DJANGO TASK DAY – 3

1. What are the key features of the Django Template language, and how do they contribute to its advantages in web development?

   The **Django Template Language (DTL)** is a lightweight templating language used in Django, a popular web framework for Python. Some key features of DTL and how they contribute to its advantages in web development include:

   1) **Simple Syntax:**
      DTL has a simple and readable syntax, making it easy for developers to understand and use. This simplicity helps in quickly building templates without the need for complex logic.
      Syntax:
      The syntax of DTL is similar to that of other template languages. DTL uses double curly braces to denote variables and filters. For example:

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

      {% block content %}
        <h1>{{ title }}</h1>
        <p>{{ body|linebreaks }}</p>
      {% endblock %}
      ```

      In this example, we are extending a base template and defining a block named "content". Within the block, we are using two variables: "title" and "body". The "linebreaks" filter is used to convert line breaks into HTML line breaks.

   2) **Template Inheritance:**
      The most powerful and thus the most complex part of Django's template engine is template inheritance. Template inheritance allows you to build a base "skeleton" template that contains all the common elements of your site and defines blocks that child templates can override. This enables efficient code reuse and reduces redundancy.

   3) **Tags:**
      DTL provides a number of tags that allow you to control the flow of your template. Some of the most commonly used tags include:

      - {% if condition %}...{% endif %}: Allows you to conditionally render content based on a boolean value.
      - {% for item in list %}...{% endfor %}: Allows you to loop over a list or queryset.
      - {% block name %}...{% endblock %}: Defines a named block that can be overridden in a child template.

- {% include "template.html" %}: Includes another template within the current template.

4) **Filters:**
   Filters in DTL allow you to modify the output of a variable. Some of the most commonly used filters include:
   - {{ variable|default:"Default Value" }}: Sets a default value for a variable if it is not defined.
   - {{ variable|length }}: Returns the length of a list or string.
   - {{ variable|title }}: Capitalizes the first letter of each word in a string.
   - {{ variable|date:"D d M Y" }}: Formats a date according to a specified format.

5) **Built-in Functions:**
   DTL also provides several built-in functions that can be used within templates. Some of the most commonly used functions include:

   - url: Generates a URL based on a view name and arguments.
   - static: Generates a URL for a static file.
   - now: Returns the current date and time.
   - range: Generates a list of numbers based on a specified range.

6) **Variables:**
   DTL allows for the insertion and manipulation of variables within templates. Variables can be defined within the template context or passed from views to templates, enabling dynamic content rendering based on data retrieved from the backend. Variables look like this: {{ variable }}.

7) **Template Escaping:**
   DTL automatically escapes variables by default to prevent common security vulnerabilities such as cross-site scripting (XSS) attacks. Developers can mark specific content as safe using the safe filter when necessary, ensuring secure rendering of user-generated or potentially unsafe data.

8) **Comments:**
   DTL supports comments within templates, allowing developers to document template code or temporarily disable sections during development without affecting the template output. To comment-out part of a line in a template, use the comment syntax: {# #}.
   For example, this template would render as 'hello':

   {# Greeting #}Hello, Good Morning

Advantages of DTL in Web Development:

- **Improved Maintainability:** The clear separation of concerns between templates (presentation) and views (logic) fostered by DTL allows for easier maintenance.

Changes to the base template automatically propagate to child templates, reducing the need for repetitive modifications.

- **Enhanced Developer Productivity:** DTL's built-in constructs for common tasks like loops, conditionals, and data manipulation significantly reduce the amount of Python code required within templates. This translates to faster development cycles and improved efficiency.

- **Promotes Secure Development:** By segregating template logic from Python code, DTL mitigates the potential for code injection vulnerabilities. This security benefit is crucial for safeguarding web applications.

- **Increased Readability and Collaboration:** DTL leverages a syntax that closely resembles HTML, making templates more understandable for designers and front-end developers. This fosters better collaboration between teams working on the same project.

2. Create a Django view using both function-based and class-based approaches. Utilize different Django Template Language (DTL) tags within these views. (attach screenshots of SITE).