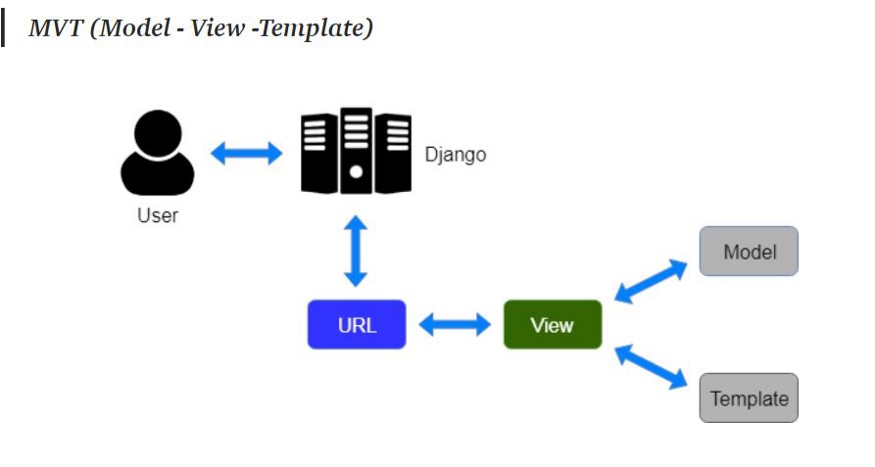
**Django Framework — DTL AND COOKIES**

**INTRODUCTION**

Django Framework is an extremely popular and fully featured server-side web framework. It is a Python-based free and open-source web framework that follows the model–view–template architectural pattern.



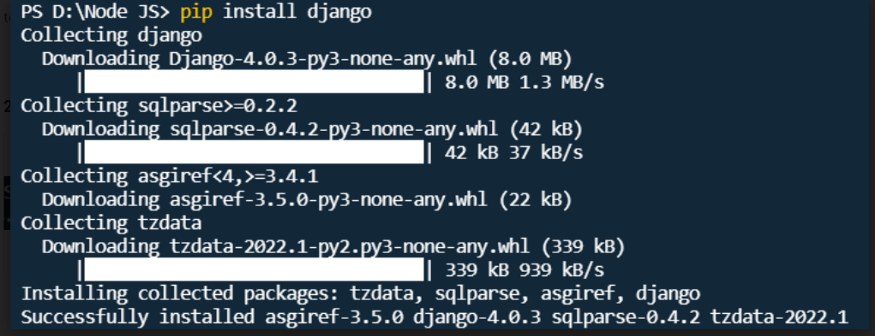
**BASIC SET-UP**

ð Installation of Django:

1. We need to install Django before going further.

*2. Write the Command in the terminal/shell —*

“pip install Django”



Installation Process

ð *Checking a version:*

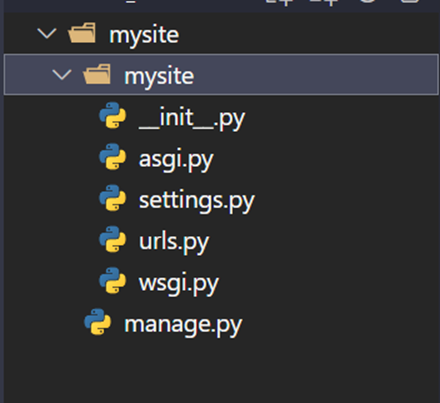
**python –m django –version**

ð *Creating a*Project*:*

1. From the command line, **cd** into a directory where you’d like to store your code, then run the following command:

**django-admin startproject mysite**

3. Let’s look at what ***startproject*** created:



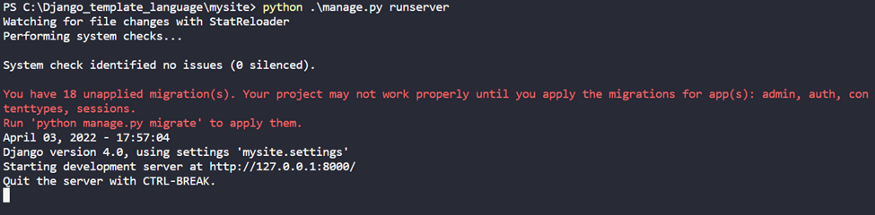
These are the files that will be created!

*The Development Server:*

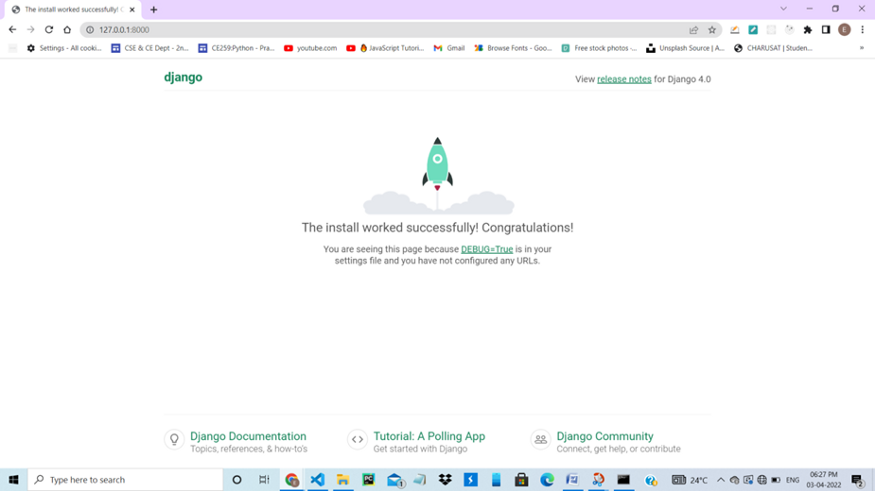
1. Let’s verify your Django project works. Change into the outer **mysite** directory, if you haven’t already, and run the following commands:

***Python manage.py runserver***

2. You’ll see the following output on the command line:



3. Now that the server’s running, visit <http://127.0.0.1:8000/> with your web browser. You’ll see a “Congratulations!” page, with a rocket taking off. It worked!

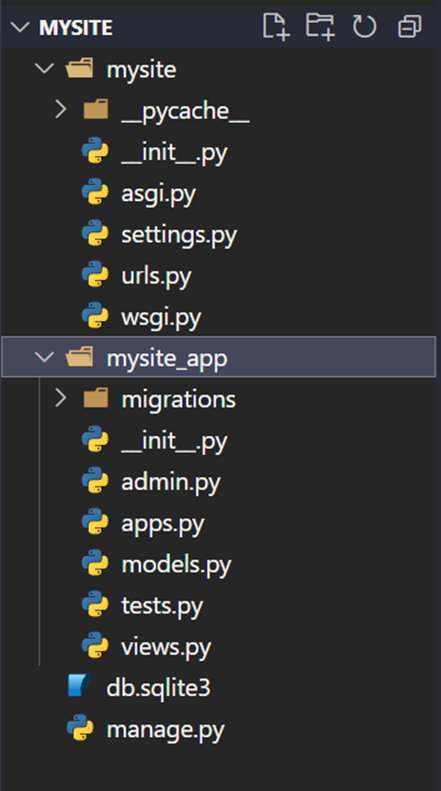


ð *Create the app:*

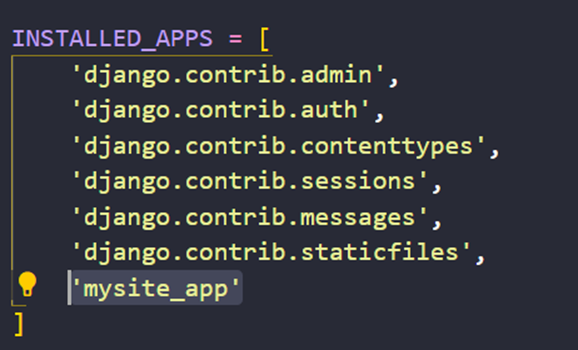
1. To create your app, make sure you’re in the same directory as **manage.py** and type this command:

python manage.py startapp mysite\_app

2. That’ll create a directory **mysite\_app**, which is laid out like this:



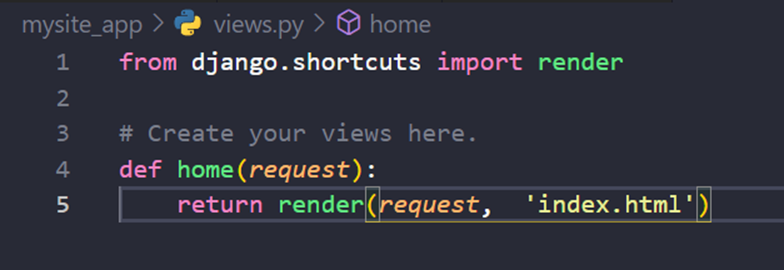
3. After Creating Application ; first of all we install it in settings.py file inside INSTALLED\_APPS list:



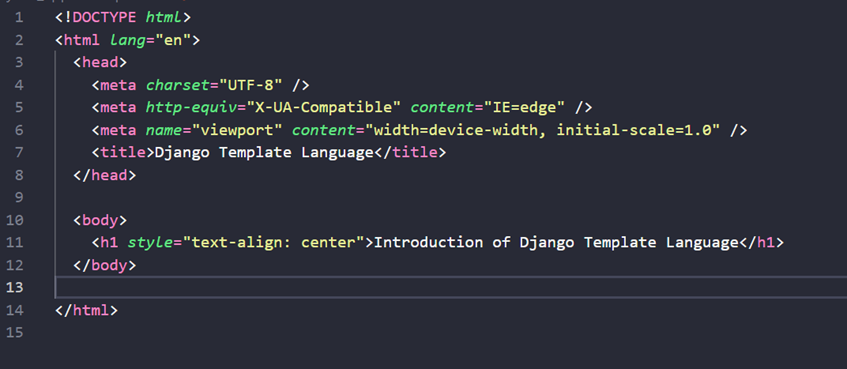
**ð *Basic Setup of Running Html file in Django:***

1.*Now; First of all making a templates Directory in the application and inside it make a***index.html** *file.*

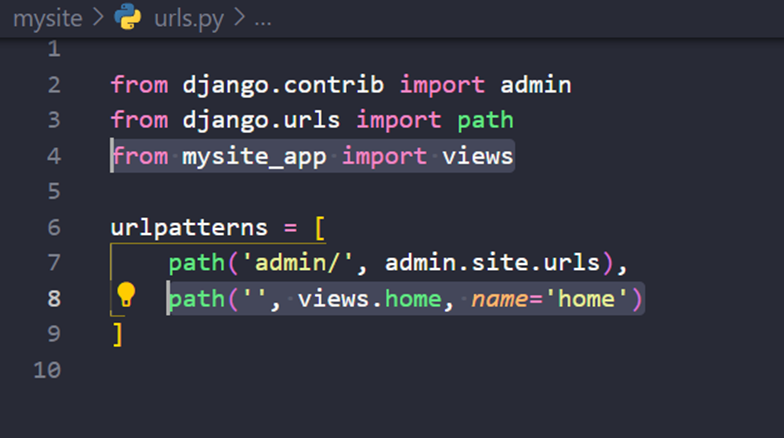
2.*Now create one function home() in views.py file which redirect us to***index.html***file.*



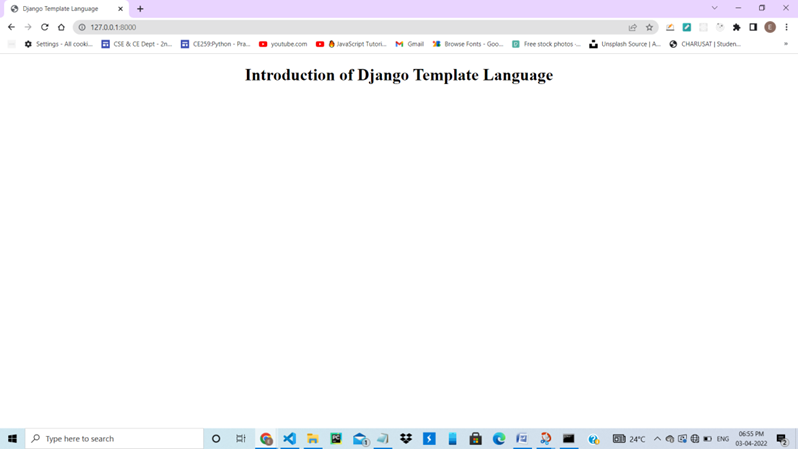
3.*Now write some code in***index.html***file.*



4.*Now, Create one path in urls.py file which connect to home() function which is in views.py file.*



5.*After that write a command in terminal python manage.py runserver and press enter and open this link*[*http://127.0.0.1:8000*](http://127.0.0.1:8000/)*in your Browser.*



**ð *Introduction of Django Template Language:***

1. **Django Template Language or DTL** is a text-based Template language that provides a bridge between scripts like HTML, CSS, JS, etc. and programming languages like python.

2. **DTL**is specifically built for developers to embed Django logic codes into HTML template files.

3. **DTL**also has a great advantage over other text-based template languages because of its

· Simplicity

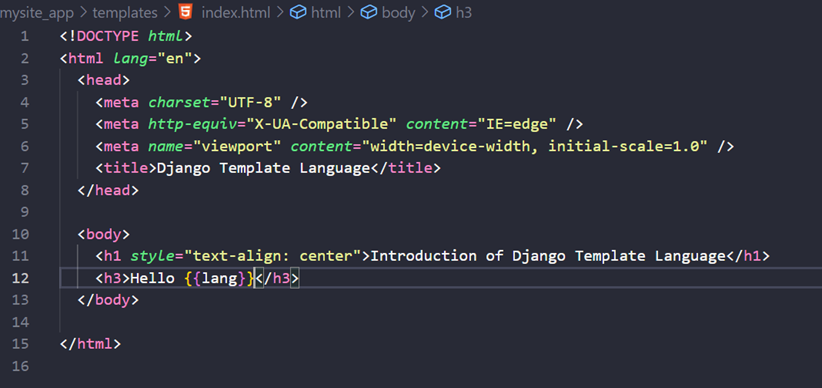
· Easy to learn this syntax

ð *Why we need the Django Template Language*:

1. A web application has two major components:

* **FRONT-END**
* **BACKE-END**

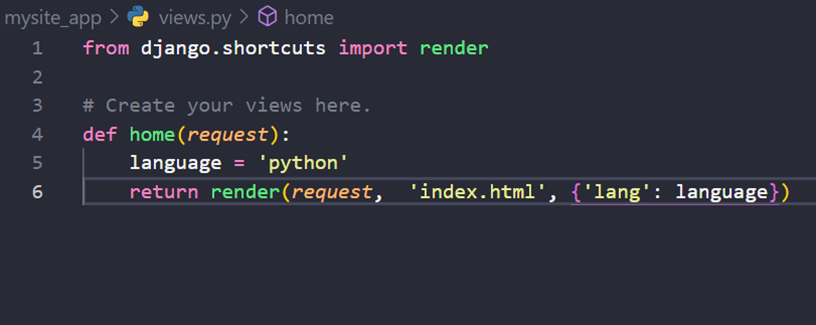
2. Therefore, it makes a lot more sense if the **Front-end developers** work separately on the HTML part while the **Back-end developers** work separately on the Python-Django part.



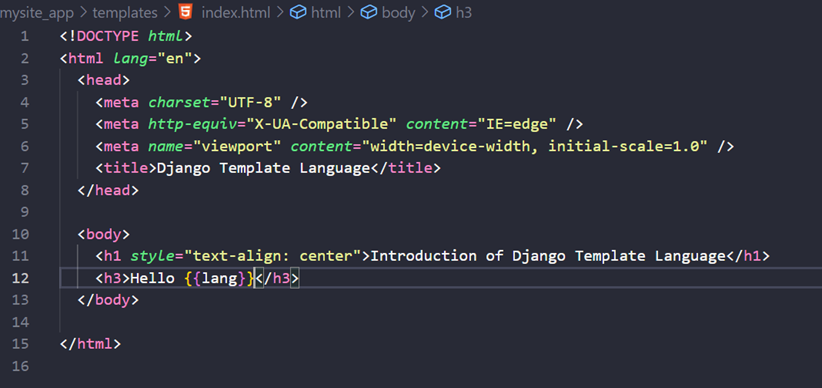
3. A front-end guy can work only on the HTML and leave HTML comments wherever he requires information from Django. Later a back-end guy will replace the HTML comments with the DTL syntaxes and hence won’t need HTML knowledge.

ð *Example of Django Template Language:*

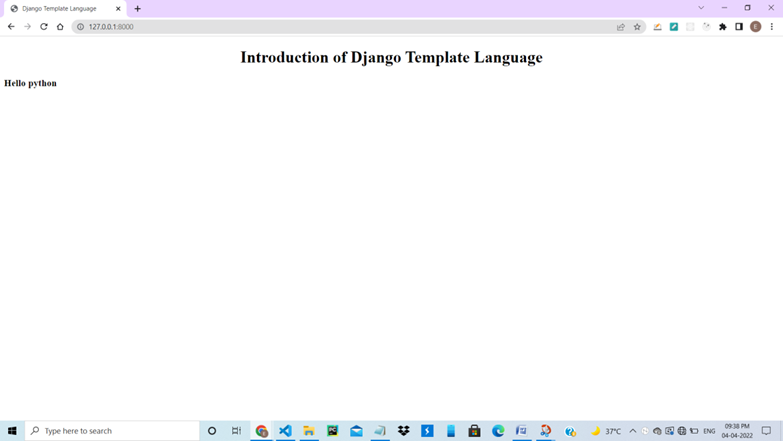
1. Now we create one variable and pass it into index.html file as a dictionary.



2. And Now I access this variable using key of the dictionary in index.html file. Ex. {{lang}}



3. And after that if I run the server using python manage.py runserver then the value of lang key is visible.



**ð *Basic Structure Of Django Template Language:***

1. The DTL syntax is very similar to Python. It consists of:

* Template Tags
* Template Variables
* Template Filters
* Template Comments
* We will now look into each one of them individually.

**1.** ***Template Tags*:**

o They **“do”**something. The Template Tag syntax:

o **Ex**:

{% Tag %}

o Template Tags itself is of **5**different types:

**1.1 Conditional Statements:**

o These, similar to the [**conditional statements**](https://www.askpython.com/python/python-if-else-elif-statement)in Python, are to used to execute logic.

o **Ex**:

{% if %}

<code>

{% end if %}

**1.2** **Loops:**

o This, similar to the [**python loop**](https://www.askpython.com/python/python-loops-in-python), is used to iterate variables in a loop.

o **Ex**:

{% for x in y %}

<code>

{% endfor %}

**1.3** **Block Declarations:**

o Block declarations are mainly used in **Template Inheritance**

o **Example**:

{% block content %}

<code>

{% endblock %}

**1.4** **File Inclusions:**

o This command includes other HTML files into the present file.

o **Ex**: {% include “<filename>” %}

**1.5** **File Inheritance:**

o This command includes other HTML files into the present file.

o **Example**:

{% extends “<filename>” %}

**2.** ***Template Variables*:**

o Template Variables in DTL functions similar to variables in Python. The syntax:

o Ex:

{{ varable\_name }}

o Some of the template variable examples are given below:

* *Simple Variable***:**{{ title }} , {{ x }}
* *List attributes:* {{ fruits\_list.0 }}
* *Dictionary attribute:* {{ dict.key }}

The data for these variables is pulled directly from the Python code and values can be implemented in the HTML code by using the above syntax.

**3.** **Template Filter:**

o Template filter is used to perform **filtering** on the template variables. The syntax for Template filters:

o Ex: {{ <Variable\_Name> | <filter\_by\_attribute> }}

o Some of the most used examples of Template Filters are:

* **Change cases:**{{ name|title }}, {{character|uppercase }}
* **List filters/slicing:** {{ list|slice=”:5” }}
* **Truncation:** {{ name|truncatewords:80 }}
* **Default:** {{ value|default = “0” }}

**4.** **Template Comments:**

o As the name suggests, this the DTL equivalent of the python comments. The Template Comment syntax:

o **Ex:**

o Just like in python, the code present inside the comment attribute will not be executed by the console.

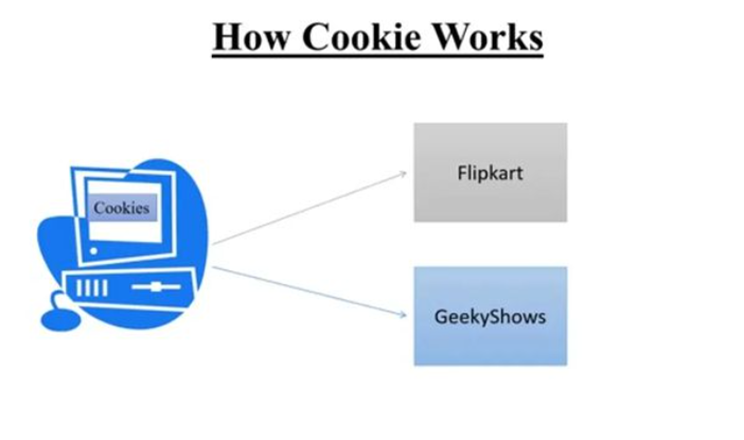
ð **Conclusion*:-***

**This was all about the Django Template Language. With this, We can efficiently link HTML codes with Python**.

ð **COOKIES*:-***

* A cookie is a small piece of information which is stored in the client browser. It is used to store user’s data in a file permanently (or for the specified time).
* Cookie has its expiry date and time and removes automatically when gets expire.
* **COOKIES — A Dictionary contains all cookies. Keys and Values are strings.**

Example: ***Add to Cart*** feature in Amazon.



ð **Explanation of Cookies in Django:**

1. First of all; I create one another project and inside that I create one application and installed this application in settings.py . All the Source code are given upper.

Now We will create a function in views.py which set a cookie in client machine

def setcookie(request):  
 response = render(request, ‘student/setcookie.html’)  
 response.set\_cookie(‘name’, ‘Hello World’)  
 return response

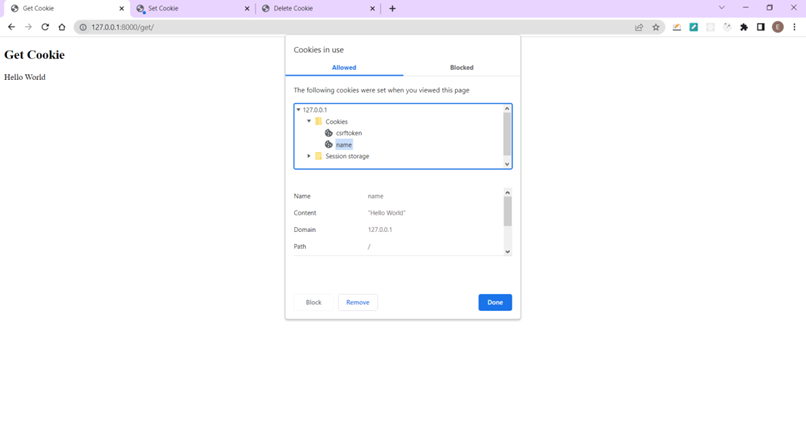
2. Now, We can create another function in views.py which will fetch the value of the cookie.

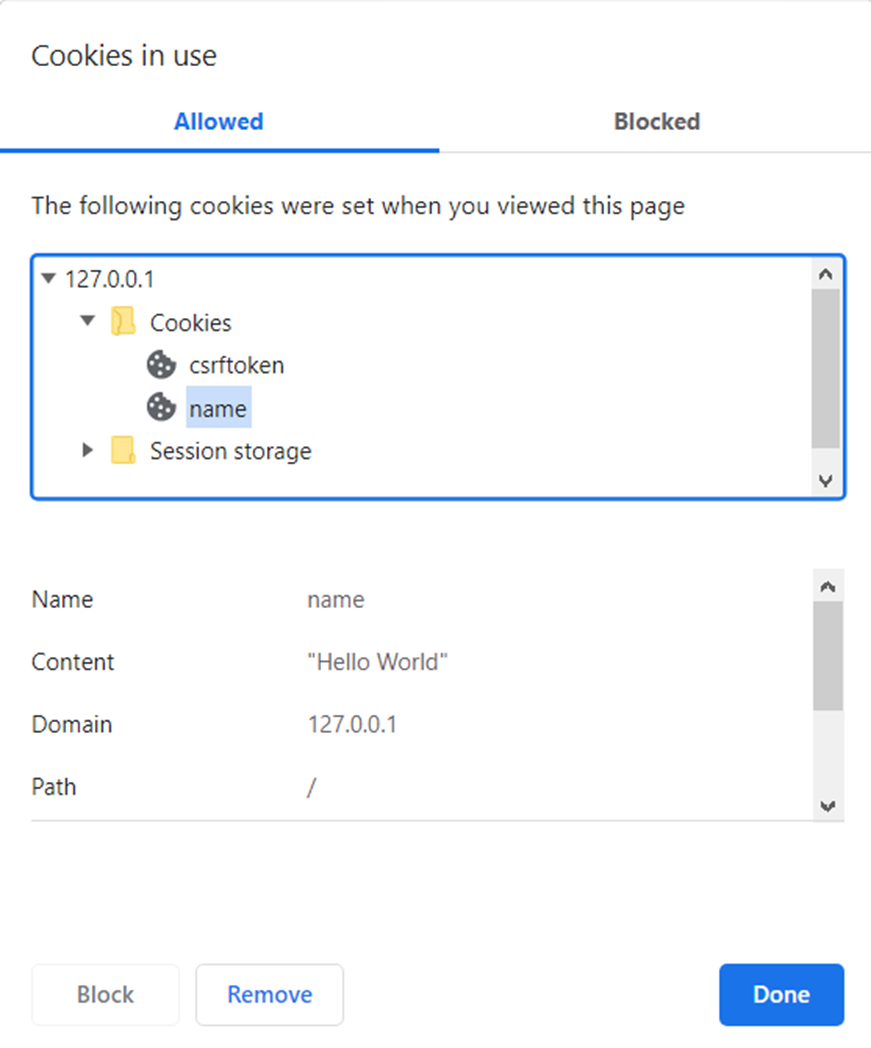
def getcookie(request):  
 name = request.COOKIES.get(‘name’)  
 return render(request, ‘student/getcookie.html’, {‘name’: name})

3. Function to delete the cookie in the client machine.

def delcookie(request):  
 response = render(request, ‘student/delcookie.html’)  
 response.delete\_cookie(‘name’)  
 return response

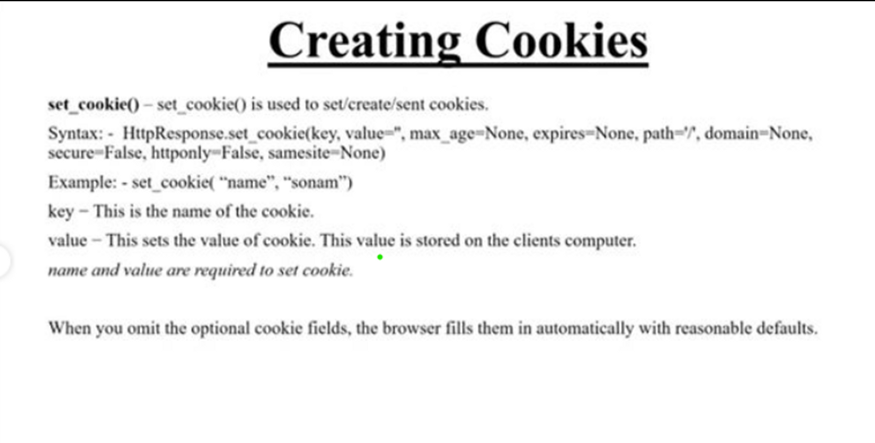
Now, after running the sever we will be able to see the cookie in the User’s machine.

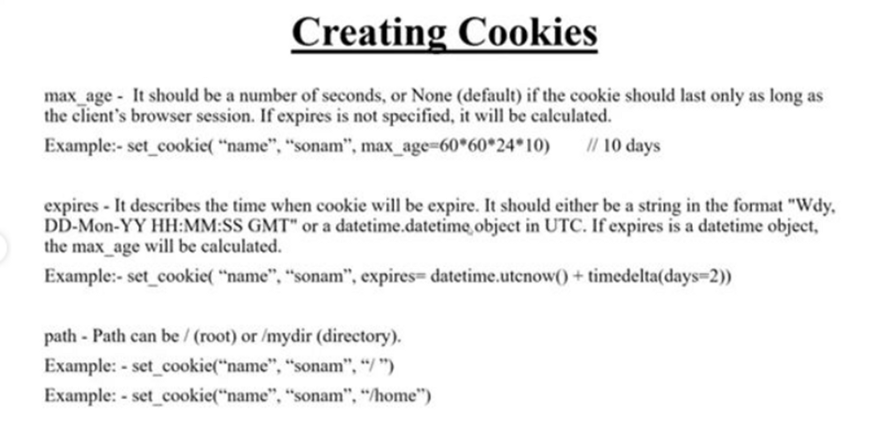


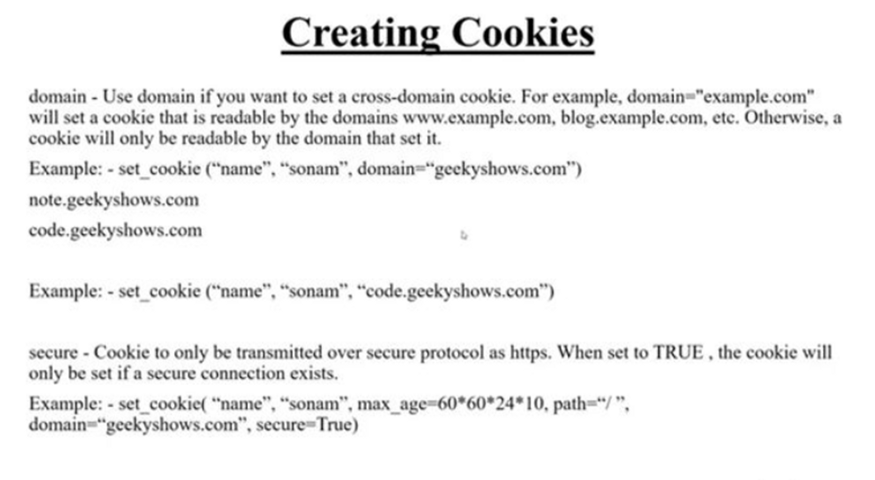


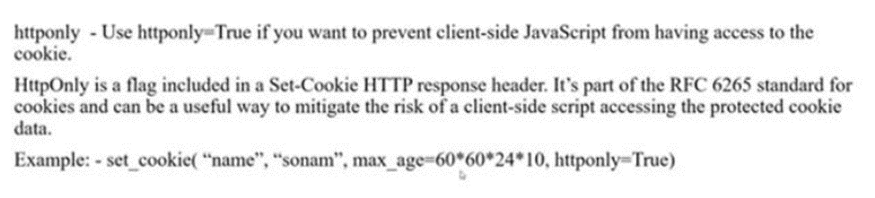
Cookies

ð **Detailed Parameters in Cookies:**









CREDITS: GEEKY-SHOWS

**ð Conclusion:-**

***This was all about the Django COOKIES. With this, We can SET SOME DATA IN CLIENT MACHINE.***

DEMONSTRATIONS:

<https://aksh60.pythonanywhere.com/>

<https://aksh61.pythonanywhere.com/>

52