## Humanization:

Humanization in Django refers to the process of presenting data in a more human-friendly and understandable format. It involves converting raw data into a format that is easier for humans to interpret, comprehend, and relate to.

In the context of Django web development, humanization can be applied to various aspects such as:

1. **Date and Time Formatting**:
   Converting date and time values into more readable formats, such as displaying dates in "YYYY-MM-DD" format as "January 1, 2022" or showing timestamps like "5 minutes ago" instead of the raw timestamp value.

2. **Boolean Values**:
   Representing boolean values (True/False) in a more human-readable form, such as displaying "Yes" or "No" instead of "True" or "False".

3. **Numeric Values**:
   Formatting numeric values to include thousands separators, decimal points, and appropriate units (e.g., "1,000" instead of "1000" or "1.5 million" instead of "1500000").

4. **Textual Representation**:
   Presenting data in a way that is easier to understand, such as converting abbreviations or acronyms into their full names or providing context to numerical values.

5. **Localization and Internationalization**:
   Adapting the presentation of data to suit different languages, cultures, and regions by translating text labels, date formats, and other elements to the user's preferred language and locale.

Overall, humanization plays a crucial role in enhancing the user experience by making data more accessible, understandable, and relatable to users, thereby improving usability and user satisfaction in Django web applications.

**django.contrib.humanize:**

`django.contrib.humanize` is a built-in Django app that provides a set of template filters to perform humanization tasks on data. It offers functionality to make data more human-readable and user-friendly in Django templates without requiring additional coding.

Let's create a Django app named `humanize_example` to demonstrate the usage of `django.contrib.humanize`.

**Step 1: Create a Django Project and App**
First, create a Django project and an app within the project:

```
django-admin startproject humanize_project
cd humanize_project
python manage.py startapp humanize_example
```

**Step 2: Configure Installed Apps**
Add the `humanize_example` app and `django.contrib.humanize` to the `INSTALLED_APPS` list in your project's `settings.py` file:

```
INSTALLED_APPS = [
   ...
   'django.contrib.humanize',
   'humanize_example',
   ...
]
```

**Step 3: Create a Template to Demonstrate Humanization**
Create a template file named `humanize_example/templates/humanize_example/index.html`:

```
<!DOCTYPE html>
<html>
<head>
   <title>Humanize Example</title>
</head>
<body>
   <h1>Humanize Example</h1>
```

```
    <h2>Humanized Dates</h2>
    <p>Current Date: {{ current_date }}</p>
    <p>Humanized Date: {{ current_date|naturaltime }}</p>

    <h2>Humanized Numbers</h2>
    <p>Large Number: {{ large_number }}</p>
    <p>Humanized Large Number: {{ large_number|intcomma }}</p>
</body>
</html>
```

**Step 4: Create a View**
Create a view in `humanize_example/views.py` to render the template:

```
from django.shortcuts import render
from django.utils import timezone

def index(request):
    context = {
        'current_date': timezone.now(),
        'large_number': 1000000,
    }
    return render(request, 'humanize_example/index.html', context)
```

**Step 5: Configure URLs**
Configure URLs in `humanize_example/urls.py`:

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.index, name='index'),
]
```

**Step 6: Run the Server**
Run the Django development server:

```
python manage.py runserver
```

**Step 7: Visit the Page**
Open your web browser and visit `http://127.0.0.1:8000` to see the humanization examples in action.

**Conclusion:**

In this example, we've demonstrated how to use `django.contrib.humanize` to humanize dates and numbers in a Django template. The `naturaltime` filter is used to display a human-readable representation of a date, while the `intcomma` filter adds commas to large numbers to improve readability. This simplifies the presentation of data and enhances the user experience in Django web applications. The `django.contrib.humanize` module provides a set of template filters and tags to perform humanization tasks on data. Below is a list of available tags along with examples:

**1. `naturalday`**

Converts a date into a human-readable representation of the number of days ago, or the day name for dates within a week.

Example:

```html
{% load humanize %}

{{ some_date|naturalday }}
```

**2. `naturaltime`**

Converts a datetime into a human-readable representation of time, such as "just now", "yesterday", or "2 hours ago".

Example:

```html
{% load humanize %}

{{ some_datetime|naturaltime }}
```

**3. `intcomma`**

Adds commas to large numbers for better readability.

Example:

```html
{% load humanize %}

{{ large_number|intcomma }}
```

**4. `ordinal`**
Converts an integer into its ordinal representation (e.g., 1st, 2nd, 3rd).

Example:
```
{% load humanize %}

{{ ordinal_number|ordinal }}
```

**5. `apnumber`**
Converts an integer into its AP-style representation (e.g., "one", "two", "three").

Example:
```
{% load humanize %}

{{ number|apnumber }}
```

**6. `timesince`**
Converts a datetime into a human-readable representation of the time since that datetime.

Example:
```
{% load humanize %}

{{ some_datetime|timesince }}
```

**7. `timeuntil`**
Converts a datetime into a human-readable representation of the time until that datetime.

Example:
```
{% load humanize %}

{{ some_datetime|timeuntil }}
```

These tags provide convenient ways to humanize data in Django templates, making it easier for users to understand and interpret the information presented.