HTML


CSS


B

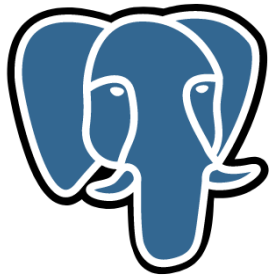

JS


aws


django


heroku


SQLite



Section:

1

2

3

4

5

Structured carefully in order to maximize simplicity and efficiency.

Section:

1

2

3

4

5

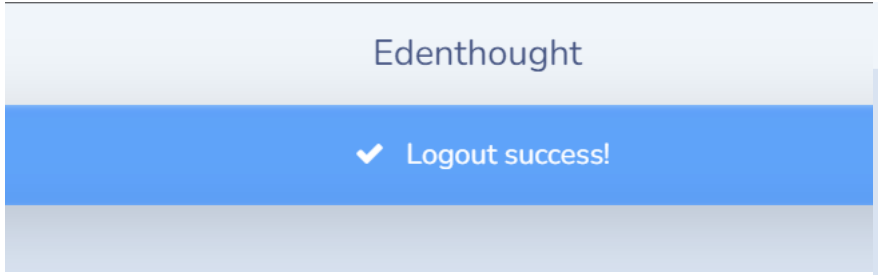
Completely new to Django

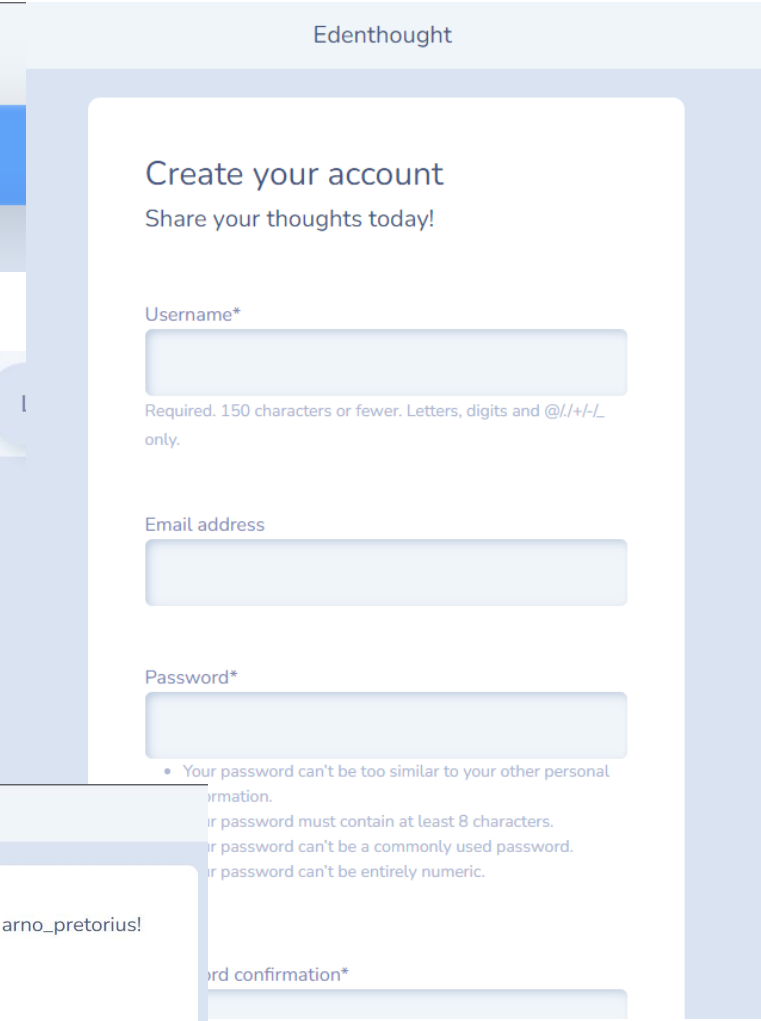
Some knowledge of Django

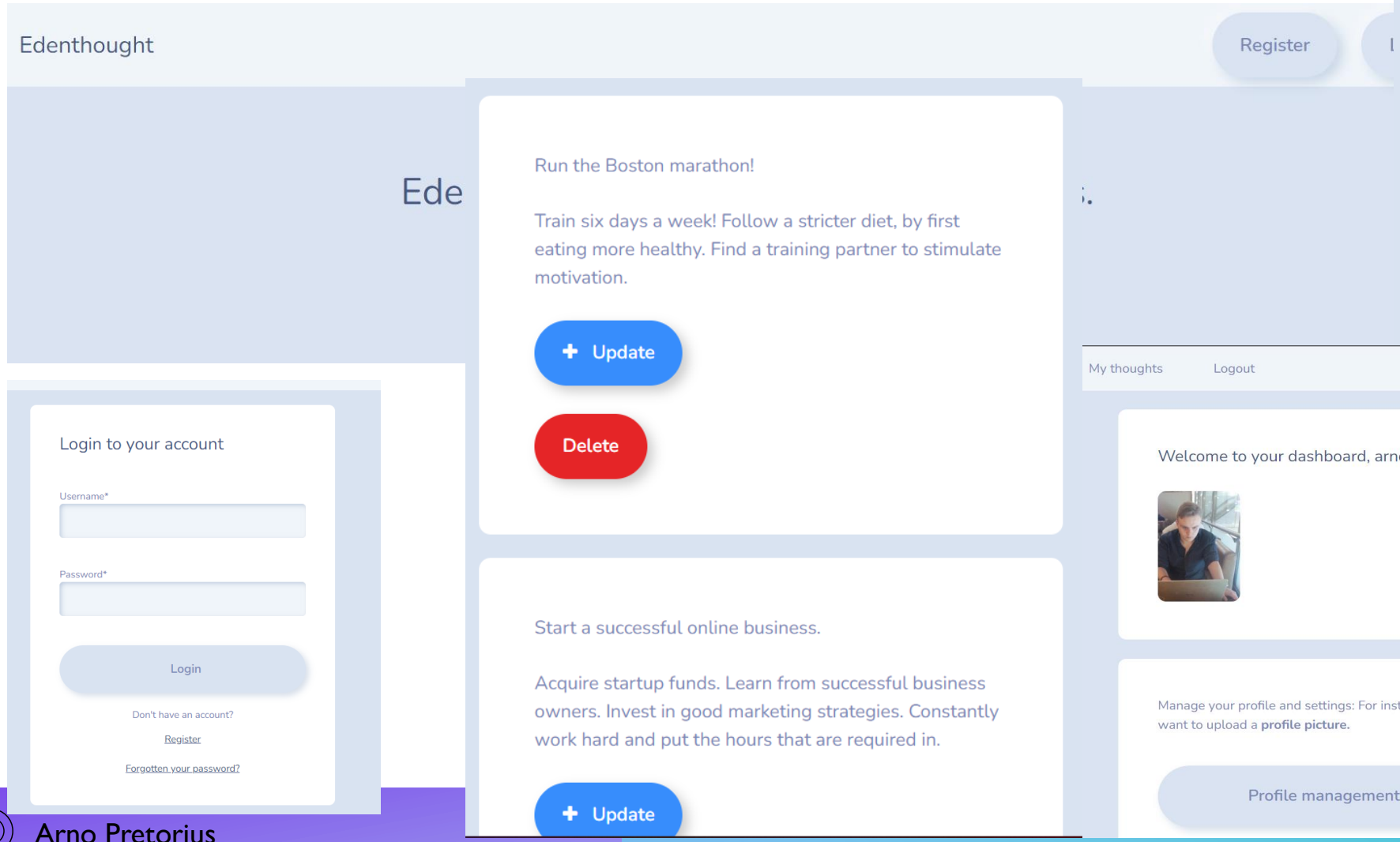
Learn Django and code our website.

Connect to our website from anywhere in the world!

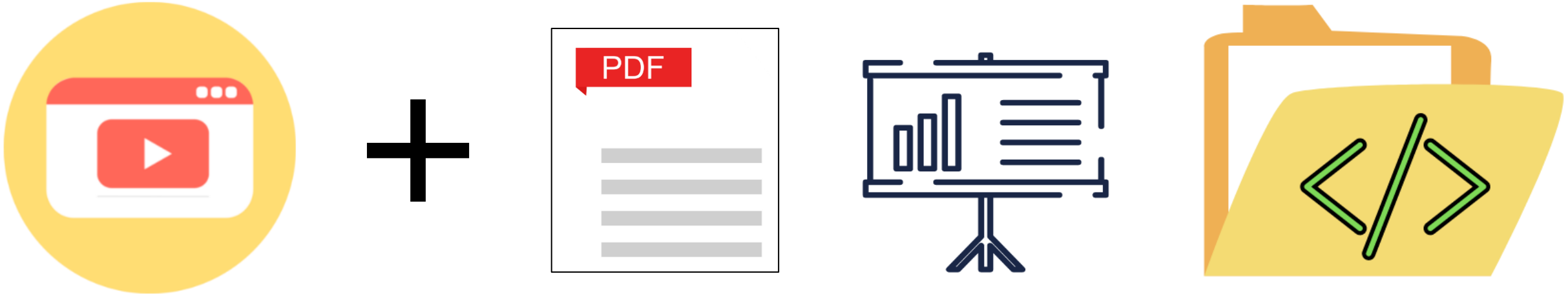
Project snippets



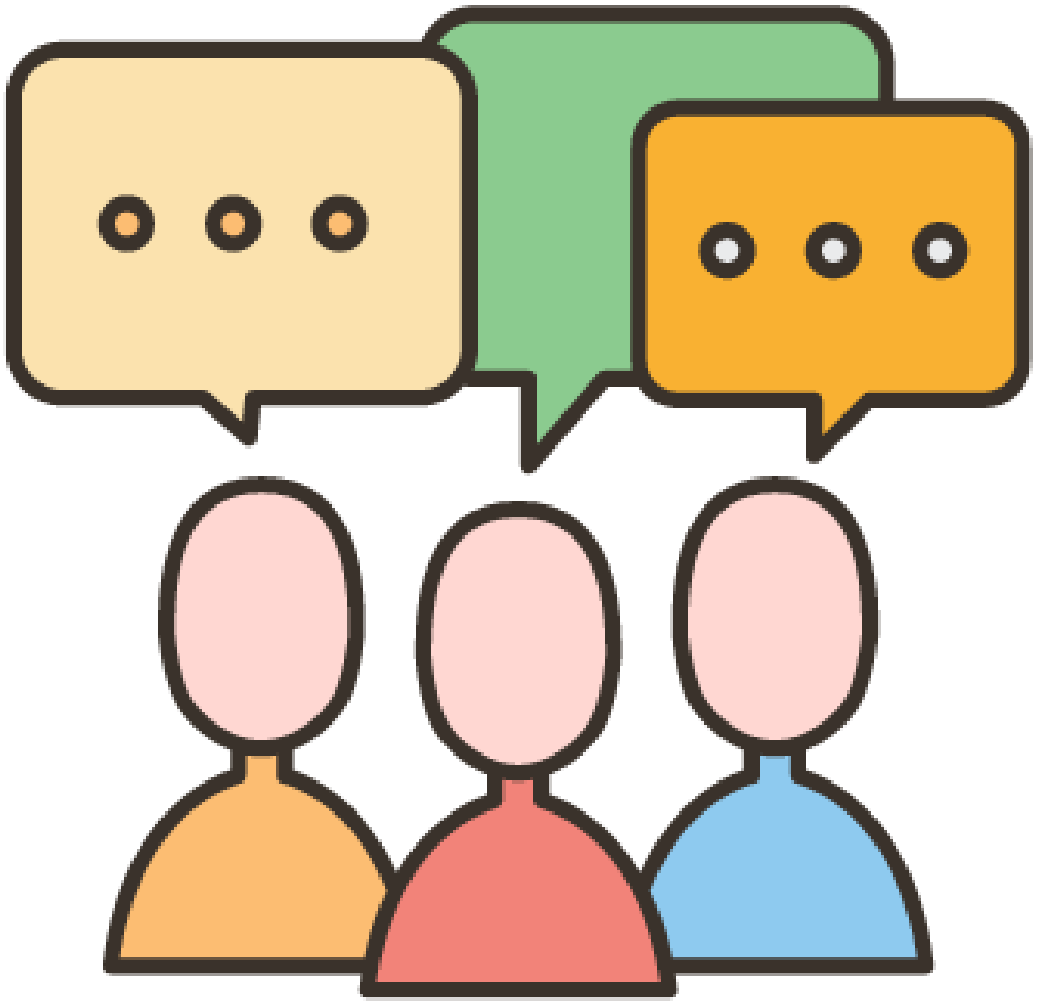




Over 70 video lectures



Along with PDF-walkthroughs, lecture slides, code snippets/references and more.



Access to the Q/A forums

Course overview

© Arno Pretorius

An overview of the course:

- I will teach you the basics of Django
- We will create a simple Django web application and take it **step-by-step**
- In this course we will mix theory with hands-on practical demonstrations
- **Don’t stress we will go slowly and tackle everything together!**

Prerequisite:

➢ A basic understanding of HTML, CSS, Python and a little bit of JavaScript

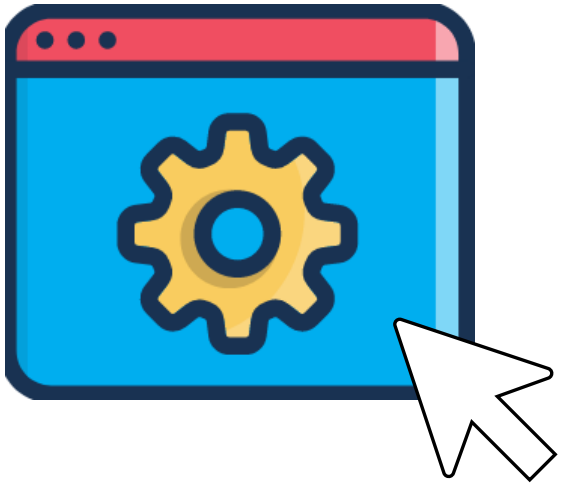
© Arno Pretorius

Just a quick note...

© Arno Pretorius

A quick note:

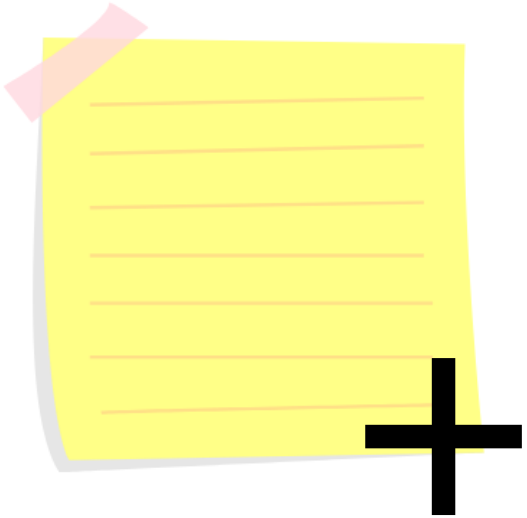
Practical / hands-on activity



© Arno Pretorius

Another quick note:

Includes a downloadable PDF guide.

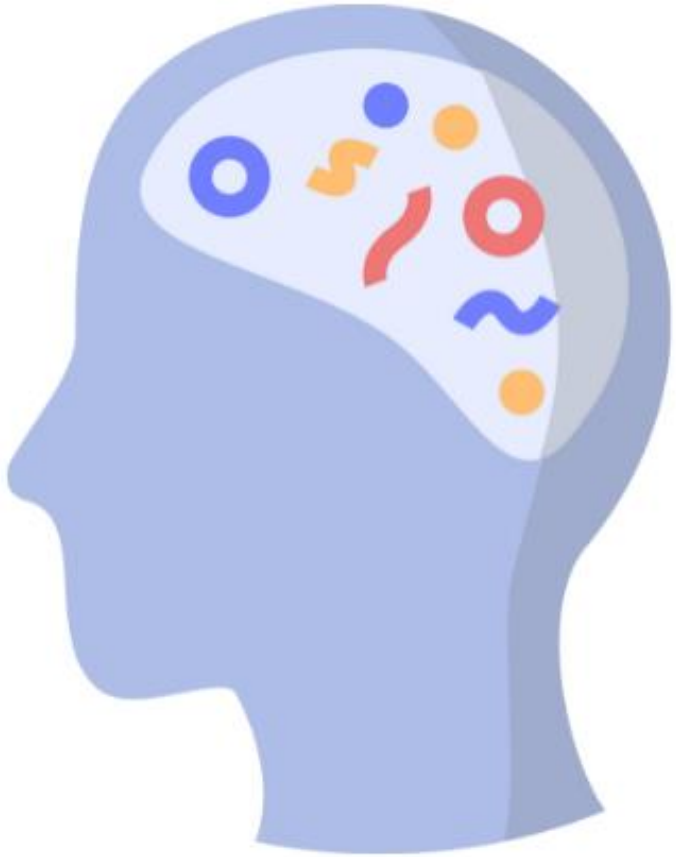


© Arno Pretorius

Thought process

© Arno Pretorius

Before we begin...



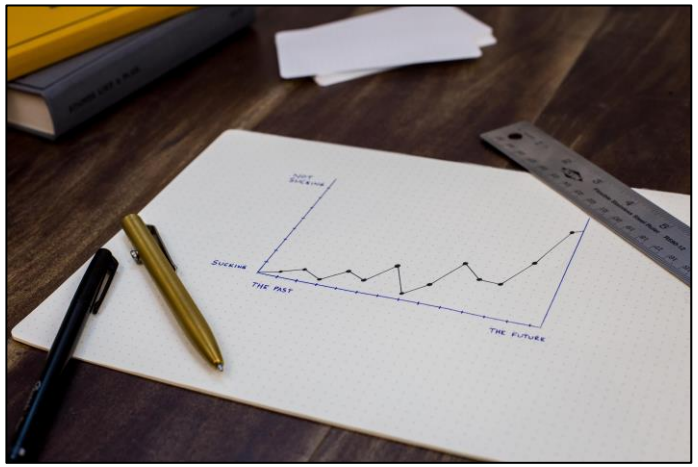
Structure our thought process

© Arno Pretorius

One step at a time...

© Arno Pretorius

Focus on progress and not on time



- "Today, I installed python and set up a virtual environment
- "Today, I learned about authentication."

- "I'm struggling to understand all these concepts, It's been 2 weeks already."

Project introduction and demonstration

© Arno Pretorius

Project introduction

We will create a journal website where users will be able to record their thoughts

Users will be able to create and login to their accounts, post their thoughts, manage their profile, and do so much more

© Arno Pretorius

Django:

A brief introduction

© Arno Pretorius

Popular python web frameworks

Although, both are popular, ***Django takes the first prize!***

So, about Django?

- Django is a high-level Python based web framework
- It is a back-end/server-side framework (E.g., Ruby on Rails, Laravel)
- Allows for rapid development
- Secure and highly scalable
- It is known as a “batteries-included” framework

To learn more about Django, please check out:
https://www.djangoproject.com/

Django:

Real-world applications

© Arno Pretorius

SECTION:

CODE AND SLIDES – DOWNLOAD

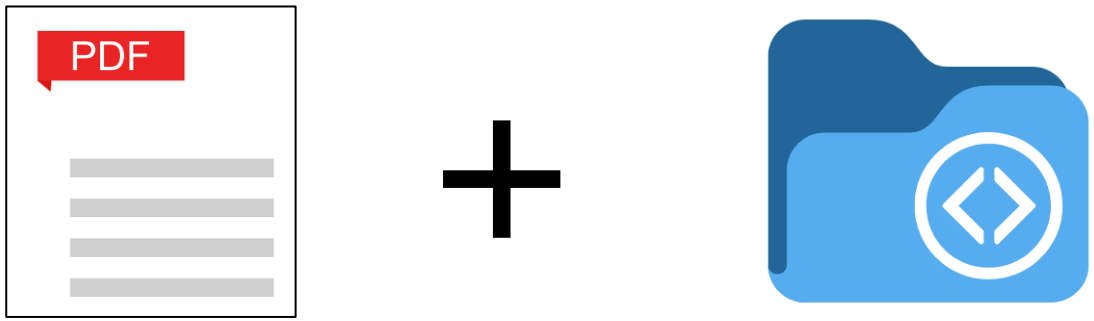
© Arno Pretorius

Resources

© Arno Pretorius

Resources - download

- The project code will be attached as a zip file (not via GitHub - as I state later in the course)
- The lecture slides will be attached as a downloadable PDF



© Arno Pretorius

SECTION:

BASELINE INSTALLATION AND SETUP

© Arno Pretorius

Python:

Installation and setup

© Arno Pretorius

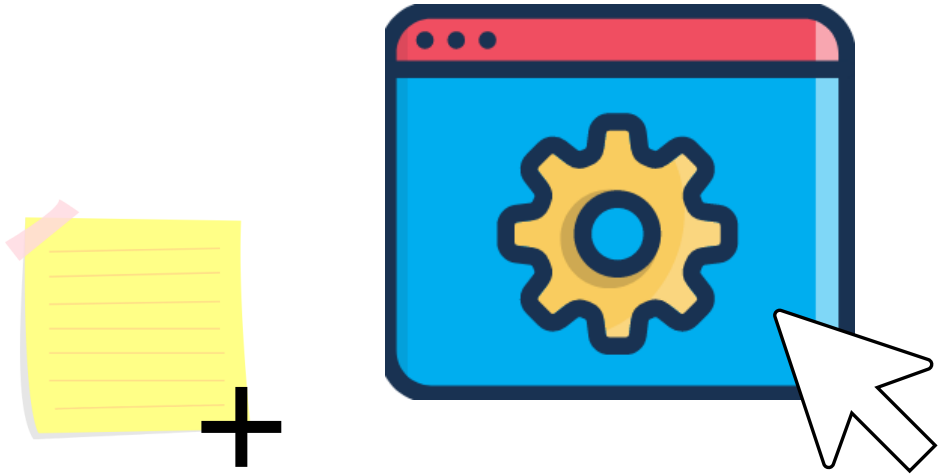
Python installation and setup

- Python needs to be installed on our computer before we can use Django
- We will install and setup Python version 3.9.13
- It is **highly recommended** that you use the exact same version for the purpose of this course
- During the process, be sure to add Python to your path

© Arno Pretorius

Practical

- Install and setup Python



© Arno Pretorius

Visual Studio Code:

Installation and setup

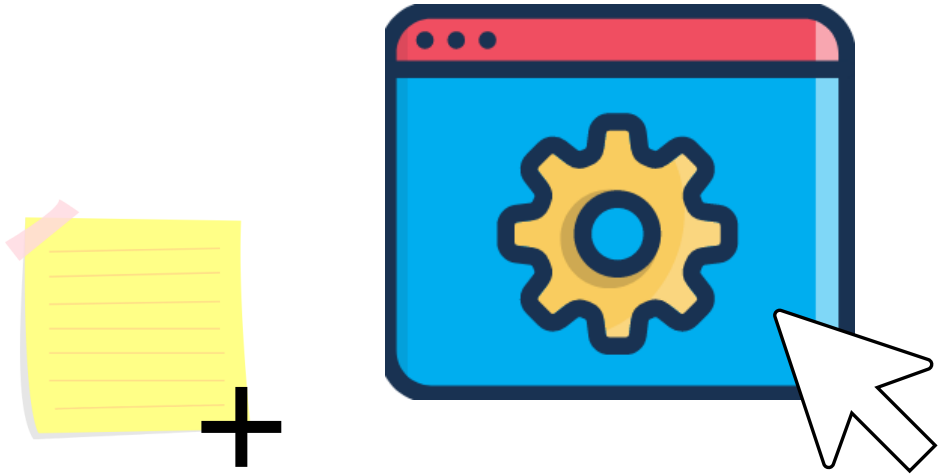
© Arno Pretorius

VS code installation and setup

- We will need to use a **source-code editor** to write our code
- There are many source-code editors, such as: VS Code, Sublime Text and Atom
- It is highly recommended that you use VS Code for the purpose of this course

Practical

- Install and setup VS code



© Arno Pretorius

SECTION:

PROJECT SETUP

© Arno Pretorius

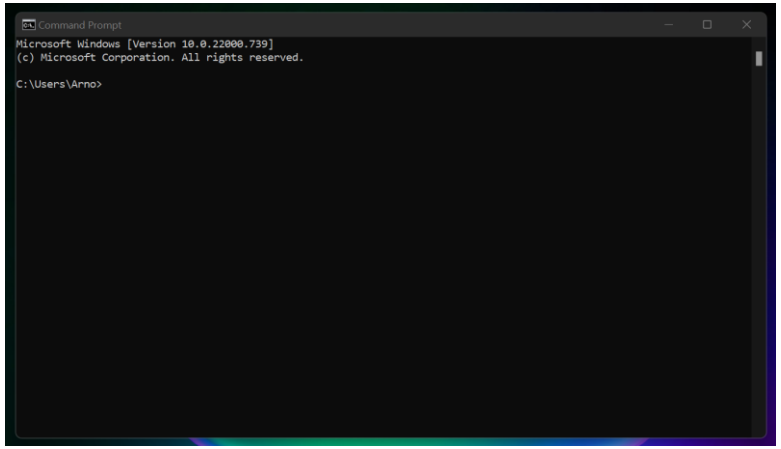
A quick overview of the CMD

© Arno Pretorius

The CMD

- **CMD** = Command prompt
- It is a command-line interpreter of the Windows operating system
- Can be used for executing **various commands** for our Django project

CMD →



Practical

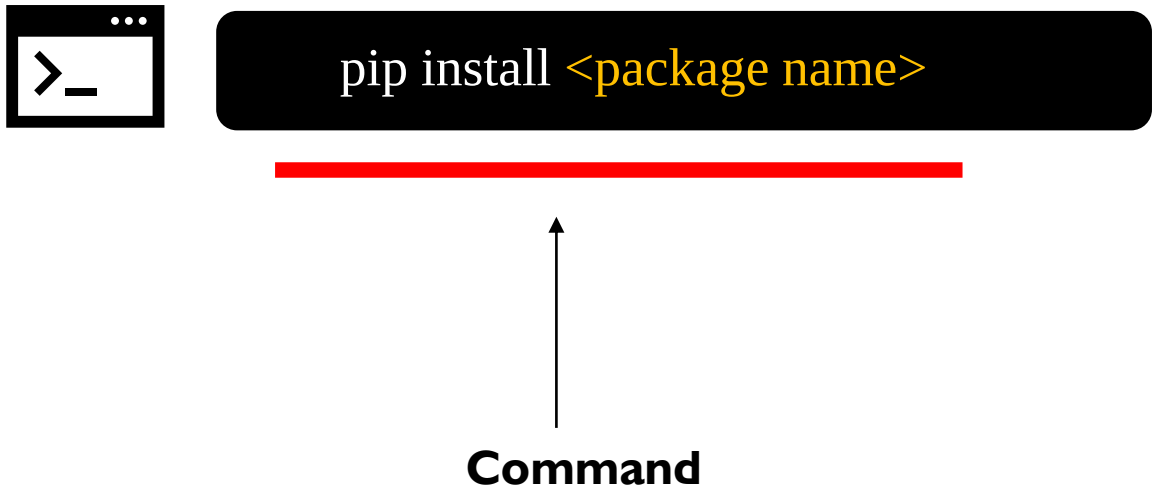
- Explore the CMD

What is pip?

© Arno Pretorius

What is pip?

- Pip is a tool that is used to install and manage Python packages
- Use the following command in your CMD to install a package:



Practical

- Identify how to utilize pip install

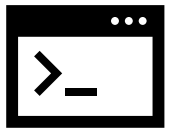
© Arno Pretorius

CMD for executing commands

© Arno Pretorius

CMD for executing commands

- Below are a few common CMD commands that you will need to know:



cd <change directory>	Change directory
mkdir <directory name>	Make a directory (folder)
rmdir <directory name>	Remove a directory (folder)
cls	Clear the CMD

Practical

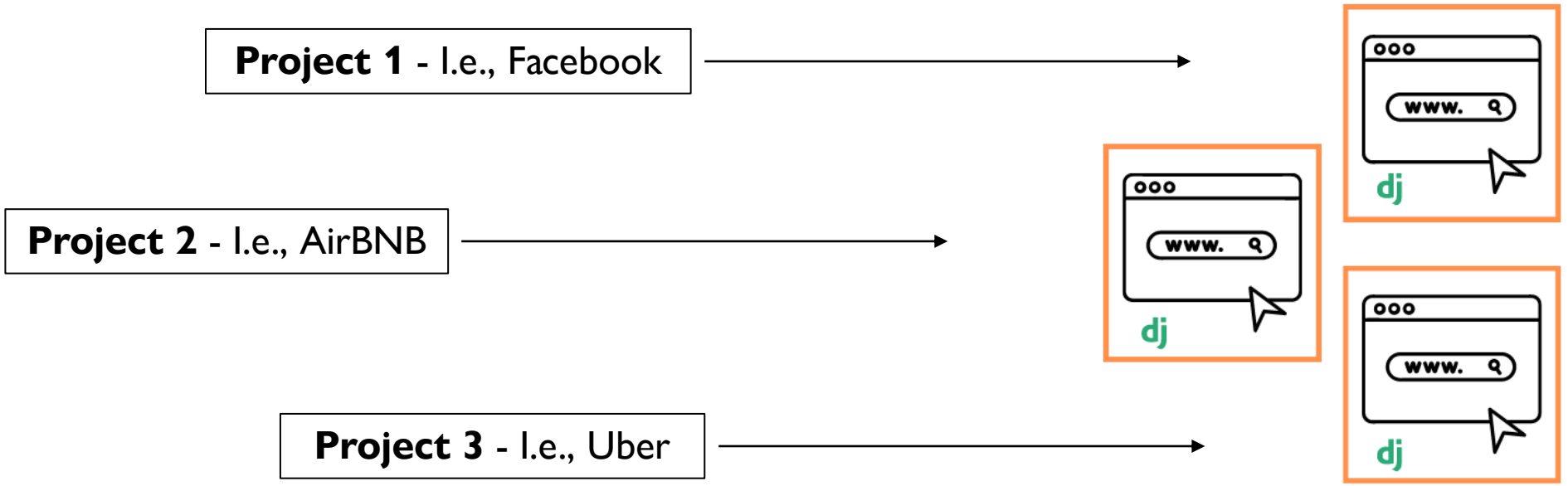
- Practice using basic CMD commands

Setting up a virtual environment

© Arno Pretorius

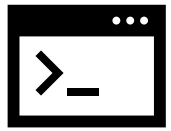
What is a virtual environment?

- A **virtual environment** is a tool that helps to isolate our **independent projects** with its own unique set of dependencies and packages.



Installing a virtual environment

- To install a virtual environment, use the following command:

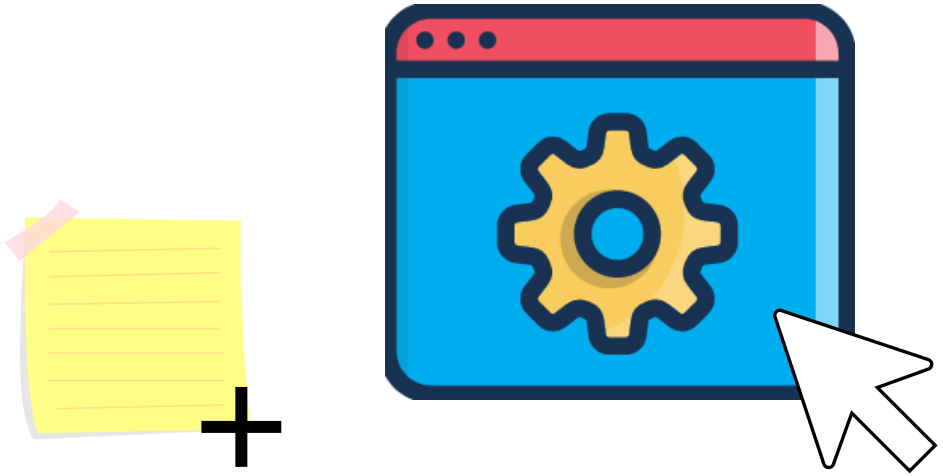


```
pip install virtualenv
```

This will be the **only time** that we install a package **globally** on our system, from now on all packages will be installed in their own virtual environments…. More on that later.

Practical

- Setup a virtual environment for our project

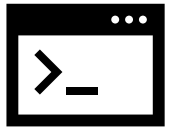


Django project setup and installation

© Arno Pretorius

Installing Django

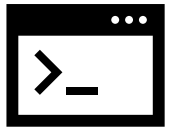
- To install Django, use the following command:



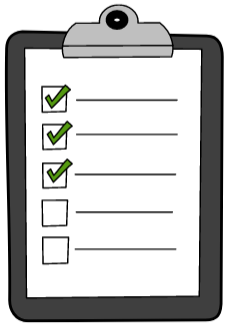
```
pip install django
```

Create a Django project

- To create a Django project, use the following command:



```
django-admin startproject <project name>
```



Practical

- Install and setup our Django project

Examining our Django project's default files

© Arno Pretorius

Examining the default project files

- There are many **default files** that come with our **Django project**
- All have their own unique role for configuration and development

© Arno Pretorius

Examining the default project files...

These files include:

- manage.py
- settings.py
- urls.py
- asgi.py
- wsgi.py
- __init__.py
- db.sqlite3

NB!

Practical

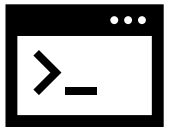
- Examine the default files in our project

Testing our server

© Arno Pretorius

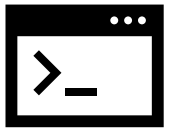
Testing our server

- To run our development server, we need to invoke the following command:



```
python manage.py runserver
```

- Connect to our server with the URL below:



```
http://127.0.0.1:8000/
```

Practical

- Test our server

SECTION:

APPS

© Arno Pretorius

The concept and creation of a Django app

© Arno Pretorius

Apps in Django

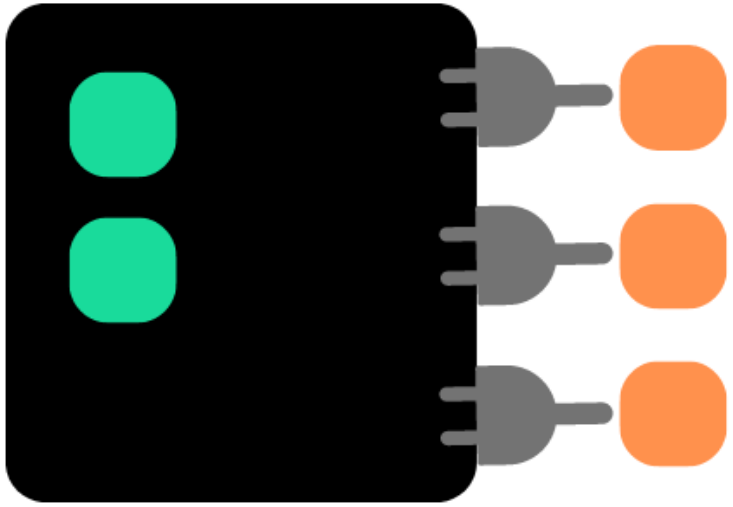
Django project:

- Is composed of a collection of various settings, configurations and apps
- We can either have one or multiple apps in a single Django project

Django App:

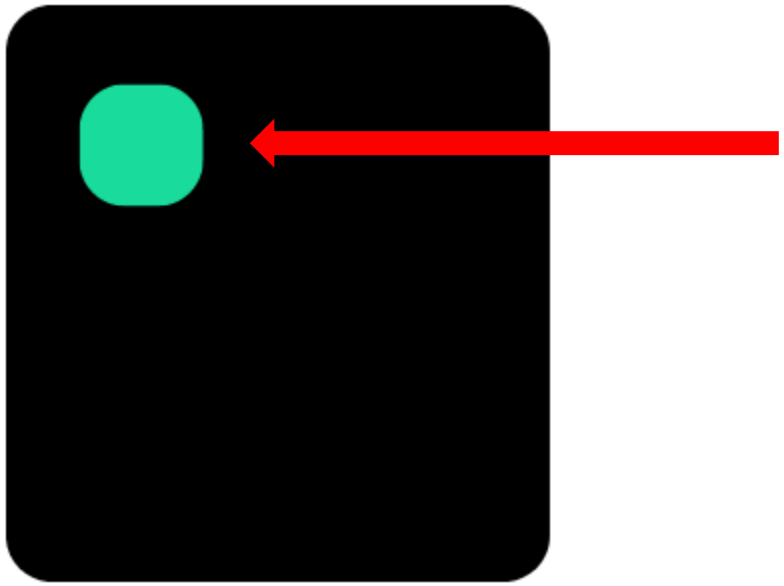
- An application that serves a unique purpose.
- An app typically consists of URLs, views, templates and models.

Apps in Django...



- **Black** represents our Django project
- **Green** represents the apps that we create ourselves
- **Orange** represents **external apps** that have already been created by others. These apps are **installed** by **PIP** (**grey**)

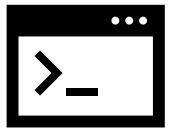
Apps in Django...



- We will create a simple journal website with one app

How to create a Django app?

- To create a Django app, use the following command:



```
django-admin startapp journal
```

Practical

- Create a Django app

© Arno Pretorius

Examining our Django app's default files

© Arno Pretorius

Examining the default app files

- There are many **default files** that come with creating a **Django app**
- All have their own unique role for configuration and development

Examining the default project files...

These files include:

- admin.py
- apps.py
- models.py
- tests.py
- views.py

NB!

Folder:

- /migrations

Practical

- Examine the default files in our app

Configuring our app

© Arno Pretorius

Configuring our app

- Now that we have created and examined our files associated with our app, we can now configure our app so we can use it.

© Arno Pretorius

Practical

- Configure our app

SECTION:

VIEWS, URLS AND TEMPLATES

© Arno Pretorius

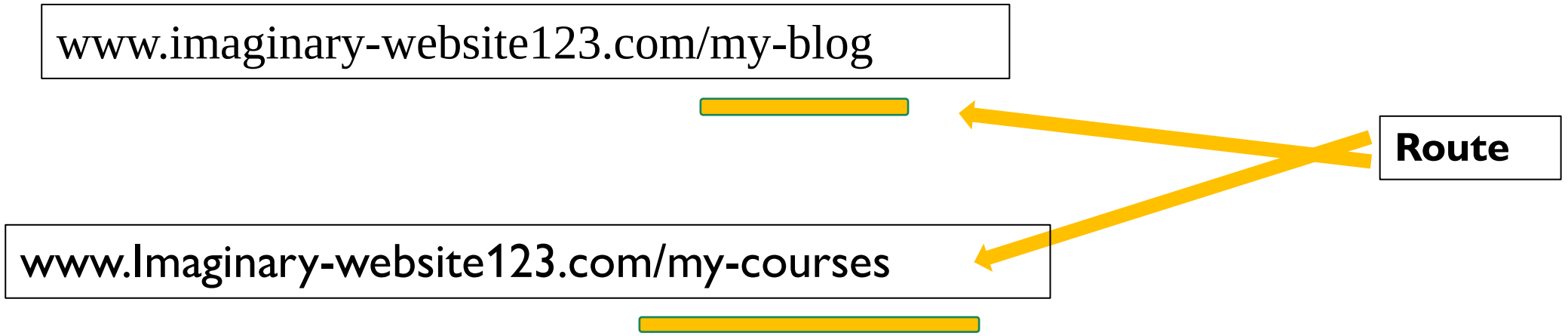
Understanding the concept of URL's and views

Part 1

© Arno Pretorius

What is a URL used for?

- To locate a particular web page on your browser



What is a view and what is it used for?

- They are Python **functions** or **classes** that are used to **return a response**

```
from django.http import HttpResponse

def register(request):

    return HttpResponse('This is the registration page')
```

Pass the **request** object to make this function a view so we can **handle** our requests

Creating a simple web page

```python
from django.urls import path
from django.http import HttpResponse
```

```python
def register(request):

    return HttpResponse('This is the registration page')
```

```python
urlpatterns = [

    path('register', register)

]
```

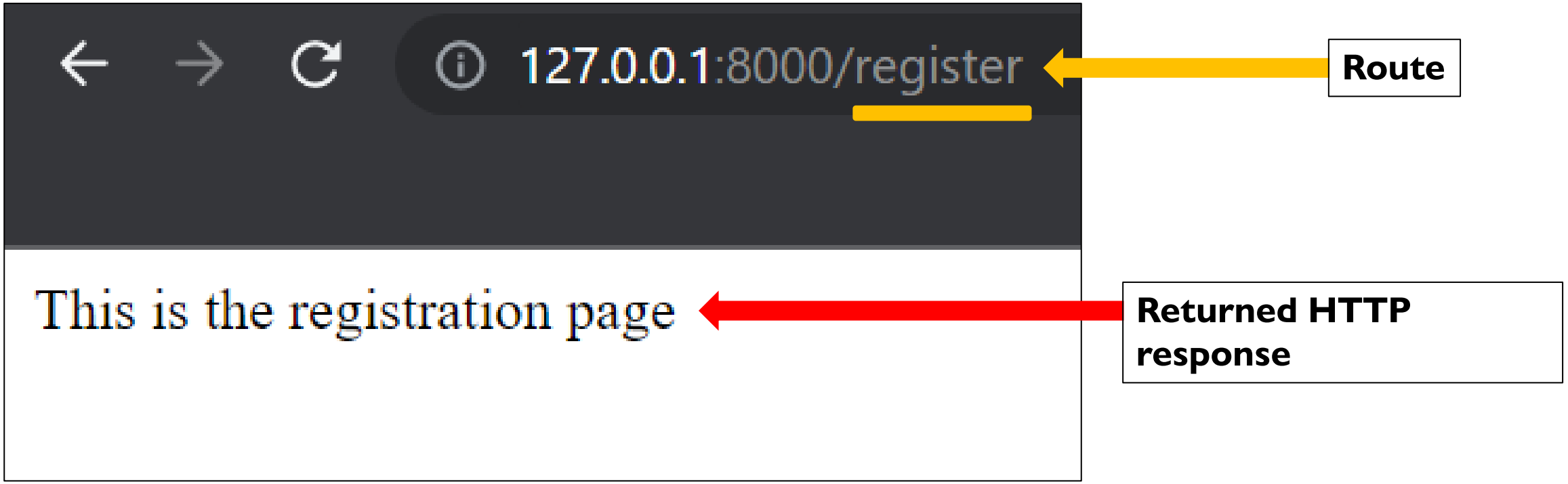
View

Route

URL patterns consist of a list of our **url's**

URL path

Creating a simple web page...



© Arno Pretorius

Practical

- Create a simple web page with a view and a url

Understanding the concept of URL's and views

Part 2

© Arno Pretorius

So, about the views.py file?

- From now on we will use the **views.py** file as it's supposed to be used for **writing** our **views**
- *We will also setup our urls.py and views.py file in our Django app properly*

Practical

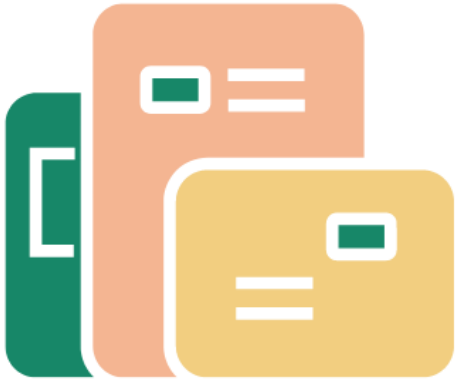
- Setup urls.py and views.py in our **Django app**

Understanding the concept of templates

© Arno Pretorius

What is a template?

- A **template** represents the layout and structure of a particular file
- This may include a simple **HTML file**



Practical

- Configure Django to render templates

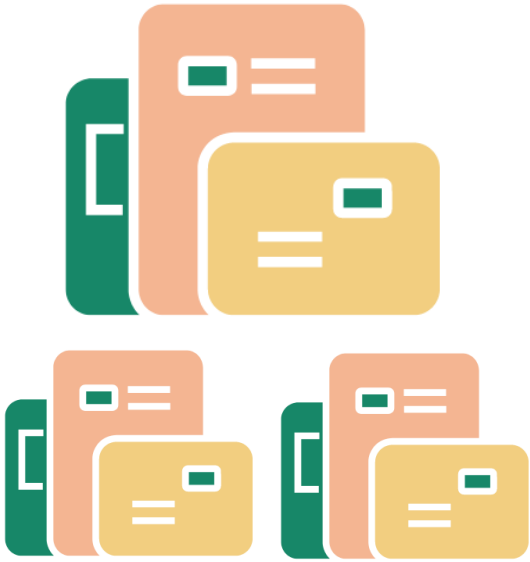
© Arno Pretorius

Template inheritance

© Arno Pretorius

Template inheritance

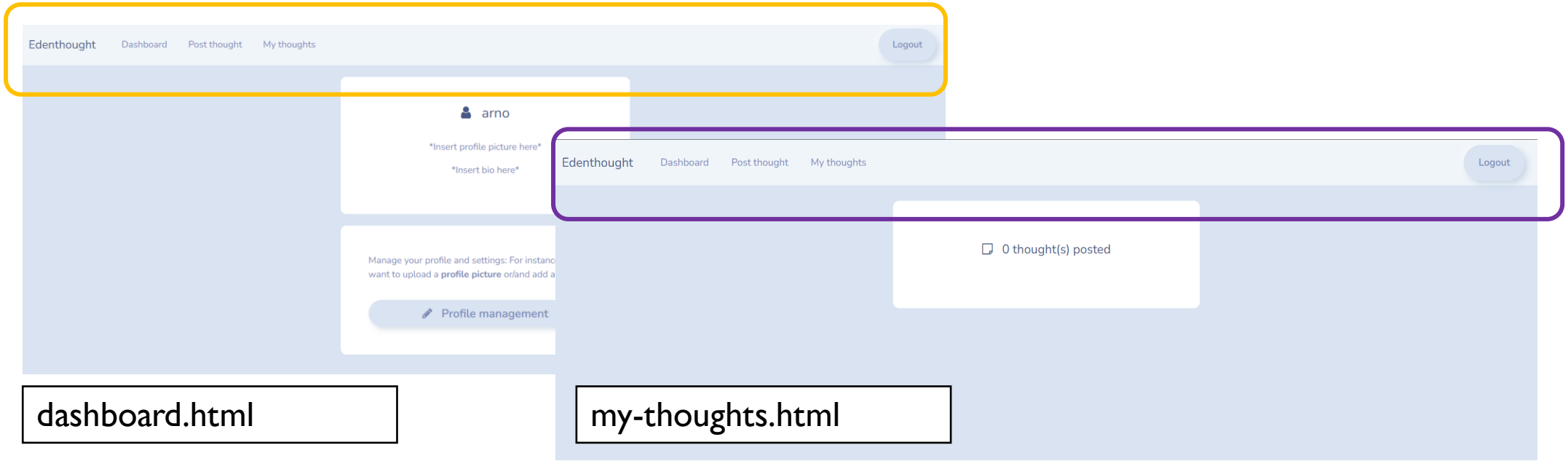
- Allows you to build a base template which contains elements that you can re-use in other templates



Template inheritance...

Try and envision it this way…

E.g., Inherit a navbar from the base template (parent - child)



dashboard.html

my-thoughts.html

Practical

- Perform template inheritance

© Arno Pretorius

Rendering data into a template

Part 1

© Arno Pretorius

Django template language - (DTL)

- The **DTL** is a so-called **“mini-language”** that we can use to **write logic** within our **templates**

- **Pass variables** to our templates
- Create **if statements** to handle our conditions
- Create **for loops** to loop through our data

Lists and dictionaries

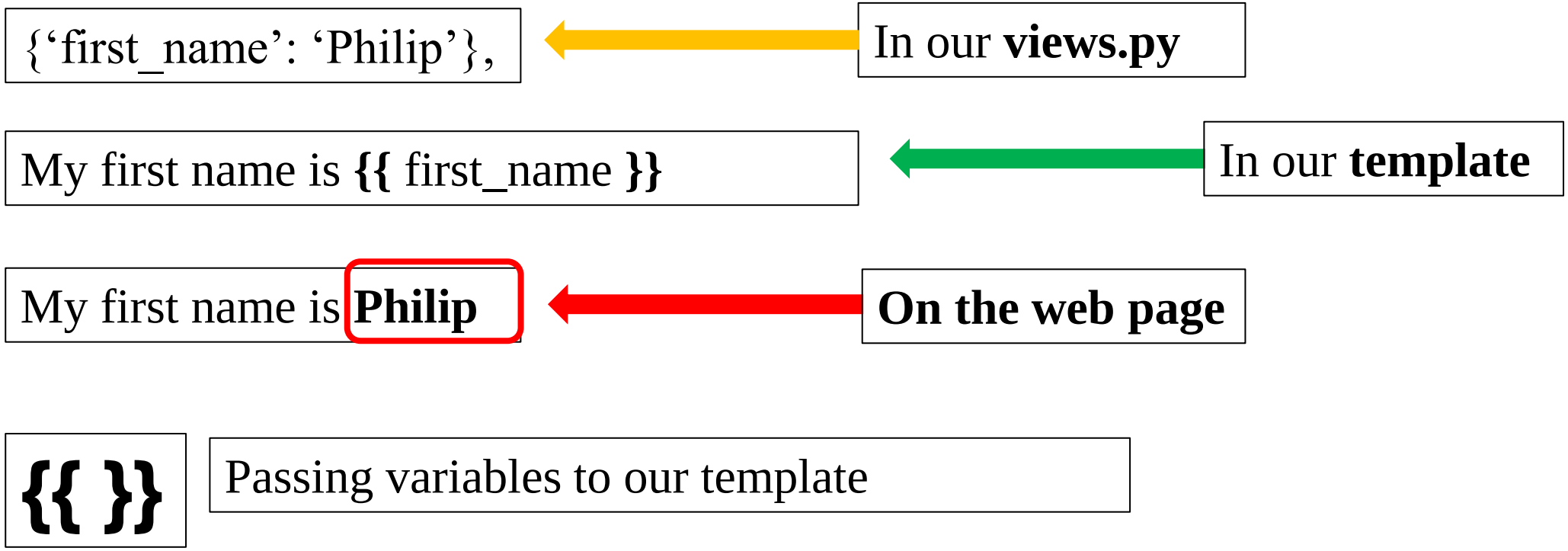
```
# - List with dictionaries

clientList = [

    {
        'id' : '1',
        'name' : 'John Smith',
        'profession' : "Web developer"


    },


    {
        'id' : '2',
        'name' : 'Sarah Evans',
        'profession' : 'Electrical engineer'

    },


]
```

clientList is our **list**

Dictionaries - each one is almost '**object-like**'

© Arno Pretorius

Django template language - (DTL)...



Practical

- Pass variables to our template

Rendering data into a template

Part 2

© Arno Pretorius

Django template language - (DTL)...

Tags **{% %}**

- They allow us to **write python-styled logic**
- There are also **pre-built tags** as well

Examples:

{% csrf_token %}

{% extends %}

{% if %} … {% endif %}

Practical

- Render data with an **{% if %} … {% endif %}** tag

Rendering data into a template

Part 3

© Arno Pretorius

Django template language - (DTL)...

- **Dictionary lookup**

```
{{ dictionary.key }}
```

- **List index lookup**

```
{{ list.2 }}
```

- **Attribute lookup**

```
{{ object.attribute }}
```

Lists and dictionaries

```
# - List with dictionaries

clientList = [

    {
        'id' : '1',
        'name' : 'John Smith',
        'profession' : "Web developer"


    },


    {
        'id' : '2',
        'name' : 'Sarah Evans',
        'profession' : 'Electrical engineer'

    },


]
```

clientList is our **list**

Dictionaries - each one is almost '**object-like**'

© Arno Pretorius

Practical

- Render data with a **{% for %} … {% endfor %}** tag

SECTION:

BUILDING A DATABASE

© Arno Pretorius

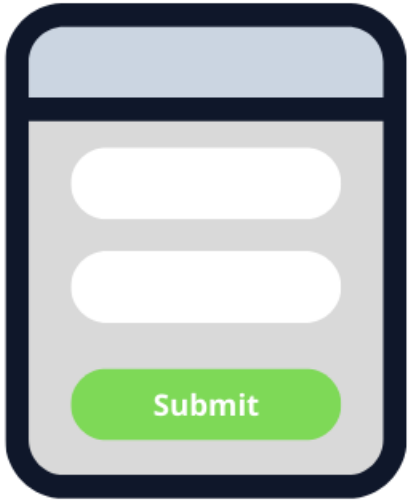
An introduction to databases

© Arno Pretorius

Database

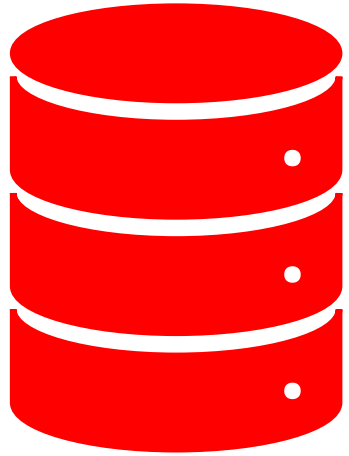
- A **database** is an organized collection of data that is accessed and stored electronically

How does a database work?



Online form

Post request

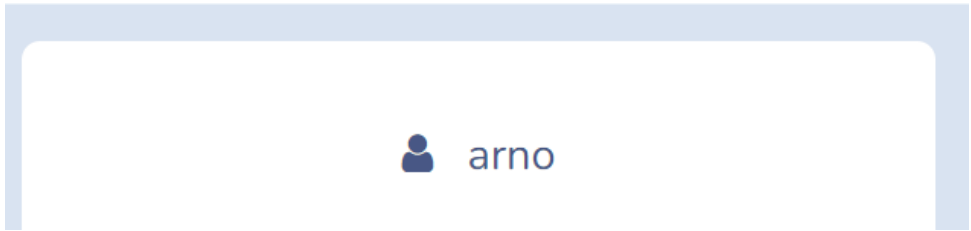


Database

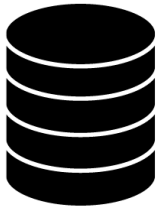
How does a database work?

In a nutshell…

Query



The types of databases

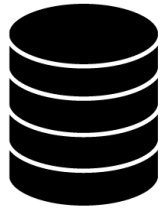


- There are two types of databases, namely:

- **Relational database**

- **Non-relational database**

© Arno Pretorius

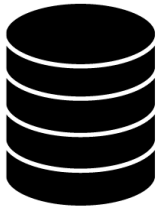


Relational database

- A relational database stores data in rows and columns like a spreadsheet
- **Uses SQL**

TASK
Id
task_name
task_description
employee_name

STUDENT
Id
student_name
student_grade
student_age



Examples of relational databases:

- Typical examples include:

- **PostgreSQL**
- **MySQL**
- **Oracle**

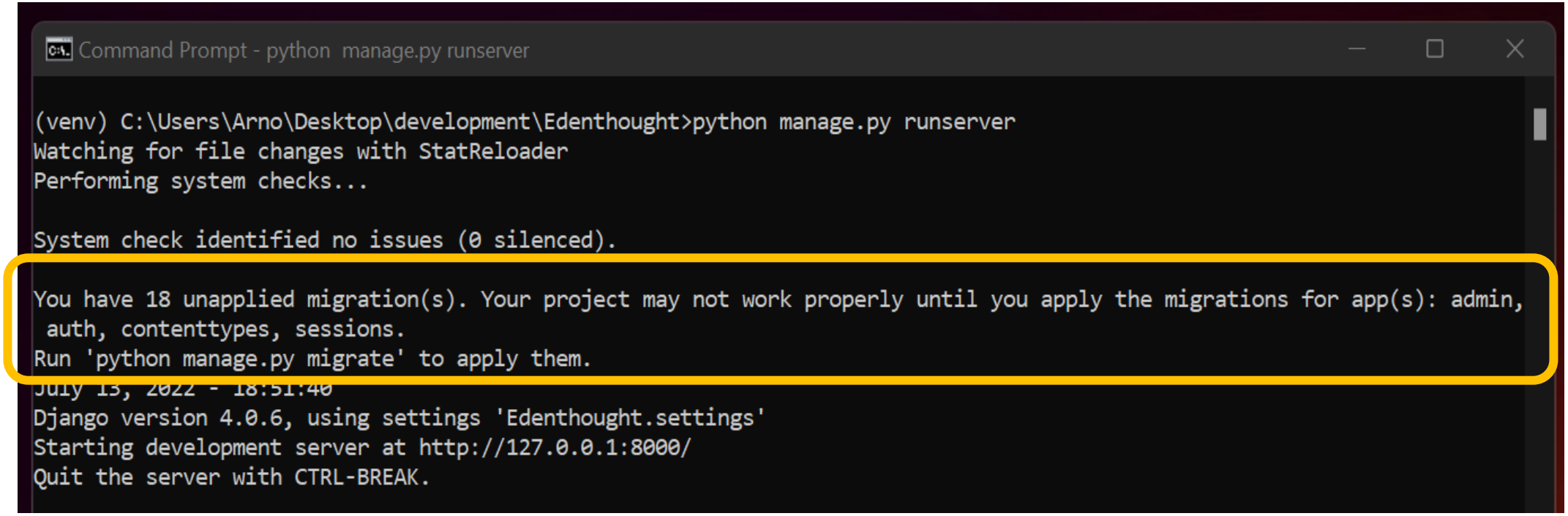
Practical

- Examine our database settings

Migrating our initial migrations

© Arno Pretorius

Remember this?



Practical

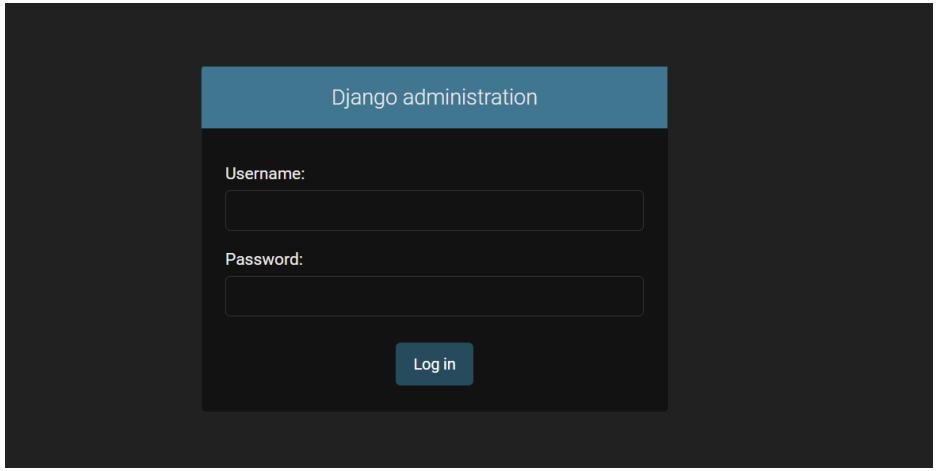
- Migrate our initial migrations

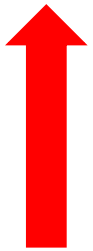
An introduction to our admin panel

© Arno Pretorius

Admin panel

- This is the administration page of our website
- Carry out basic admin tasks and operations
- Built-in with Django by default

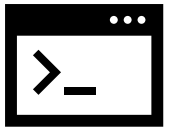




Access our admin panel

Accessing the admin panel

- **Create a superuser** using the following command:



```
python manage.py createsuperuser <super user name>
```

Practical

- Accessing our admin panel

An introduction to Django models

© Arno Pretorius

What is a Django model?

- A built-in feature used by Django to create tables along with their fields

© Arno Pretorius

What is a Django model?...

Django model

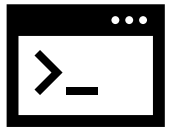
```python
from django.db import models

# Create your models here.

class Register(models.Model):

    name = models.CharField()
    id = models.UUIDField()
```

Database table

Register	
id	**name**
1	Jack
2	Sarah
3	Luke

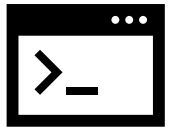
Create a migration

- To **make a migration** use the following command:



```
python manage.py makemigrations
```

- To **migrate** to our **database**, use the following command:



```
python manage.py migrate
```

Practical

- Create a model and migrate it to our database

Registering our models

© Arno Pretorius

Registering our models

- To access our database tables in our admin panel we need to register them first

Practical

- Register a model

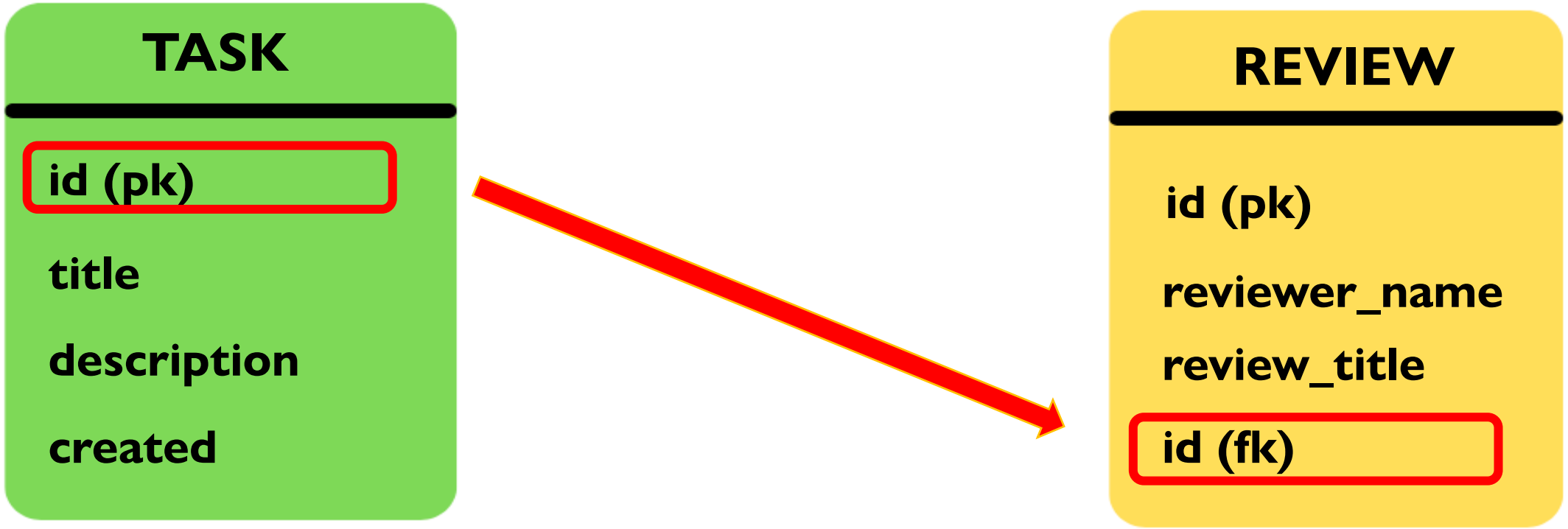
An introduction to foreign keys (FK)

© Arno Pretorius

What is a Foreign Key (FK)?

- *In simple terms* - It is a key that is used to **link two tables** together
- Can be used to **reference attributes from another table**

How does a Foreign Key work?



Practical

- Create a foreign key

Database queries

© Arno Pretorius

What is a database query?

- A query used to perform **an action or a selection** on a database table

- I.e., **Selecting** all the **records/objects** from a database table (**users**)
- I.e., **Updating** a record (first_name) from a database table (**users**)
- I.e., **Deleting** a record from a database table (**users**)

How do we query records in Django?

- Well enter….

Django ORM

Allows us to communicate with our database, and query data with Object Relational Mapping, without running **SQL commands**

Example: Querying data in Django

```
# Import the model that you would like to perform queries on

from .models import ModelName


# Select all the objects from a table

queryDataAll = ModelName.objects.all()


# Select a single object from a table

queryDataSingle = ModelName.objects.get(attribute='value')
```

Example: Querying data in Django...

```python
# Import the model that you would like to perform queries on

from .models import User


# Select all the objects from a table

queryDataAll = User.objects.all()


# Select a single object from a table

queryDataSingle = User.objects.get(username='john_678')
```

Practical

- Query data in Django

© Arno Pretorius

SECTION:

CRUD OPERATIONS AND MODEL FORMS

© Arno Pretorius

CRUD and model forms

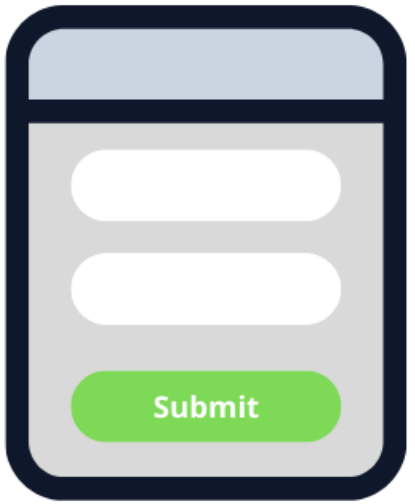
© Arno Pretorius

CRUD?

- Consists of a list operations that need to be performed on data - typically in our database
- CRUD - CREATE, READ, UPDATE, DELETE

Model forms

- A model form is a class that is used to transform a Django model into a form
- In-turn allows us to send data to a database and query data from a database



Example: Model forms

forms.py

```python
from django.forms import ModelForm
from . models import Task

class TaskForm(ModelForm):
    class Meta:
        model = Task
        fields = '__all__'
```

views.py

```python
from django.shortcuts import render
from django.http import HttpResponse

from .forms import TaskForm


def createTask(request):

    form = TaskForm()

    context = {'form': form}

    return render(request, 'task-form.html', context)
```

HTML template

```html
<form method="POST" autocomplete="off" enctype="multipart/form-data">


    {% csrf_token %}

    {{ form.title }}

    {{ form.description }}


    <input type="submit" value="submit">


</form>
```

Example: Model forms...

HTML form

```
<form method="POST" autocomplete="off" enctype="multipart/form-data">
    <label for="fname">First name:</label><br>
    <input type="text" name="fname" value="John"><br>
    <label for="lname">Last name:</label><br>
    <input type="text" name="lname" value="Doe"><br><br>
    <input type="submit" name="submit" value="submit">
</form>
```

Utilizing a model form in HTML

```
<form method="POST" autocomplete="off" enctype="multipart/form-data">
    {% csrf_token %}
    {{form.first_name}}
    {{form.last_name}}
    <input type="submit" name="submit" value="submit">
</form>
```

Practical

- Create a model form

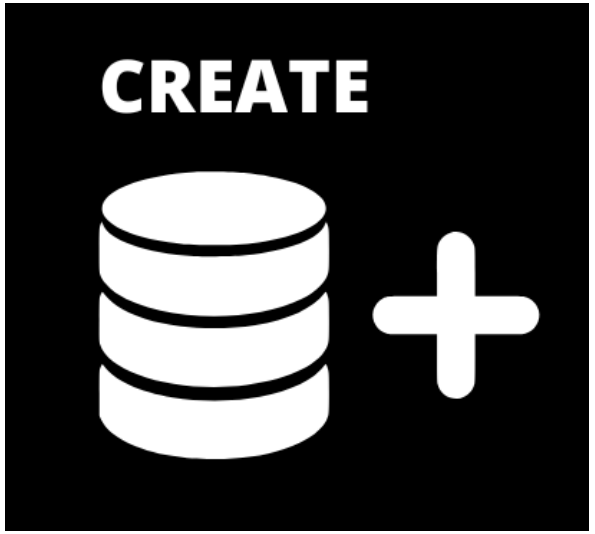
Model forms - CRUD

[CREATE]

© Arno Pretorius

Model forms: CRUD - [CREATE]

- **Create a new record** in our database table



Practical

- Create a record

Model forms - CRUD

[READ]

© Arno Pretorius

Model forms: CRUD - [READ]

- **Read a record** from our database table

Practical

- Read a record

© Arno Pretorius

Model forms - CRUD

[UPDATE]

© Arno Pretorius

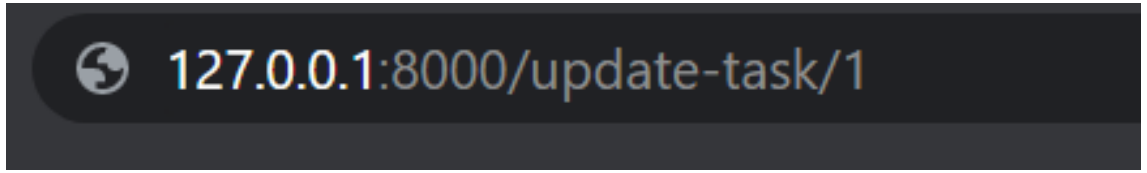
Model forms: CRUD - [UPDATE]

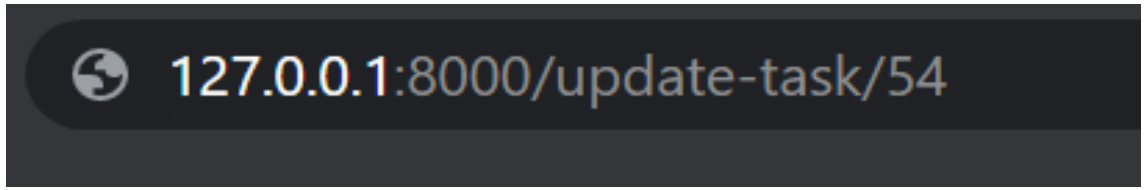
- **Update a record** in our database table

Model forms: CRUD - [UPDATE]...

```
path('update-task/<str:pk>', views.update_task,name="update-task")
```

Creating a dynamic URL





Practical

- Update a record

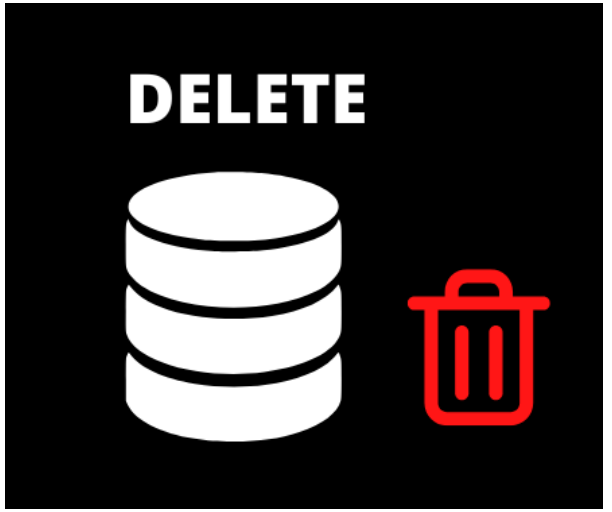
Model forms - CRUD

[DELETE]

© Arno Pretorius

Model forms: CRUD - [DELETE]

- **Delete a record** from our database table



Practical

- Delete a record

© Arno Pretorius

SECTION:

STATIC FILES

© Arno Pretorius

An introduction to static files

© Arno Pretorius

What are static files?

- Static files are files that don’t change when a web application is running
- They consist of **CSS**, **JavaScript** and **image files**

CSS (Cascading Style Sheets)

- **CSS** stands for **Cascading Style Sheets**
- Responsible for the **visual aspect** and **aesthetic** of a **web page**

JavaScript

- Responsible for **enriching the user experience** by making **web pages interactive**

Practical

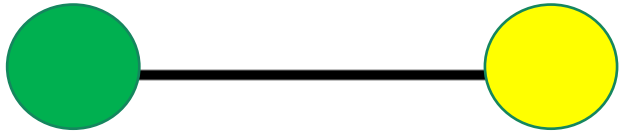
- Configure static files

© Arno Pretorius

Configure and connect - CSS

© Arno Pretorius

Configure and connect - CSS



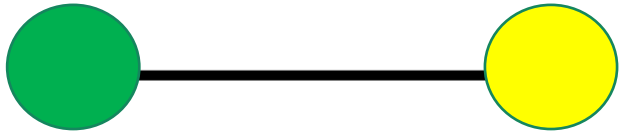
Practical

- Configure and connect - CSS

Configure and connect - JavaScript

© Arno Pretorius

Configure and connect - JavaScript



Practical

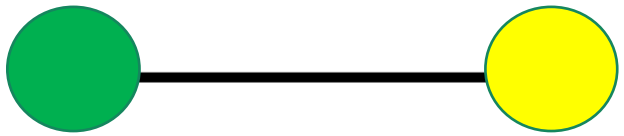
- Configure and connect - JavaScript

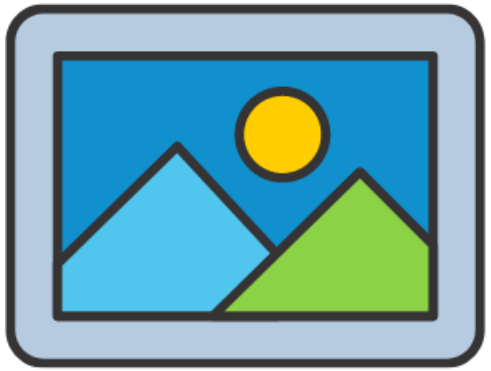
© Arno Pretorius

Configure and connect - Images

© Arno Pretorius

Configure and connect an image





Practical

- Configure and connect an image

SECTION:

MID-WAY REFLECTION

© Arno Pretorius

Mid-way reflection

© Arno Pretorius

© Arno Pretorius

SECTION:

USER CREATION AND AUTHENTICATION

© Arno Pretorius

Create a new user

© Arno Pretorius

Create a new user

- Create a new user and give them a username and a password

Practical

- Create a new user

An introduction to authentication

© Arno Pretorius

What is authentication?

- The process of proving that you are who you say you are

Practical

- Authenticate a user [login - logout]

How to protect a view

© Arno Pretorius

Why should we protect our views?

- To only give access to those that are allowed access
- Prevent unauthorized users from accessing certain views

Practical

- Protect a view

© Arno Pretorius

SECTION:

PRE-PROJECT (CLEAN-UP)

© Arno Pretorius

Pre-project

(cleanup)

© Arno Pretorius

Pre-project (cleanup)

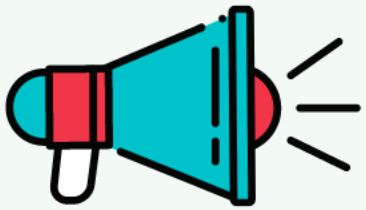
- To get **ready to build** our project we need to **clean up** and remove some files

Practical

- Prepare for our project

SECTION:

BUILDING OUR PROJECT – PART I



PROJECT

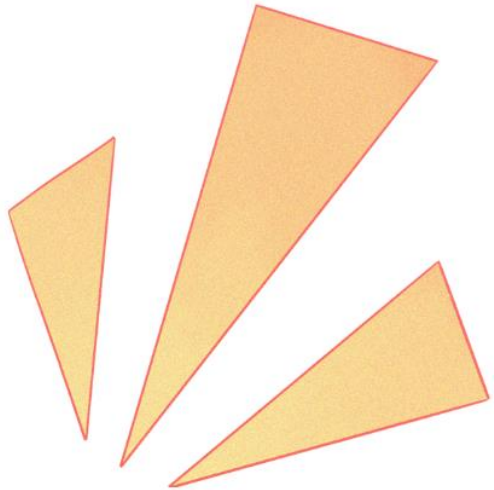
© Arno Pretorius

Re-structuring and styling our homepage

© Arno Pretorius

Re-structuring and styling our homepage

- Add some components to our homepage
- Style our home page and add a bootstrap theme - via bootswatch



Practical

- Re-structure and style our homepage

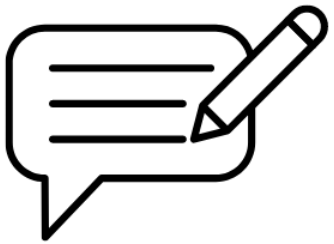
© Arno Pretorius

Registering new users

© Arno Pretorius

Register a new user

- Register a new user and give them a username and a password



Practical

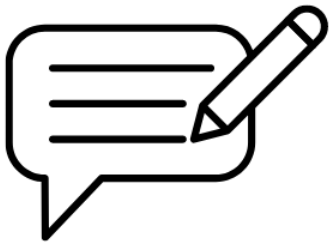
- Register a new user

Authenticating a user

© Arno Pretorius

What is authentication?

- The process of proving that you are who you say you are



Practical

- Authenticate a user [login - logout]

Utilizing Django messages

© Arno Pretorius

What are Django messages?

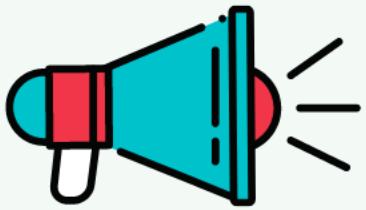
- Django messages/flash messages can be utilized to send us a notification of an event
- I.e., A new user has registered or if a user has logged in successfully to their account

Practical

- Create a flash message

SECTION:

BUILDING OUR PROJECT – PART II



PROJECT

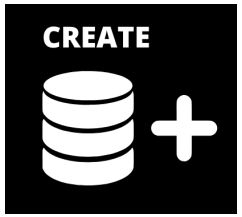
© Arno Pretorius

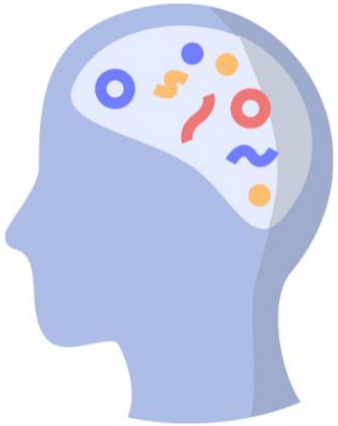
Create a thought

© Arno Pretorius

Create a thought

- We will utilize the create operation from CRUD to create a thought





Practical

- Create a thought

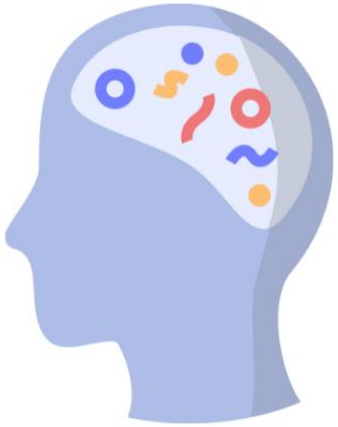
© Arno Pretorius

Read a thought

© Arno Pretorius

Read a thought

- We will utilize the read operation from CRUD to read a thought



Practical

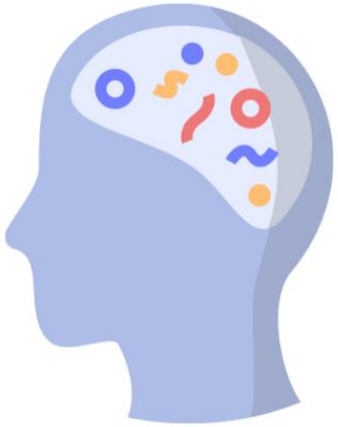
- Read a thought

Update a thought

© Arno Pretorius

Update a thought

- We will utilize the update operation from CRUD to update a thought



Practical

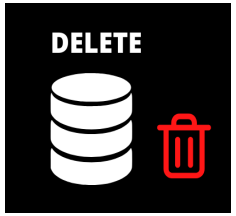
- Update a thought

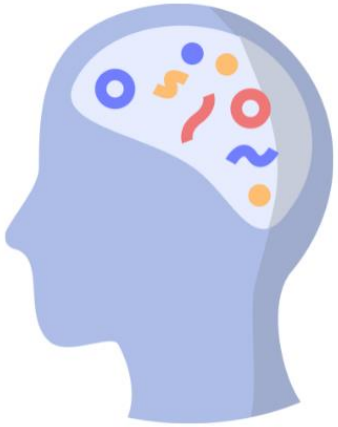
Delete a thought

© Arno Pretorius

Delete a thought

- We will utilize the delete operation from CRUD to delete a thought





© Arno Pretorius

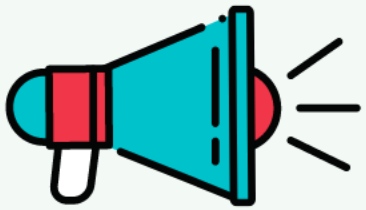
Practical

- Delete a thought

© Arno Pretorius

SECTION:

BUILDING OUR PROJECT – PART III



PROJECT

© Arno Pretorius

Profile management:

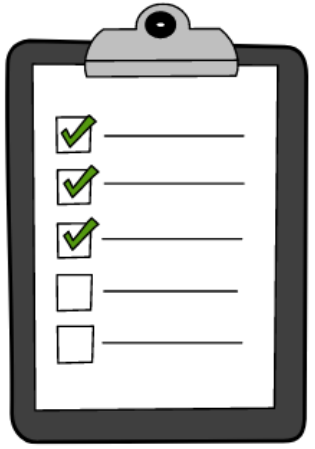
Updating our username and email

© Arno Pretorius

Profile management

- *Manage our profile in the following ways:*

- **Updating** our username and email
- **Deleting** our account
- **Updating** our profile picture



Practical

- Update our username and email

Profile management:

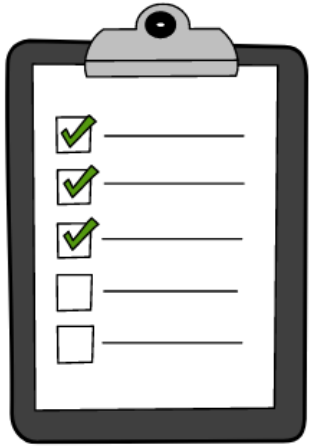
Deleting an account

© Arno Pretorius

Profile management

- *Manage our profile in the following ways:*

- **Updating** our username and email
- **Deleting** our account
- **Updating** our profile picture



Practical

- Delete an account

© Arno Pretorius

Uploading user content:

Configure Django to allow file uploads

© Arno Pretorius

Configure Django to allow file uploads

- Before we can upload a file in our project, we need to configure Django to allow file uploads

Practical

- Configure Django to allow file uploads

© Arno Pretorius

Uploading user content:

Upload and render a profile picture

© Arno Pretorius

Upload and render a profile picture

- Users who have created an account will be able to upload a profile picture and then see it on your dashboard

Practical

- Upload and render a profile picture

© Arno Pretorius

Profile management:

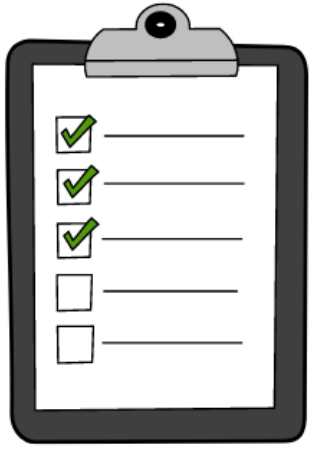
Updating our profile picture

© Arno Pretorius

Profile management

- *Manage our profile in the following ways:*

- **Updating** our username and email
- **Deleting** our account
- **Updating** our profile picture



Practical

- Update our profile picture

Utilizing Django messages with font awesome

© Arno Pretorius

Django messages + Font awesome

- We will use Django messages/flash messages to send us a notification of an all the events in our project and integrate it with font awesome icons
- I.e., A new user has registered or if a user has logged in successfully to their account

+

Practical

- Utilize Django messages with font awesome icons

© Arno Pretorius

Resetting a user's password

© Arno Pretorius

Resetting a user's password

- Users should be able to **reset their password**, in-case they have **forgotten it or if they want to change it**

Resetting a user's password...

- We will use **Django's built-in views** to handle our **password reset** views

- ***We will create four views:***

1 - Allow us to enter our email in order to receive a password reset link

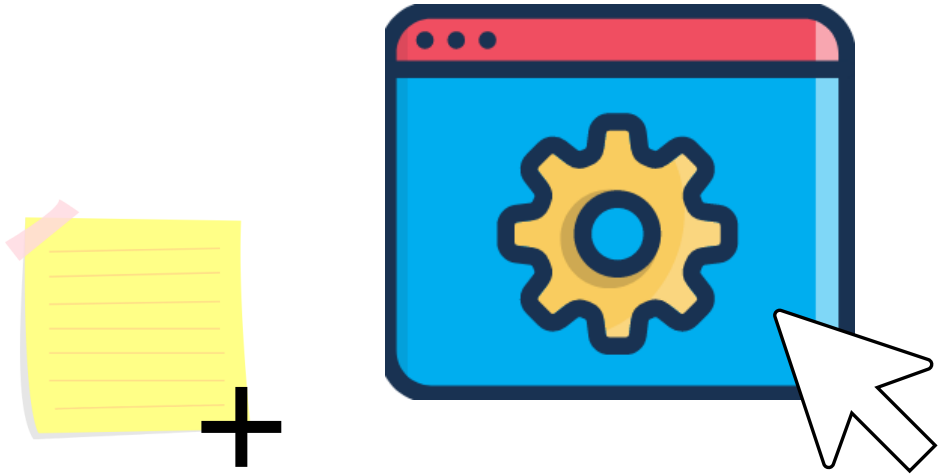
2 - Show a success message stating that an email was sent to reset our password

3 - Send a link to our email, so that we can reset our password

4 - Show a success message stating that our password was changed

Practical

- Reset a user's password



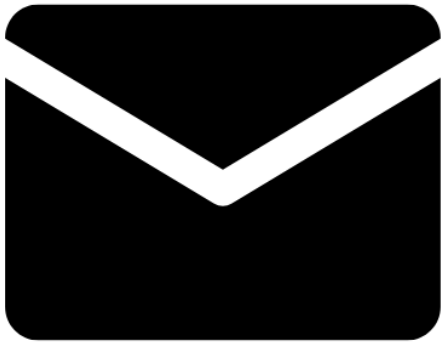
© Arno Pretorius

Sending a welcome email

© Arno Pretorius

Send a welcome email

- Once users have created an account, we can send them a welcome email, thanking them for signing up



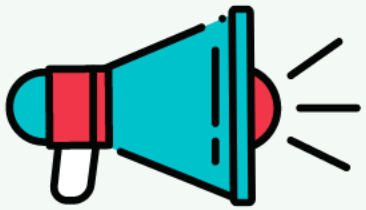
Practical

- Send a welcome email

© Arno Pretorius

SECTION:

DEPLOYMENT



PROJECT

© Arno Pretorius

Create an AWS account

© Arno Pretorius

Create an AWS account

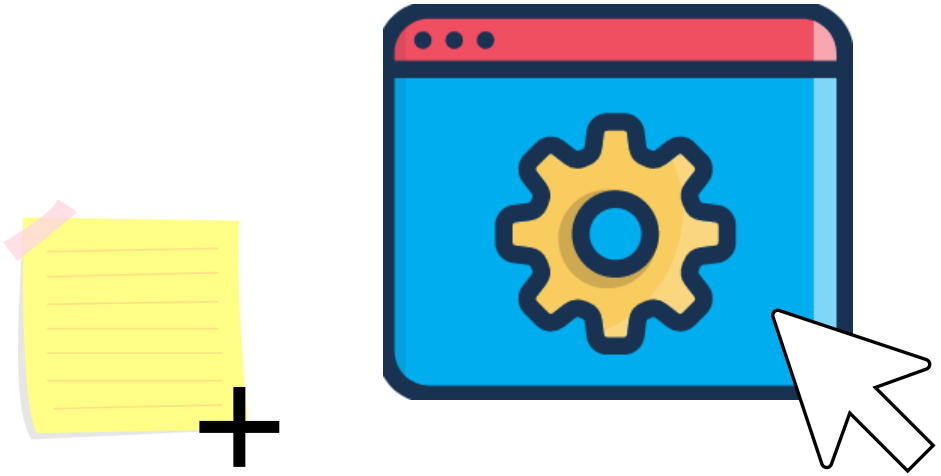
We will use AWS to:

- Create and connect a PostgreSQL database to our Django project
- Store our static files in Amazon S3 buckets

© Arno Pretorius

Practical

- Create an AWS account



© Arno Pretorius

AWS Identity and Access Management (IAM)

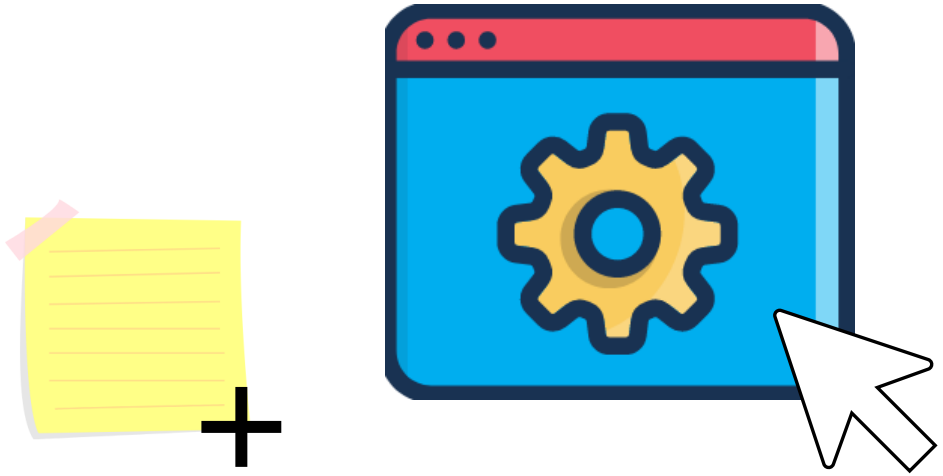
© Arno Pretorius

IAM - Overview

- IAM helps us to manage our users in AWS, and thereby allowing us to create user accounts
- It is good practice to rather create and use a user account than your root account
- It is also safer and more secure to use a user account

Practical

- Create an IAM user account on AWS



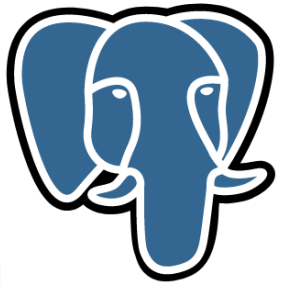
Amazon Relational Database Service (RDS)

© Arno Pretorius

RDS - Overview

- RDS is a managed **relational database service**
- With this service you will be able to create relational databases in AWS
- *Such databases include, but are not limited to:*

- *PostgreSQL*
- *MySQL*
- *Oracle*



Practical

- Create and connect a PostgreSQL database

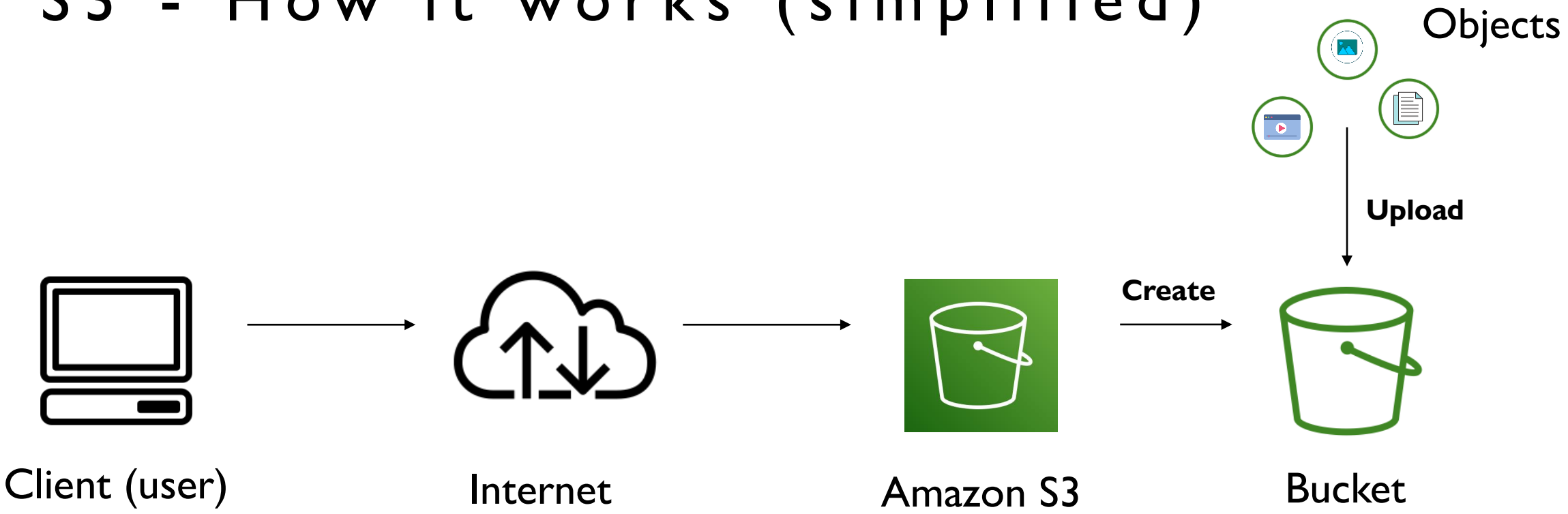
Amazon S3

© Arno Pretorius

S3 - Overview

- Amazon S3 allows you to store your **objects (files)** in **buckets (directories)**
- All bucket names must have a unique name
- You can use S3 to store any type of file that you want - commonly user images or static files
- One of the main building blocks of AWS
- Can be used for: backup and storage, hosting static websites, disaster recovery

S3 - How it works (simplified)



Practical

- Deploy our static files to an Amazon S3 bucket

Security tips

© Arno Pretorius

Security tips

- Before we deploy our project, we must make sure that **debug = False**
- Store all our important information, such as **AWS access keys**, **postgres database username and password**, **project secret key** in environment variables

Code and style cleanup

© Arno Pretorius

Code and style cleanup

- Add some more comments to our code
- Add an additional navigation bar
- Remove additional spaces in our code
- Implement extra functionality here and there…

Practical

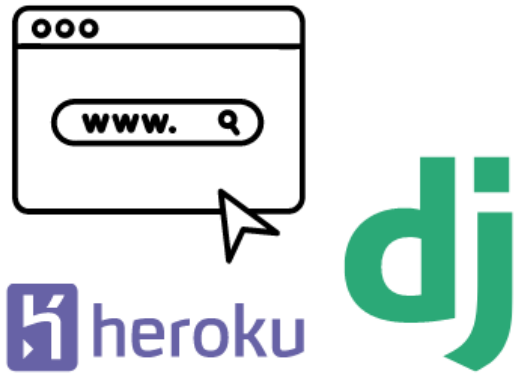
- Clean our code

Deploying our project to a live server

© Arno Pretorius

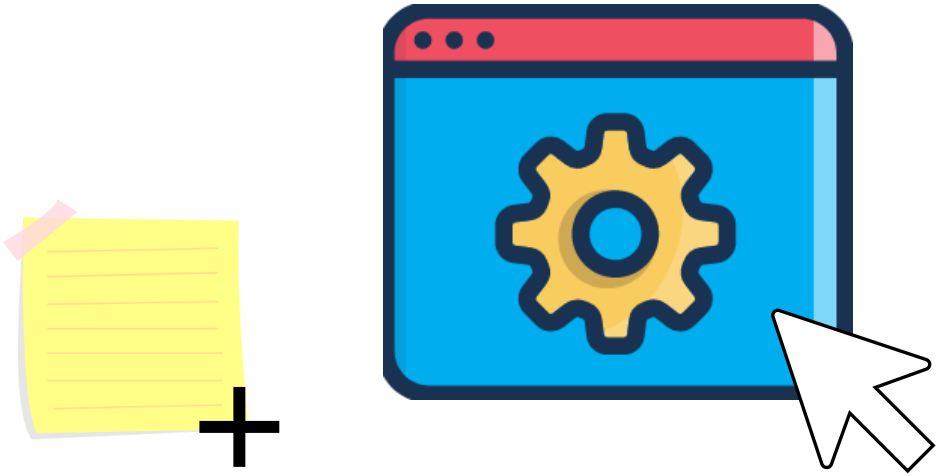
Deploy our project

- What we've all been waiting for…
- We will deploy our project on Heroku, and make it accessible online



Practical

- Deploy our project to Heroku



© Arno Pretorius

SECTION:

THANK YOU

© Arno Pretorius

© Arno Pretorius

Practical

- Keep on learning!

© Arno Pretorius