setting functons for urls:

A function that takes a prefix, and an arbitrary number of URL patterns, and returns a list of URL patterns in the format Django needs.

Create a Django App:

First, make a Django app. If not, create one using the following command:

python manage.py startapp your_appname

Define Views:

Create views in your views.py file within your app.Views are functions that handle HTTP request and return Http response.

```
# views.py

from django.http import HttpResponse


def home(request):
    return HttpResponse("Hello, this is the home page!")


def about(request):
    return HttpResponse("This is the about page.")
```

Create URL's:

In your apps urls.py file,define URL patterns and associate thwm with the corresponding view function.

```
# yourappname/urls.py

from django.urls import path

from . import views


urlpatterns = [
    path('', views.home, name='home'),
    path('about/', views.about, name='about'),
]
```

Include App URLs in Project URLs:

Include your apps's URLs in the projects 'urls.py file

```
# yourprojectname/urls.py
```

```
from django.contrib import admin

from django.urls import path, include


urlpatterns = [

   path('admin/', admin.site.urls),

   path('yourappname/', include('yourappname.urls')),

]
```

To understand the basics of Python Django and understanding the rules?

Django is a Python framework that makes it easier to create web sites using Python.

Django takes care of the difficult stuff so that you can concentrate on building your web applications.

Django emphasizes reusability of components, also referred to as DRY (Don't Repeat Yourself), and comes with ready-to-use features like login system, database connection and CRUD operations (Create Read Update Delete).

Django is especially helpful for database driven websites

Django follows the MVT design pattern (Model View Template).

- Model - The data you want to present, usually data from a database.
- View - A request handler that returns the relevant template and content - based on the request from the user.
- Template - A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.

Model

The model provides data from the database.

In Django, the data is delivered as an Object Relational Mapping (ORM), which is a technique designed to make it easier to work with databases.

The most common way to extract data from a database is SQL. One problem with SQL is that you have to have a pretty good understanding of the database structure to be able to work with it.

Django, with ORM, makes it easier to communicate with the database, without having to write complex SQL statements.

The models are usually located in a file called `models.py`.

View

A view is a function or method that takes http requests as arguments, imports the relevant model(s), and finds out what data to send to the template, and returns the final result.

The views are usually located in a file called `views.py`.

Template

A template is a file where you describe how the result should be represented.

Templates are often .html files, with HTML code describing the layout of a web page, but it can also be in other file formats to present other results, but we will concentrate on .html files.

Django uses standard HTML to describe the layout, but uses Django tags to add logic:

```
<h1>My Homepage</h1>



<p>My name is {{ firstname }}.</p>
```

The templates of an application is located in a folder named `templates`.

URLs

Django also provides a way to navigate around the different pages in a website.When a user requests a URL, Django decides which *view* it will send it to.

This is done in a file called `urls.py`.

Http request and response (get,post)

HTTP

The Hypertext Transfer Protocol (HTTP) is designed to enable communications between clients and servers.

HTTP works as a request-response protocol between a client and server.

Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

HTTP Methods

- **GET**
- **POST**

The two most common HTTP methods are: GET and POST.

GET Method

GET is used to request data from a specified resource.

Note that the query string (name/value pairs) is sent in the URL of a GET request:

```
/test/demo_form.php?name1=value1&name2=value2
```

Some notes on GET requests:

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests are only used to request data (not modify)

POST Method

POST is used to send data to a server to create/update a resource.

The data sent to the server with POST is stored in the request body of the HTTP request:

```
POST /test/demo_form.php HTTP/1.1
Host: w3schools.com

name1=value1&name2=value2
```

notes on POST requests:

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length