Django Tutorials:

Started on 8 May 2020 - Udemy Course

 # Video 3 Python:

Notes:
Open the terminal.
Minimum python requirement is 3.6 for Django.
Always use python3 to start coding.
If you don't have, then go to python.org and download it.

Command: python —version
results 2.7.16

Command: python3
result: 3.7.3

 # Video 4 Variable, Strings, Int and print :

Notes:
Author is using Atom and i am using Virtual studio
Create a new file such as pythonPractise.py

Commands:
cd Desktop

Copy and paste the file in the terminal to execute the file

python3 /Users/nagz-digital/Desktop/Django/pythonPractise.py

or

Command:
cd Desktop/Django_project

Notes: "This will change the directory to Django project and now we can use python to open the file"

Command:
python3 pythonPractise.py and enter

Move directory (backward) :
command: cd ..

To comment out multiple lines use in editor: “ Command + / ”

Installing Django:

open terminal:

install pip: sudo easy_install pip

install virtual environment: sudo pip install virtualenv

Change the directory to Desktop: cd Desktop

Add a directory on the desktop with virtualenv: virtualenv pwd_generator

check the files by changing the directory by: cd pwd_generator

See the files by ls command: ls

activate the virtual environment: source bin/activate

Alternative:
install python 3 to virtualev: virtualenv -p python3 ~/pwd_generator
activate the virtual environment: source ~/pwd_generator/bin/activate

Directory will change to Virtual environment:

check python version: python -V

Now Install Djando: pip3 install django

Type: ‘pip freeze’ to see if installed

Upgrade django: pip install django —upgrade

Notes: we can also install on our main machine, but it is safe to use virtual environment.

Once we are in the virtual env we can create a new project:

Start a project:

‘django-admin.py startproject first_project’

or

‘django-admin startproject first_project’

find all the files with ‘ls’ and if you see manage.py, then you are installed properly

Start the server: python3 manage.py runserver

You will see the http link, copy and open in a browser.

Cheatsheet: https://zappycode.com/post/1/django-cheat-sheet

https://blog.couchbase.com/tips-and-tricks-for-upgrading-from-python-2-to-python-3/

Updated Installation technique:

Open terminal:

install pip: sudo easy_install pip

install virtual environment: sudo pip install virtualenv

Change the directory to Desktop: “cd Desktop”

Add a directory on the desktop with virtualenv: “virtualenv django_project”

I**nstall python 3 to above folder:** virtualenv -p python3 ~/django_project

Activate the virtual environment: source ~/django_project/bin/activate
Notes: once activated the path will start with the folder name

Check python version: python -V

Install Djando: pip3 install django

Change the directory to django_project: cd django_project

Create a new project: django-admin startproject password_generator
Notes: password_generator is the folder name
Change this name later to ‘password_generator_project’ later

Change the directory to password_generator: cd password_generator

Start the server: python3 manage.py runserver

Copy the url and open in a browser: http://127.0.0.1:8000

To stop the server: control C

Notes: correct path to start the server

(django_project) Nagzs-MBP:password_generator nagz-digital$ python3 manage.py runserver

==========
==========
==========

Process to start Django project:

Change the directory to desktop : cd Desktop

Activate the virtual environment: source ~/django_project/bin/activate
Notes: once activated the path will start with the folder name

Change the directory to django_project: cd django_project
(change directory accordingly)

Change the directory to password_generator_project: cd password_generator_project
(change directory accordingly)

Start the server: python3 manage.py runserver

==========
==========
==========

Process to create a new project:

Change the directory to Desktop: “cd Desktop”

Activate the virtual environment: source ~/django_project/bin/activate
Notes: once activated the path will start with the folder name

Change the directory to django_project: cd django_project

Create a new project: django-admin startproject personal_portfolio
Notes: personal_portfolio is the folder and project name
Change this name to ‘personal_portfolio_project’

Change the directory to newly created folder: cd personal_portfolio_project

Create a new app inside the folder: python3 manage.py startapp blog
Create a new app inside the folder: python3 manage.py startapp portfolio

Open the folders in editor

==========

Folder structure:

Rename the password_generator folder inside the djando_project to password_generator_project or anything else, because there is another folder with the same name inside this folder. Changing the name will stop all the confusion.

Top Level folder inside the password_generator_project folders are:

manage.py
db.sqlite3
password_generator folder

To see the connect and help command inside the manage.py. Stop the server and enter - “python3 manage.py help” this will open all the available commands in the terminal.

Password_generator/settings.py

Lesson: 24 Apps

Project is a website and app is a part of it, apps are pieces.

Create a new app

Stop the server

python3 manage.py startapp generator. (generator is the name, you can say anything)

Once the folder is created, then go to **password_generator/settings.py**

and add the recently created app under Installed apps, like:

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

In the above case i added the generator app in that list. After that restart the server again and there should not be any problem.

Lesson: 25 - URLs

Open password_generator/urls.py in the editor

Author showed how to open the admin page
for example;

http://127.0.0.1:8000/admin will open the admin page

Author delete the admin path and the admin import

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

urlpatterns = [
    path('admin/', admin.site.urls),

]
```

But i am keeping them for the time being.

Now creating a new path for the home page: path(‘’, views.home),

Home is name for the page which we created for the generator app. Now we need to import the views from the generator.
ex: from generator import views

Once saved the default page is gone and showing nothing.

Now open generator/views.py

Code which we added:

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

# Create your views here.

def home(request):
    return HttpResponse('Hello this is my first app')
```

The above code added the home page.

Create sub pages we can use the below code:

First go to urls.py and create a path:

path(‘subPageName’, views.subPageName),

ex:

path(‘contactus’, views.contactus),

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

Then go back to views.py and create a new function with the same name:

```
def contact(request):
    return HttpResponse('This is the contact us page.')
```

go to templates and create a new page called fo “contact”

Lesson 26: Templates

Templates are used for all the visual purpose

We need to create a Template folder manually inside the generator folder.

Create another folder called ‘generator’ inside the templates folder

Notes: This helps us to keep the templates separate from other apps.

Create a file inside the generator folder, which we just created. for ex:

home.html

and add some test html code.

Now got to views.py and change the home function to

```
def home(request):
    return render(request, 'generator/home.html')
```

Now the home.html is powering the home page.

A normal page works like this:

When a client enter the url, first the request go to urls.py and from there to views.py and to templates

urls.py —-> views.py. ——> templates (home.html)

Dictionary test:

We can add dictionary to respective page functions in views.py and later we can show them on the templates ex:

```
def home(request):
    return render(request, 'generator/home.html', {'password': 'ghhwf8978b78978'})
```

We can display this in the templates by adding the below code in home.html :

```
<h1>This is the new Home page</h1>
<p>Hello this is the home page</p>

{{ password }}
```

Lets create the app home page for the generator app:

Create a form:

```
<form action="">

    <select name ="length">
    <option value="6">6</option>
    <option value="7">7</option>
    <option value="8">8</option>
```

```
    </select>

    <input type="submit" value="Generate Password"></input>

</form>
```

Remember when we click the generate password button, the url will show the option we selected. which could be security risk.

To send the input to a new page we have to create a new page called password.html to display the results. We also have to add the name of the page in the
< form action = “Password”>

New page setup:
First go to urls.py and add the code:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.home),
    path('contact', views.contact),
    path('password/', views.password),
]
```

Second go to views.py and add the code:

```
def password(request):
    return render(request, 'generator/password.html')
```

Third step: go to templates folder and create a **password.html** file.

Notes: now if you add the password in form action. it will show you this password.html page, once you clicked the password generator button.

```
<form action="password">
```

Better technique to add the password page to form:

(Custom url)

```
<form action=" {% url 'password' %} ">
```

```
 path('generatedpassword/', views.password, name='password'),
```

Now it send the password to generatedpassword url.

http://127.0.0.1:8000/generatedpassword/?length=6

Lesson 28: making a random password

```
import random
```

Import random in views.py and add the below code:

```
def password(request):

    characters = list('abcdefghijklmnopqrstuvwxyz')

    length = 10

    thepassword = ''

    for x in range(length):
        thepassword += random.choice(characters)

    return render(request, 'generator/password.html', {'password': thepassword})
```

In the above code we are creating a function for the password page, that will display the random password.

add the below code in password.html

```
<h2>Your password is: {{ password }}</h2>
```

Lesson 29: using the form

create a full random password function inside the views.py

```
def password(request):
```

```
    characters = list('abcdefghijklmnopqrstuvwxyz')
    # Add the upper case to the passwords
    if request.GET.get('uppercase'):
        characters.extend(list('ABCDEFGHIJKLMNOPQRSTUVWXYZ'))

    # Add the special charaters to the passwords
    if request.GET.get('special'):
        characters.extend(list('!@£$%^&**&()'))

    # Add the numbers to the passwords
    if request.GET.get('special'):
        characters.extend(list('0123456789'))

    # 12 will be the default length and length is from home.html form
    length = int(request.GET.get('length', 12))

    thepassword = ''

    for x in range(length):
        thepassword += random.choice(characters)

    return render(request, 'generator/password.html', {'password': thepassword})
```

To add home page link on the password.html:

add the below code to password.html

```
<a href="{% url 'home' %}">Home Page</a>
```

and the below code to urls.py

```
path('', views.home, name ='home'),
```

—==========

Lesson 32: bootstrap

just add the link;

copy the link from the bootstrap website.

```
<link rel="stylesheet" href="https://
stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/
bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/
Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh"
crossorigin="anonymous">
```

Complete project:
https://github.com/zappycode/django3-password-generator
—==========

Section 4: Github

Lesson 36: Intro

Terminal : git

If installed you will see lots of items. if not go to:
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Change to directory to Desktop: cd Desktop

Change the directory to django_project: cd django_project

Change the directory to password_generator_project: cd password_generator_project

Add git: git init

To check invisible folder: ls -a

Add name and email to git:

https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

```
I used Aj Nagz and nagra888@hotmail.com
```

How to use safe point:
points are called commit's

Check status -
Terminal : git status

To add all the files -
Terminal : git add -A (staged)

Check the status again: git status

Finally to commit: git commit -m "Our first commit"

Note: to add new commits we must first use the 'git add -A' and then the

git commit -m "Some description here or version number"

==========

To go back to old version:

Terminal: git stash (this is move back to previous version)

Go to a particular version:

find all commits - Terminal: git log

copy the commit code and add: git checkout [paste here]

==========

GITHUB setup

Create an account

username: AjNagz
pass; n8@ail888G

Click on add new and name the repository

select private or public (i selected private)

get the push code and paste one by one in the same directory through terminal:

git remote add origin https://github.com/AJNagz/django3-password-generator.git
git push -u origin master

Notes: source tree website for drag and drop repository

https://www.sourcetreeapp.com

==========

Section 5 : Personal portfolio site

Lesson 38: New project setup

Process to create a new project:

Change the directory to Desktop: "cd Desktop"

Activate the virtual environment: source ~/django_project/bin/activate
Notes: once activated the path will start with the folder name

Change the directory to django_project: cd django_project

Create a new project: django-admin startproject personal_portfolio
Notes: personal_portfolio is the folder and project name
Change this name to 'personal_portfolio_project'

Change the directory to newly created folder: cd personal_portfolio_project

Create a new app inside the folder: python3 manage.py startapp blog
Create a new app inside the folder: python3 manage.py startapp portfolio

Open the folders in editor

We created two apps, so we need to enter that in the personal_portfolio/settings.py

Add the two apps under Installed_Apps:

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

```
    'portfolio',
]
```

Now start the server and see if we installed correctly:

python3 manage.py runserver

==========

Lesson 39: Models

Model fields: https://docs.djangoproject.com/en/3.0/ref/models/fields/#charfield

Open portfolio/models.py and add the following code for the database.

```
from django.db import models

class Project(models.Model):
    title = models.CharField(max_length=100)
    description = models.CharField(max_length=250)
    image = models.ImageField(upload_to='portfolio/images/')
    url = models.URLField(blank=True)
```

Pillow is required for images, to install pillow, stop the server:

'pip3 install pillow'

Warning:

```
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.
```

The above warning is a database migration warning.
To fix the above warning which show up when we start the server, can be fixed by:

'python3 manage.py migrate'

To create migration for the above entered code in models.py

stop the server: python3 manage.py makemigrations

If we start the server again it will show 1 unapplied migration, so we can use the same code again to fix the migration; 'python3 manage.py migrate'

To do this change and save. then use makemigrations command:
`python3 manage.py makemigrations`
If we decided to change any thing in the above models.py, the system will create a new file under migration/0002_initial.py (number will change) once we used the below command

After this the terminal will ask Y/N for the changes.

==========

Lesson: 41 - Admin and images

To create username and password, stop the server and enter:

python3 manage.py createsuperuser

It will ask for:
username: ajeetpal
email: nagra888@hotmail.com
password: Hardip@1978

To change passwords in future:

python3 manage.py changepassword

Use the above details to login to admin panel: http://127.0.0.1:8000/admin

To add models on the admin page, go to portfolio/admin.py

import code:

```
from django.contrib import admin
from .models import Project

# Register your models here.

admin.site.register(project)
```

After adding the above code, we can see the models in our admin page.

Once we uploaded all the details . The system will create an image folder under the portfolio. BUT this is not the right way explained by the author. He asked to **delete the folder** (portfolio/images) and add the following lines in settings.py

MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

save the file and upload a new image, The above code will create a **media folder under the root folder**.

Note: even if you click the image link directly from the admin panel, it won't open. To fix that.

Go to settings.py and add the below code:

```
# below code will create a media folder and a static link
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

Then go to urls.py

Import a new lib and code, look below:

```
from django.contrib import admin
from django.urls import path
from django.conf.urls.static import static
from django.conf import settings

urlpatterns = [
    path('admin/', admin.site.urls),
]

# This code will add the link to images
urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

After you saved it, you can click the images and it will open. ex url:

http://127.0.0.1:8000/media/portfolio/images/NaGGra-Digital-logo.png

==========

Lesson: 41 - Displaying objects

Create a home.html

1. **Go to portfolio_portfolio/urls.py. (we are using the main urls.py for portfolio app)**

import the following code:

```
# code to add the portfolio views
from portfolio import views
```

and add the path inside urlpatterns.

```
 path('', views.home, name='home'),
```

2. Go to portfolio/views.py

Make a home function:

```
from django.shortcuts import render
# this import will add all the models (database)
from .models import Project


def home(request):
    # projects variable will get all the database object
    projects = Project.objects.all()
    # here we created a dictionary to access the objects
    return render(request, 'portfolio/home.html',
{'projects':projects})
```

3. Create a template/portfolio/home.html in portfolio app

Create a new file called **home.html** .

To show model data on the home page, go to portfolio/views.py

Add a new import and a new code inside the home function:

```
from django.shortcuts import render
# this import will add all the models (database)
from .models import Project
```

```
def home(request):
    # projects variable will get all the database object
    projects = Project.objects.all()
    # here we created a dictionary to access the objects
    return render(request, 'portfolio/home.html', {'projects':projects})
```

Now open home.html and add the below code to access all the data objects:

```
<h1>This is home.html</h1>

<!-- below code is getting all the database object -->
<!-- but this will only show a list -->
<!-- we get '<QuerySet [<Project: Project object (1)>]>' -->
<!-- {{ projects }} -->
<!-- to fix that we have to use % and a for loop -->

{% for project in projects %}

<h3>{{ project.title }}</h3>
<p>{{ project.description }}</p>
<img src="{{ project.image.url }}" >
<br>
{% if project.url %}
<a href="{{ project.url }}">Link</a>
{% endif %}

{% endfor %}
```

==========

Lesson: 42 - Another sets of urls (for blog app) - VERY IMPORTANT

We can use the below style for all other apps.

1. Go to personal_portfolio/urls.py and add:

```
from django.urls import path, include

# see the reason above for this import
    path('blog/', include('blog.urls')),
```

2. Create a new urls.py inside the blog app

blog/urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.all_blogs, name='all_blogs'),
]
```

3. Go to blogs/views.py and create a new function

```
from django.shortcuts import render

# Create your views here.

def all_blogs(request):
    return render(request, 'blog/all_blogs.html')
```

4. Create a ne template folder inside blog app

blog/templates/blog/all_blogs.html

==========

Lesson: 43: Blog Model

1. Go to blog/models.py and add the following code:

```
from django.db import models

# Create your models here.
class Blog(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField()
    date = models.DateField()
```

2. Stop the server - control C

Create a migration: python3 manage.py makemigrations

Migrate: python3 manage.py migrate

Start the server: python3 manage.py runserver

3. Go to blog/admin.py and add the code (Blog is the class name)

```
from django.contrib import admin
from .models import Blog

# Register your models here.
admin.site.register(Blog)
```

4. Go to admin and create some blog posts.

5. Go to blog/views.py

```
from django.shortcuts import render
# import models from Blog class
from .models import Blog

# Create your views here.

def all_blogs(request):
    # store all the Blog class models in a variable
    blogs = Blog.objects.all()
    # Add a dictionary to the below code
    return render(request, 'blog/all_blogs.html', {'blogs': blogs})
```

6. Open templates/blog/all_blogs.html. (actually it is the main blog)

```
<h1>This is from all_blogs.html</h1>

{% for blog in blogs %}

<h3>{{ blog.title }}</h3>
<p>{{ blog.description }}</p>
```

```
<p>{{ blog.date }}</p>
```

```
{% endfor %}
```

7.To limit the number of blog posts - Go to blog/views.py

```
def all_blogs(request):
    # store all the Blog class models in a variable
    # blogs = Blog.objects.all()
    # To limit the number of posts us the below code
    blogs = Blog.objects.order_by('-date')[:3]
    # Add a dictionary to the below code
    return render(request, 'blog/all_blogs.html', {'blogs': blogs})
```

==========

Lesson: 44 - Looking inside the database

Download a sqlite3 viewer to view what's inside

Default database is sqlite3 and it's in the root folder (personal_portfolio_project)

We can change the database and for that we have to change the database setting in the settings.py

Note: Author hasn't explained how to change to mysql to mongoDB. Need more research on this topic.

Downloaded from https://sqlitebrowser.org/dl/

==========

Lesson: 45 - Static Files

Go to settings.py and check if the STATIC_URL = '/static/' is there

Open any app where you want the static files ex: portfolio and create a static/portfolio folder.

Add images inside the portfolio folder

To add them on any template:

open portfolio/templates/portfolio/home.html and add the code
Remember to add {% load static %} before the code

```
{% load static %}
<img src="{% static 'portfolio/penguin.jpg' %}”>
```

Also added the below code in urls.py because the above code was showing 404 not found error.

```
# The below code helps to show the static images
# code told by the author was not working and was showing 404 error
urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

Some interesting tutorials:
https://data-flair.training/blogs/django-static-files-handling/

https://docs.djangoproject.com/en/3.0/howto/static-files/