**Module – 5 (DB and Python Framework)**

1) Why Django should be used for web-development? Explain how you  
can create a project in Django

Ans : Django is a powerful web framework for building web applications quickly and efficiently. Here are some reasons why Django is often chosen for web development:

1. **Batteries-included**: Django comes with a lot of built-in features, like an admin panel, authentication system, and ORM (Object-Relational Mapping) for database interaction. This saves time and effort in setting up basic functionalities.
2. **Scalability**: Django's design allows developers to scale applications easily from small to large projects. It provides tools and patterns to handle growth in traffic and complexity.
3. **Security**: Django helps developers avoid common security mistakes by providing built-in protections against CSRF (Cross-Site Request Forgery), SQL injection, clickjacking, and more.
4. **Versatility**: It supports various databases (MySQL, PostgreSQL, SQLite, etc.) and can integrate with different technologies through its extensive ecosystem of libraries and extensions.
5. **Documentation and Community**: Django has excellent documentation and a large community of developers. This means you can find help, tutorials, and plugins easily.

Creating a project in Django involves a few steps:

1. \*\*Install Django\*\*: Use `pip install django` to install Django on your computer.

2. \*\*Create a Project\*\*: Run `django-admin startproject projectname` to start a new Django project.

3. \*\*Start the Server\*\*: Navigate to your project folder and run `python manage.py runserver` to start the development server.

4. \*\*Create an App\*\*: Use `python manage.py startapp appname` to create a new app within your project.

5. \*\*Define Models\*\*: Edit `models.py` in your app folder to define your data models using Python classes.

6. \*\*Migrate Database\*\*: Run `python manage.py makemigrations` and `python manage.py migrate` to create database tables from your models.

7. \*\*Write Views and Templates\*\*: Define views in `views.py` and create HTML templates in a `templates` folder in your app directory.

8. \*\*URL Routing\*\*: Map URLs to views by editing `urls.py` files in your project and app folders.

9. \*\*Admin Panel\*\*: Access Django's built-in admin panel to manage your app's data.

10. \*\*Deploy\*\*: When ready, deploy your Django app to a production server.

2) How to check installed version of django?

Ans : To check the installed version of Django on your system, you can use either of the following methods:

1. \*\*Using Command-Line Interface (CLI)\*\*:

Open your terminal or command prompt and type:

```

python -m django --version

```

This command will display the installed version of Django.

2. \*\*Using Python Interpreter\*\*:

You can also check the Django version directly from within a Python script or interpreter:

```python

import django

print(django.get\_version())

```

3) Explain what does django-admin.py make messages command is used  
for?

Ans : The `django-admin.py makemessages` command is used to find translatable text in your Django project and generate message files (`.po` files) for translation. This helps in internationalizing your app by preparing it for multiple languages.

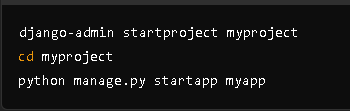
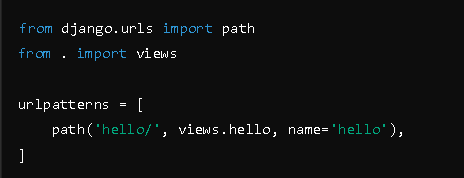
4) What is Django URLs?make program to create django urls

Ans: Django URLs (Uniform Resource Locators) are used to map the paths in a web application to specific views. Each URL pattern corresponds to a view function or class that processes the request and returns a response.

Here's a simple guide to creating Django URLs:

**1) Create a Django Project and App**:

**2) Define a View in Your App**: Open myapp/views.py and add a simple view function:

3) **Create URL Patterns in Your App**: Create a file myapp/urls.py and define URL patterns:

4) **Include App URLs in the Project**: Open myproject/urls.py and include the app's URL patterns:

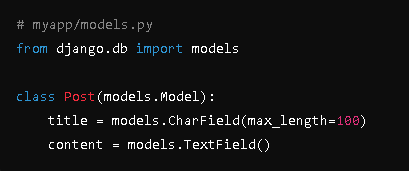
5) **Run the Server**: Start the development server

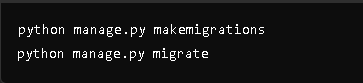
5) What is a QuerySet? Write program to create a new Post object in  
database

Ans: A **QuerySet** in Django is a collection of database queries to retrieve, filter, and manipulate data from your database. It's an abstraction that allows you to interact with the database in a Pythonic way.

Here’s how you can create a new Post object in a Django database:

1. **Define the Post Model**: First, define a Post model in your models.py file.



 **2. Create and Apply Migrations**: Make and apply the migrations to create the Post table in your database.

**3**. **Create a New Post Object**: Now you can create a new Post object. This can be done in the Django shell or in a view.

6) Mention what command line can be used to load data into Django?

Ans: To load data into a Django application, you can use the `loaddata` command. This command is typically used to load data from fixture files (usually in JSON, XML, or YAML format) into your database.

Here’s the command to load data:

```bash

python manage.py loaddata <fixture\_filename>

```

For example, if you have a JSON fixture file named `initial\_data.json`, you would use:

```bash

python manage.py loaddata initial\_data.json

```

7) Explain what does django-admin.py make messages command is used  
for?

Ans : The `django-admin.py makemessages` command is used to extract translatable strings from your Django project and create `.po` (Portable Object) files. These files are used for translating your application into different languages. Here's the basic usage:

1. \*\*Mark strings for translation\*\*: Use `gettext` or `{% trans %}` in your code and templates.

2. \*\*Run the command\*\*:

```bash

django-admin.py makemessages -l <language\_code>

```

For example, to create message files for French:

```bash

django-admin.py makemessages -l fr

```