PYTHON DJANGO

By Rishi

JANUARY 1, 2019

RISHI.H2KINFOSYS@GMAIL.COM
www.h2kinfosys.com

```
C:\>cd Rishi

C:\RISHI>cd Dev

C:\RISHI\Dev>mkdir myDjang

C:\RISHI\Dev>dir
 Volume in drive C is Windows
 Volume Serial Number is 0CDC-A763

 Directory of C:\RISHI\Dev

31-07-2019  21:38    <DIR>          .
31-07-2019  21:38    <DIR>          ..
31-07-2019  21:38    <DIR>          myDjang
               0 File(s)              0 bytes
               8 Dir(s)  575,761,174,528 bytes free

C:\RISHI\Dev>cd myDjang

C:\RISHI\Dev\aprDjango>virtualenv .
created virtual environment CPython3.8.2.final.0-32 in 19295ms
  creator CPython3Windows(dest=C:\RISHI\Dev\aprDjango, clear=False, global=False)
  seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=latest, via=copy,
app_data_dir=C:\Users\Ambarish\AppData\Local\pypa\virtualenv\seed-app-data\v1.0.1)
  activators
BashActivator,BatchActivator,FishActivator,PowerShellActivator,PythonActivator,XonshActivator

C:\RISHI\Dev\myDjang>dir
 Volume in drive C is Windows
 Volume Serial Number is 0CDC-A763

 Directory of C:\RISHI\Dev\myDjang

31-07-2019  21:39    <DIR>          .
31-07-2019  21:39    <DIR>          ..
16-04-2019  11:20    <DIR>          Include
31-07-2019  21:39    <DIR>          Lib
28-03-2018  17:07            30,340 LICENSE.txt
31-07-2019  21:40    <DIR>          Scripts
31-07-2019  21:39    <DIR>          tcl
               1 File(s)         30,340 bytes
               6 Dir(s)  575,724,306,432 bytes free

C:\RISHI\Dev\myDjang>cd Scripts

C:\RISHI\Dev\myDjang\Scripts>dir
 Volume in drive C is Windows
 Volume Serial Number is 0CDC-A763

 Directory of C:\RISHI\Dev\myDjang\Scripts
```

```
31-07-2019  21:40    <DIR>          .
31-07-2019  21:40    <DIR>          ..
31-07-2019  21:40             2,315 activate
31-07-2019  21:40               883 activate.bat
31-07-2019  21:40             2,038 activate.ps1
31-07-2019  21:40             1,159 activate.xsh
31-07-2019  21:40             1,517 activate_this.py
31-07-2019  21:40               512 deactivate.bat
31-07-2019  21:40           102,783 easy_install-3.6.exe
31-07-2019  21:40           102,783 easy_install.exe
31-07-2019  21:40           102,765 pip.exe
31-07-2019  21:40           102,765 pip3.6.exe
31-07-2019  21:40           102,765 pip3.exe
31-07-2019  21:39           100,504 python.exe
31-07-2019  21:39            58,520 python3.dll
31-07-2019  21:39         3,610,776 python36.dll
31-07-2019  21:39            98,968 pythonw.exe
31-07-2019  21:40           102,761 wheel.exe
              16 File(s)      4,493,814 bytes
               2 Dir(s)  575,724,306,432 bytes free

C:\RISHI\Dev\myDjang\Scripts>activate

(myDjang) C:\RISHI\Dev\myDjang\Scripts>
(myDjang) C:\RISHI\Dev\myDjang\Scripts>cd ..

(myDjang) C:\RISHI\Dev\myDjang>mkdir src

(myDjang) C:\RISHI\Dev\myDjang>dir
 Volume in drive C is Windows
 Volume Serial Number is 0CDC-A763

 Directory of C:\RISHI\Dev\myDjang

31-07-2019  21:43    <DIR>          .
31-07-2019  21:43    <DIR>          ..
16-04-2019  11:20    <DIR>          Include
31-07-2019  21:39    <DIR>          Lib
28-03-2018  17:07            30,340 LICENSE.txt
31-07-2019  21:40    <DIR>          Scripts
31-07-2019  21:43    <DIR>          src
31-07-2019  21:39    <DIR>          tcl
               1 File(s)         30,340 bytes
               7 Dir(s)  575,725,363,200 bytes free

(aprDjango) C:\RISHI\Dev\aprDjango\src>pip install django
Collecting django
  Downloading Django-3.0.6-py3-none-any.whl (7.5 MB)
     |████████████████████████████████| 7.5 MB 3.3 MB/s
Collecting asgiref~=3.2
```

```
  Using cached asgiref-3.2.7-py2.py3-none-any.whl (19 kB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.3.1-py2.py3-none-any.whl (40 kB)
Collecting pytz
  Using cached pytz-2020.1-py2.py3-none-any.whl (510 kB)
Installing collected packages: asgiref, sqlparse, pytz, django
Successfully installed asgiref-3.2.7 django-3.0.6 pytz-2020.1 sqlparse-0.3.1
WARNING: You are using pip version 20.0.2; however, version 20.1 is available.
You should consider upgrading via the 'C:\RISHI\Dev\aprDjango\Scripts\python.exe -m pip install --
upgrade pip' command.


(myDjang) C:\RISHI\Dev\myDjang\ src >
(myDjang) C:\RISHI\Dev\myDjang\src >pip freeze
Django==2.0.7
pytz==2019.2

(myDjang) C:\RISHI\Dev\myDjang>cd src
(myDjang) C:\RISHI\Dev\myDjang\src>django-admin

Type 'django-admin help <subcommand>' for help on a specific subcommand.

Available subcommands:
[django]
    check
    compilemessages
    createcachetable
    dbshell
    diffsettings
    dumpdata
    flush
    inspectdb
    loaddata
    makemessages
    makemigrations
    migrate
    runserver
    sendtestemail
    shell
    showmigrations
    sqlflush
    sqlmigrate
    sqlsequencereset
    squashmigrations
    startapp
    startproject
    test
    testserver
```

Note that only Django core commands are listed as settings are not properly configured (error: Requested setting INSTALLED_APPS, but settings are not configured. You must either define the

environment variable DJANGO_SETTINGS_MODULE or call settings.configure() before accessing settings.).

(*myDjang) C:\RISHI\Dev\myDjang\src>django-admin startproject myDjang .*

```
(myDjang) C:\RISHI\Dev\myDjang\src>dir
 Volume in drive C is Windows
 Volume Serial Number is 0CDC-A763

 Directory of C:\RISHI\Dev\myDjang\src

31-07-2019  21:50    <DIR>          .
31-07-2019  21:50    <DIR>          ..
31-07-2019  21:50    <DIR>          myDjang
31-07-2019  21:50               559 manage.py
               1 File(s)            559 bytes
               3 Dir(s)  575,693,553,664 bytes free

(aprDjango) C:\RISHI\Dev\aprDjango\src>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you apply the
migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
May 04, 2020 - 20:42:08
Django version 3.0.6, using settings 'aprDjango.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Open second command prompt and execute below command:

```
(septDjango) C:\RISHI\Dev\septDjango\src>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
```

Applying auth.0009_alter_user_last_name_max_length... OK
Applying sessions.0001_initial... OK

(septDjango) C:\RISHI\Dev\septDjango\src>python manage.py makemigrations
No changes detected

(myDjang) C:\RISHI\Dev\myDjang\src>python manage.py createsuperuser
Username: admin
Email address: admin@myDjang.com
Password:
Password (again):
Superuser created successfully.

(myDjang) C:\RISHI\Dev\myDjang\src>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
July 31, 2019 - 22:07:54
Django version 2.0.7, using settings 'myDjang.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
[31/Jul/2019 22:08:08] "POST /admin/login/?next=/admin/ HTTP/1.1" 302 0
[31/Jul/2019 22:08:09] "GET /admin/ HTTP/1.1" 200 2984
[31/Jul/2019 22:08:09] "GET /static/admin/css/dashboard.css HTTP/1.1" 200 412
[31/Jul/2019 22:08:09] "GET /static/admin/css/responsive.css HTTP/1.1" 304 0
[31/Jul/2019 22:08:09] "GET /static/admin/css/base.css HTTP/1.1" 304 0
[31/Jul/2019 22:08:09] "GET /static/admin/css/fonts.css HTTP/1.1" 304 0
[31/Jul/2019 22:08:09] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 304 0
[31/Jul/2019 22:08:09] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 304 0
[31/Jul/2019 22:08:09] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 304 0
[31/Jul/2019 22:08:09] "GET /static/admin/img/icon-addlink.svg HTTP/1.1" 200 331
[31/Jul/2019 22:08:09] "GET /static/admin/img/icon-changelink.svg HTTP/1.1" 200 380
[31/Jul/2019 22:08:21] "GET /admin/auth/user/ HTTP/1.1" 200 7089
[31/Jul/2019 22:08:21] "GET /static/admin/css/changelists.css HTTP/1.1" 200 6170
[31/Jul/2019 22:08:21] "GET /admin/jsi18n/ HTTP/1.1" 200 3185
[31/Jul/2019 22:08:22] "GET /static/admin/js/jquery.init.js HTTP/1.1" 200 363
[31/Jul/2019 22:08:22] "GET /static/admin/js/vendor/jquery/jquery.js HTTP/1.1" 200 258648
[31/Jul/2019 22:08:22] "GET /static/admin/js/core.js HTTP/1.1" 200 7134
[31/Jul/2019 22:08:22] "GET /static/admin/js/admin/RelatedObjectLookups.js HTTP/1.1" 200 6897
[31/Jul/2019 22:08:22] "GET /static/admin/js/actions.js HTTP/1.1" 200 6502
[31/Jul/2019 22:08:22] "GET /static/admin/js/prepopulate.js HTTP/1.1" 200 1538
[31/Jul/2019 22:08:22] "GET /static/admin/js/vendor/xregexp/xregexp.js HTTP/1.1" 200 128820
[31/Jul/2019 22:08:22] "GET /static/admin/js/urlify.js HTTP/1.1" 200 8729
[31/Jul/2019 22:08:22] "GET /static/admin/img/search.svg HTTP/1.1" 200 458
[31/Jul/2019 22:08:22] "GET /static/admin/img/icon-yes.svg HTTP/1.1" 200 436
[31/Jul/2019 22:08:22] "GET /static/admin/img/tooltag-add.svg HTTP/1.1" 200 331
[31/Jul/2019 22:08:22] "GET /static/admin/img/sorting-icons.svg HTTP/1.1" 200 1097
[31/Jul/2019 22:08:40] "GET /admin/ HTTP/1.1" 200 2984
[31/Jul/2019 22:08:42] "GET /admin/auth/group/ HTTP/1.1" 200 3584
[31/Jul/2019 22:08:42] "GET /admin/jsi18n/ HTTP/1.1" 200 3185

[31/Jul/2019 22:08:46] "GET /admin/ HTTP/1.1" 200 2984
Performing system checks...

System check identified no issues (0 silenced).
July 31, 2019 - 22:17:44
Django version 2.0.7, using settings 'myDjang.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

On Another Command Prompt

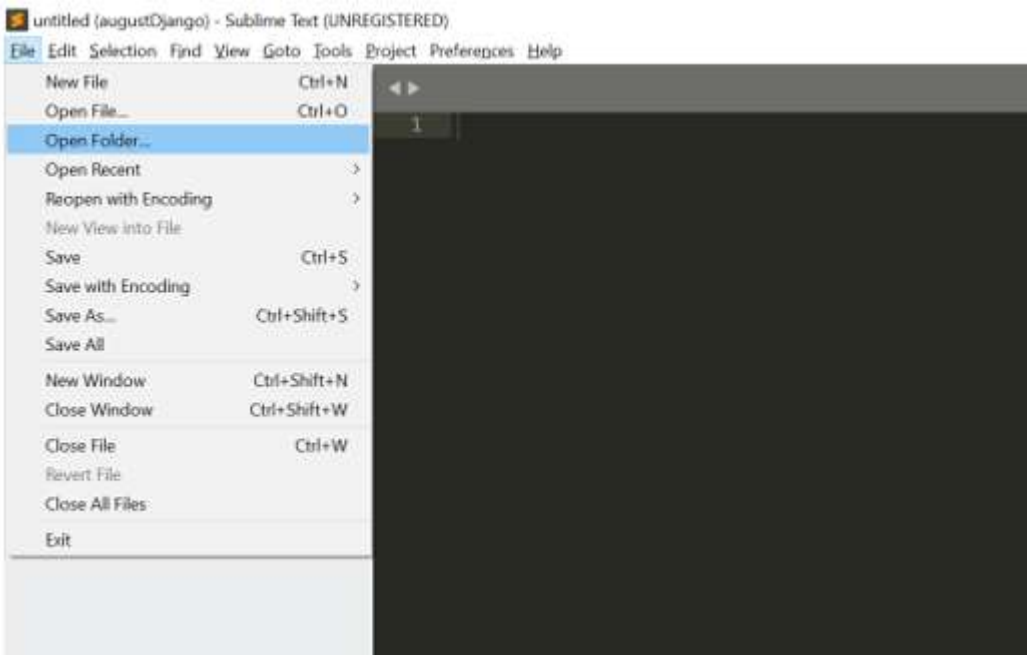
(*myDjang) C:\RISHI\Dev\myDjang\src>python manage.py makemigrations*

(myDjang) C:\RISHI\Dev\myDjang\src>python manage.py migrate

(myDjang) C:\RISHI\Dev\myDjang\src>

https://docs.djangoproject.com/en/2.2/ref/models/fields/

Important note for you, download Subline text: https://www.sublimetext.com/3



Use this option to navigate to Virtual environment folder and open the project here.

Understanding Settings:

BASE_DIR – this is where your project is.
DEBUG = True → Very important for development. Don't deploy this in production
INSTALLED_APPS → think of apps more like a component you are going to develop
ROOT_URLCONF → Context Root of application
TEMPLATES → Configure your HTML pages here. We will learn more along the way
DATABASES → Django by default maps sql lite 3. You can map your database instance here

AUTH_PASSWORD_VALIDATORS → Password authenticators

Creating your Own App:

Creating your own applications (Apps):
(*myDjango) C:\RISHI\Dev\myDjango\src>python manage.py startapp products*

Product app should do only product related things. It should be concise to product functionality.

first:

```
from django.db import models

# Create your models here.
class Product(models.Model):

    title       = models.TextField()
    description = models.TextField()
    price       = models.TextField()
```

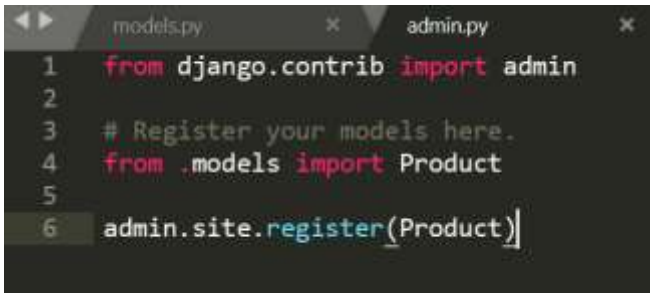
That's it. Now go to settings and add it in Installed Apps.

```
#Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    #own apps
    'products',
]
```

Every time you change a model or create a new model, run below commands

```
(myDjango) C:\RISHI\Dev\myDjango\src>python manage.py makemigrations
Migrations for 'products':
  products\migrations\0001_initial.py
    - Create model Product

(myDjango) C:\RISHI\Dev\myDjango\src>python manage.py migrate
  Apply all migrations: admin, auth, contenttypes, products, sessions
Running migrations:
  Applying products.0001_initial... OK

(myDjango) C:\RISHI\Dev\myDjango\src>
```

Now go to admin.py and make these changes:



Now start the server if its not already running and open Admin page.

(myDjango) C:\RISHI\Dev\myDjango\src>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
August 01, 2019 - 10:49:38
Django version 2.0.7, using settings 'myDjango.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

Login to Admin page and see new Product got added on Page. Can you add new products? Check it out.

Let's change the Model:

We will use these model fields https://docs.djangoproject.com/en/2.2/ref/models/fields/

You may not need to delete DB, please try the steps directly:

(septDjango) C:\RISHI\Dev\septDjango\src>python manage.py makemigrations
You are trying to add a non-nullable field 'promoted' to product without a default; we can't do that (the database needs something to populate existing rows).
Please select a fix:
1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
2) Quit, and let me add a default in models.py
Select an option: 1
Please enter the default value now, as valid Python
The datetime and django.utils.timezone modules are available, so you can do e.g. timezone.now
Type 'exit' to exit this prompt

```
>>> False
Migrations for 'products':
  products\migrations\0002_auto_20190910_2055.py
    - Add field promoted to product
    - Add field summary to product
    - Alter field description on product
    - Alter field price on product
    - Alter field title on product

(septDjango) C:\RISHI\Dev\septDjango\src>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, products, sessions
Running migrations:
  Applying products.0002_auto_20190910_2055... OK
```

Use this only when above options are not working.

If above commands doesn't work.. then do below steps:

1. Stop the server
2. Delete all the files in migrations folder
3. Delete pyCache. You can keep init.
4. Delete sql.lite DB file

Lets make the change in Model now.:

Now before making migrations, you need to create user, you need to recreate it.. why? Remember you have deleted the database.

```
(myDjango) C:\RISHI\Dev\myDjango\src>python manage.py makemigrations
Migrations for 'products':
  products\migrations\0001_initial.py
    - Create model Product

(myDjango) C:\RISHI\Dev\myDjango\src>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, products, sessions
Running migrations:
  Applying products.0001_initial... OK
```

Now login to admin → navigate to create Product and enjoy the new view. Have you need Boolean Field?

Handling Runtime Model Changes:

Make some changes in Model now:

```
from django.db import models

# Create your models here.
class Product(models.Model):

    name        =   models.CharField(blank=False,null=False, max_length=20) # max_length = required
    description =   models.TextField(null=True) # blank = True or null = True
    price       =   models.DecimalField(max_digits=5, decimal_places=2) # Check the docs for requir
    Summary     =   models.TextField(default='Very Interesting')
    pramotions  =   models.BooleanField()
    newField    =   models.BooleanField()
```

Now, Database don't know how to handle this new field. What should be the value of this field for previous entries. It will ask you same question while migrating.

(myDjango) C:\RISHI\Dev\myDjango\src>python manage.py makemigrations
You are trying to add a non-nullable field 'newField' to product without a default; we can't do that (the database needs something to populate existing rows).
Please select a fix:
1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
2) Quit, and let me add a default in models.py
Select an option:

Give option as you think. Option 2 is better when you want to go back and enter some default value. Here we will use option 1

Select an option: 1
Please enter the default value now, as valid Python
The datetime and django.utils.timezone modules are available, so you can do e.g. timezone.now
Type 'exit' to exit this prompt
>>> True
Migrations for 'products':
products\migrations\0002_auto_20190801_1140.py
- Add field newField to product
- Alter field Summary on product

Observe migrations folder. Check how Django handled this migration.

Now, go ahead and add migrate, then start server and check previous products.

```
(myDjango) C:\RISHI\Dev\myDjango\src>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, products, sessions
Running migrations:
  Applying products.0002_auto_20190801_1140... OK

(myDjango) C:\RISHI\Dev\myDjango\src>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).
August 01, 2019 - 11:43:27
Django version 2.0.7, using settings 'myDjango.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK
```

blank = True or null = True? Which one to use.

Blank is about Field on Web Page. Null Is about database. If Blank is false, you cannot leave the field blank.

Default Home Page to Custom Home Page

Before starting this section, just do this for me:

1. Create a new app called “pages”
2. Add your new module in settings.

If you don’t remember how to do it, please revise the videos.

3. Go to pages app and open views.py. Add below code.

```python
from django.http import HttpResponse
from django.shortcuts import render

# Create your views here.

def home_view(*args, **kwargs): # args and kwargs is Python, we will discuss this
    return HttpResponse("<h1> Hello World </h1>") # String of HTML body
```

4. Now go to myDjango folder and open url.py – as mentioned in comments above, please add our view here

```python
from django.contrib import admin
from django.urls import path

from pages.views import home_view

urlpatterns = [
    path('', home_view, name='home'),
    path('admin/', admin.site.urls),
]
```

5. Now hit the URL: http://localhost:8000/

URL Routing and Requests:

```python
urlpatterns = [
    path('', home_view, name='home'),
    path('home/', home_view, name='home'),
    path('admin/', admin.site.urls),
]
```

Try access URL with http://localhost:8000/home/ ← Observe if anything changes?

Assignment:

Can you add Contact page like this? Lets do it. Add as many view as you want. Play around with it.

```python
from django.contrib import admin
from django.urls import path

from pages.views import home_view, contact_view

urlpatterns = [
    path('', home_view, name='home'),
    path('home/', home_view, name='home'),
    path('contact/', contact_view, name='contact'),
    path('admin/', admin.site.urls),
]
```

What else we can do with views?

Can we access Request Parameters coming in web request.

```python
def home_view(request, *args, **kwargs): # args and kwargs is Python, we will discuss this
    print(request.user, args, kwargs)
    return HttpResponse("<h1> Hello World </h1>") # String of HTML body
```

Lets go to HTML Page:

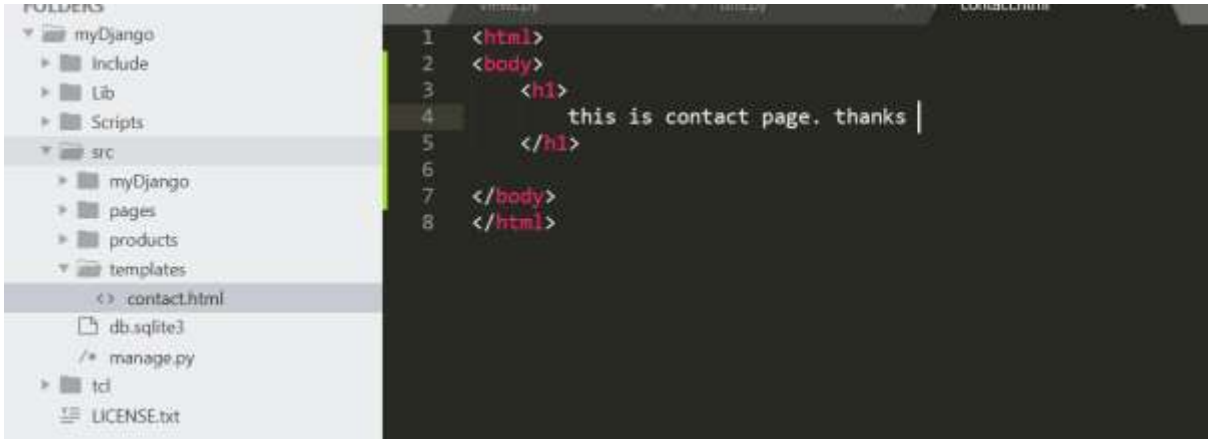
```python
def contact_view(request, *args, **kwargs): # args and kwargs is Python, we will discuss this
    print(request.user, args, kwargs)
    return render(request, "contact.html", {})
```

add method to views.py. Create HTML page

views.py | urls.py | contact.html

```python
from django.http import HttpResponse
from django.shortcuts import render

# Create your views here.

def home_view(request, *args, **kwargs): # args and kwa
    print(request.user, args, kwargs)
    return HttpResponse("<h1> Hello World </h1>") # Str


def contact_view(request, *args, **kwargs): # args and
    print(request.user, args, kwargs)
    return render(request, "contact.html", {})
```

HTML Page in template folder. Create ***templates*** folder directly under src.



Go to settings and add this directory in DIRS in TEMPLATES section.

Run this project now. See if you get web page.

But the path given here will not work on your machine if I send you this code. So what should we do? Lets use BASE_DIR

Assignment 2:

Can you create more html pages now like – about.html, contact.html, home.html and point it to web address? Please try and let me know.

Django Template Inheritance.

You can use UserModel in template engine. Lets try this for a simple thing like **request.user**.

```
<html>
<body>
    <h1>
        this is contact page. thanks
    </h1>
    {{request.user}}
</body>
</html>

```

Check the response now.

You might need to use something common across all html pages, lets say a navigation bar or some meta data. So lets create a page called in template to handle this.

```
<html>
    <head>
        <title>Coding for H2KInfosys Students</title>
    </head>
    <body>
        {% block content %}
            Replace me
        {% endblock %}
    </body>
</html>
```

Now use this block content in contact.html:

base.html | contact.html

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

{% block content %}
    <h1>
            this is contact page. thanks
    </h1>
    <p> {{request.user}} </p>
{% endblock %}
```

Now refresh the output screen and check the output. Can you make similar change in other html pages you created?

Assignment 3:

Remember {% block content %} → block is Django stuff while "content" is my variable. You can change that too. Just make sure you are using same variable everywhere.

Wait.. can you add more blocks then? Answer is – YES. Can you try that? Please do.

Include template tag:

Now I want a navbar in all pages. So do I have to add it in my base.html? YES. But that make base.html really heavy in UI related entities. So lets do this: I am creating another html page called navbar.html

```
<nav>
    <ul>
        <li>Electronics</li>
        <li>Daily Needs</li>
        <li>Stationary</li>
    </ul>
</nav>
```

Now use **include** to use this html page in base.html

```
<html>
    <head>
        <title>Coding for H2KInfosys Students</title>
    </head>
    <body>
        {% include 'navbar.html' %}
        {% block content %}
            Replace me
        {% endblock %}
    </body>
</html>
```

Load **contact.html** now. See the change.

Rendering Context in a Template:

What we really need on web page is Data from database isn't it?

For Django, **User Page = template + context**. What do I mean by that? Remember that empty dictionary we passed to html page? Add something in it now.

```
def contact_view(request, *args, **kwargs):
    my_context = {
        "my_text" : "This is a text from Context",
        "my_intVar" : 1000,
        "my_list" : [123, 234, 345, 456],
    }
    return render(request, 'contact.html', my_context)
```

Can we add a List in context? How can we show that on screen as HTML List?

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

    {% block content %}
    <h1> Contact Us page: </h1>

    <p> {{ my_text }} </p>
    <p> {{ my_intVar }} </p>
    <p> {{ my_list }} </p>

    <ul>
    {% for eachItem in my_list %}
        <li><p> {{ forloop.counter }} - {{ eachItem }} </p> </li>
    {% endfor %}
    </ul>
{% endblock %}
```

Can we use **Conditions** in Templates?

```
    <ul>
    {% for eachItem in my_list %}
        {% if eachItem == 565 %}
            <li><p> {{ forloop.counter }} - {{ eachItem|add:2 }} </p> </li>
        {% elif eachItem == "abc" %}
            <li><p> This is not a good data </p> </li>
        {% else %}
            <li><p> {{ forloop.counter }} - {{ eachItem }} </p> </li>
        {% endif %}
    {% endfor %}
    </ul>
```

What is this |add:2 ← this is built in template tag filter. Lets check Django page for this. Search for "Built-in filter reference"

https://docs.djangoproject.com/en/2.2/ref/templates/builtins/

I strongly recommend you go through entire set and try few options.

Getting Data from DB:

Step 1: Open views from product app and create a view method. You access Product object with

Product.objects.get(id=N)

views.py

```python
from django.shortcuts import render

from .models import Product

# Create your views here.

def product_details_view(request):
    obj = Product.objects.get(id=1)
    context = {
        "name"          : obj.name,
        "description"   : obj.description,
    }
    render(request, "product/details.html", context)
```

Add this view in URLs.

```python
from django.contrib import admin
from django.urls import path

from pages.views import home_view, contact_view
from products.views import product_details_view

urlpatterns = [
    path('', home_view, name='home'),
    path('home/', home_view, name='home'),
    path('contact/', contact_view, name='contact'),
    path('product/', product_details_view),
    path('admin/', admin.site.urls),
]
```

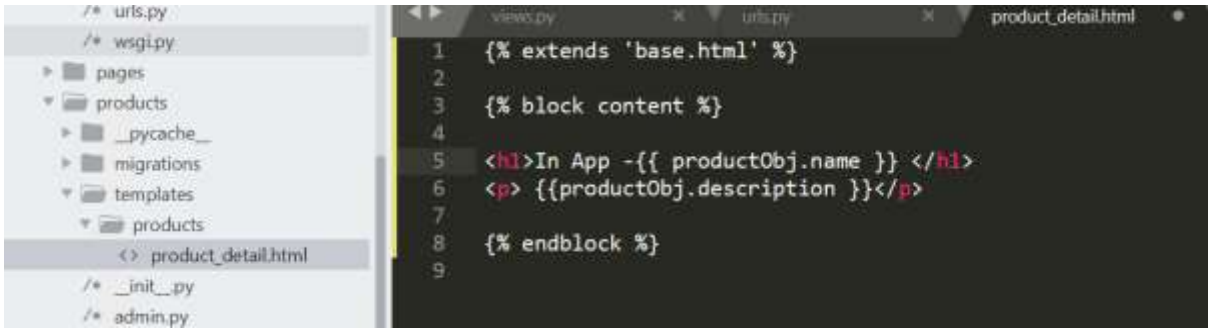
Create product/details.html page in templates folder.

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

{% block content %}

<h1>{{ name }} </h1>
<p> {{ description }}</p>

{% endblock %}
```

But then, why are we adding such a complicated context. Can we make it simple?

```python
def product_details_view(request):
    obj = Product.objects.get(id=1)
    print(obj.name)
    context = {
        "productObj" : obj,
    }
    return render(request, "products/product_detail.html", context)
```

Now I am moving my template into App itself. How? Lets create template folder under product app.



This make my code more modular. Isn't it?

Django Forms:

Aim is to create Product Form, which will take user inputs to create record in DB. First Step to create forms.py and create model Form like below:

```python
forms.py
from django import forms
from .models import Product

class ProductForm(forms.ModelForm):
    class Meta:
        model = Product
        fields = [
            "name",
            "description",
            "price",
            "Summary",
            "UniqueFeature",
            "pramotions",
        ]
```

We have to load this form from view. So add method:

```python
def product_create_view(request):
    forms = ProductForm(request.POST or None)
    if forms.is_valid():
        forms.save()
        forms = ProductForm()

    my_context = {
        "forms" : forms,

    }
    return render(request, 'product/product_create.html', my_context)
```

Now complete product_create.html

product_create.html

```
{% extends 'base.html' %}
    {% block content %}
    <h1>Product Create Page: </h1>

    <form method="POST">
        {{ forms.as_p }}
        <input type="submit" name="Save">
        {% csrf_token %}
    </form>
{% endblock %}

```

Understand csrf_token as a part of security. We will discuss it in detail in class. Also, forms.as_p makes entire form as <P> tags.

Next step? Add this view in URL.

```python
from django.contrib import admin
from django.urls import path
from pages.views import home_view, contact_view
from products.views import product_detail_view, product_create_view

urlpatterns = [
    path('admin/', admin.site.urls),
    path('home/', home_view, name='home'),
    path('', home_view, name='home'),
    path('contact/', contact_view, name='contact'),
    path('product/', product_detail_view),
    path('create/', product_create_view),

]
```

Restart the server and check the code. If you see any error, revise the steps and look for mistake. Still can't get it? Email me: rishi.h2kinfosys@gmail.com

Form Validation Method:

First I want to tell you that you can override every field in model class. Can we try one?

```python
class ProductForm(forms.ModelForm):
    title       = forms.CharField(label='',
                    widget=forms.TextInput(attrs={"placeholder": "Your title"}))
    description = forms.CharField(
                        required=False,
                        widget=forms.Textarea(
                                attrs={
                                    "placeholder": "Your description",
                                    "class": "new-class-name two",
                                    "id": "my-id-for-textarea",
                                    "rows": 20,
                                    'cols': 120
                                }
                            )
                        )
    price       = forms.DecimalField(initial=199.99)

    class Meta:
        model = Product
        fields = [
            'title',
            'description',
            'price'
        ]
```

Lets talk about widgets and attributes.

With this code, you have overridden the form coming from Model by default.

Now how validation works?

Suppose I want to check my title field has “H2K” letters, I can write a specific method with name **clean_<my_field_name>(self)**

```python
def clean_title(self, *args, **kwargs):
    title = self.cleaned_data.get("title")
    if not "CFE" in title:
        raise forms.ValidationError("This is not a valid title")
    if not "news" in title:
        raise forms.ValidationError("This is not a valid title")
    return title
```

Of course, if title is not valid, you can raise a ValidationError which will be shown on screen.

So will you write email validation for me?

Setting Initial Data to Fields:

Use initial_data dictionary to set initial data. Please see the example below:

```python
def render_initial_data(request):
    initial_data = {
        'title': "My this awesome title"
    }
    form = ProductForm(request.POST or None, initial=initial_data)
    context = {
        'form': form
    }
    return render(request, "products/product_create.html", context)
```

You can also load Database entry to set as initial value:

```
def render_initial_data(request):
    initial_data = {
        'title': "My this awesome title"
    }
    obj = Product.objects.get(id=1)
    form = ProductForm(request.POST or None,instance=obj)
    if form.is_valid():
        form.save()
    context = {
        'form': form
    }
    return render(request, "products/product_create.html", context)
```

Dynamic Data Loading:

Dynamically pass the data with URL:

```
urlpatterns = [
    path('products/<int:my_id>/', dynamic_lookup_view, name='product'),
```

Handle this in method which shows view:

```
from django.shortcuts import render
from .models import Product

def dynamic_lookup_view(request, my_id):
    obj = Product.objects.get(id=my_id)
    context = {
        "object": obj
    }
    return render(request, "products/product_detail.html", context)
```

But what if someone sends an ID which doesn't exists in DB?

Handle DoesNotExist / 404

You can achieve 404 Handling with get_object_or_404() or Http404 – see the code below:

```
from django.http import Http404
from django.shortcuts import render, get_object_or_404
from .models import Product

def dynamic_lookup_view(request, id):
    #obj = Product.objects.get(id=id)
    # obj = get_object_or_404(Product, id=id)
    try:
        obj = Product.objects.get(id=id)
    except Product.DoesNotExist:
        raise Http404
    context = {
        "object": obj
    }
    return render(request, "products/product_detail.html", context)
```