----------- Django Notes -------------

# DJango :->
-> It is backend web-development Framwork.

# Web-Developemnt :
Web-Developemnt = front-end + backend

# front-end TECH :->
=> It is use to for presentation purpose.
=> HTML , CSS , Bootstrap

# Backend TECH :=>
=> use to build a Bussines Logic.
=> Python , database(MySQL) , Django Framework

# Website => website is a combination of webpages.
=> any HTML page is called as webpage.
=> HTML code will be excute by browser.

# How To install django Framework ??
=> official website of django => www.djangoproject.com
=> pip install django

# Django Follows MVT Desgin Pattern.

M => Model	=> models.py	=> Database Logic	=> BackEnd
V => views	=> views.py	=> Bussiness Logic	=> BackEnd
T => Template	=> HTML file	=> Presentation Logic	=> FrontEnd

# For web development Editor :
Atom , sublim , vscode , pycharm
# vscode

==

# What is django project ??
=> django project is a combination of one or more django application and settings.
=> django_project = django_applications + settings

==

############# General steps to create django project ##################

# Common step for all projects:->
-> We need to create workspace for all django project.
-> workspace is nothing but a folder.
-> within workspace we will have all django projects.

```
cmd:-  mkdir "floder_name"
```
# mkdir commond is used to craete a floder / workspace.

```
cmd:-  cd 'floder_name'
```
# cd commond is used to change a directory / folder.

1) Django project
```
cmd: cd  'workspace/floder_name'
cmd: django-admin startproject project_name
cmd: cd project_name
```

2) runserver and send request
```
cmd: py manage.py runserver
```

#########General steps to create django project with view function #####

# V => views => views.py   => Bussiness Logic   => BackEnd
1) Django project
-> django-admin startproject commond is responsiable to create project in Django.

```
cmd: cd  'workspace/floder_name'

cmd: django-admin startproject project_name # create django project
cmd: cd project_name
```

2) Create Django Application.
-> manage.py file is responsiable to create django application.

cmd: py manage.py startapp app_name # create django application.

3) Register newly created django application in settings.py file.

```
INSTALLED_APPS = [
                'app_name'
          ]
```

4) Django Follows M V T Desgin pattern.so open views.py file and write some bussiness logic.

# V => View => views.py file => Bussiness Logic

# What is view ??
=> view is a function will take request as a input and return HttpResponse as a output to the end user.
=> view function accept user request and will provide output to the enduser.
=> HttpResponse will take HTML code as a input so that browser can read.

# How to define view function ??
=> by using def keyword in views.py file
=> sty:

```
from django.http import HttpResponse
def function_name(request):
    return HttpResponse("HTML code")
```

# How to call view function.
=> by defining url pattern in urls.py file.
=> sty:

```
from appname import views
path('url_name/' , views.function_name)
```

5) runserver and send request.

===
===
# M V T => (Template) => HTML file

#### General steps to create django project with view function and Template ###########

# T => Template	=> HTML file => Presentation Logic => FrontEnd

1) Django project
-> django-admin startproject commond is responsiable to create project in Django.

cmd: cd 'workspace/floder_name'

```
cmd: django-admin startproject project_name # create django project
cmd: cd project_name
```

2) Create Django Application.
-> manage.py file is responsiable to create django application.

```
cmd: py manage.py startapp app_name # create django application.
```

3) Register newly created django application in settings.py file.

```
INSTALLED_APPS = [
                        'app_name'
                ]
```

4) Q- where we will create HTML files ??
=> django bydefault serach templates folder in application folder.and HTML files are present in templates folder

steps:
1) right click on application folder and create new folder with the same name of templates.
2) right click on templates folder and create a new folder with same name of application folder
3) right click on app floder and create new HTML file

floder :
```
--> application folder # eg:->(demoapp)
        --> templates
                --> appname # eg:-> (demoapp)
                        --> HTML files # eg:-> (home.html)
```

5) Django Follows M V T Desgin pattern.so open views.py file and write some bussiness logic.

# V => View => views.py file => Bussiness Logic

# What is view ??
=> view is a function will take request as a input and return HttpResponse as a output to the end user.
=> view function accept user request and will provide output to the enduser.
=> HttpResponse will take HTML code as a input so that browser can read.

# How to define view function ??
=> by using def keyword in views.py file
=> sty:

```
            from django.shortcuts import render
            def function_name(request):
                return render(request , 'template_name(HTML file name)' , context = {})
```

Q- Use of render function??
=> render function will excute HTML file / render function will run HTML file

Q- How to pass data from views.py file to HTML file ??
=> by using context .. and it is a dict
=> eg : l = [10 , 20 , 30 , 40] # send this list on HTML file
=> render(request , template_name , context = {'data' : l})
=> {{ data }} --> access context data on HTML file by using dict key with {{ key }}

# How to call view function.
=> by defining url pattern in urls.py file.
=> sty:

```
            from appname import views
            path('url_name/' , views.function_name)
```

6) runserver and send request.

# application folder will contains apps.py file.
# project folder will contains settings.py
==

```
# views.py
from django.http import HttpResponse

def add(request , p , q):
    return HttpResponse(f'<h1> {p} + {q} = {p+q} </h1>')



def sub(request , p , q):
    return HttpResponse(f'<h1> {p} - {q} = {p-q} </h1>')




def area_of_circle(request , r):
    area = 3.14 * r * r
    return HttpResponse(f'<h1> Area of circle is {area} </h1>')
```

```
# urls.py
    path('add/<int:p>/<int:q>/' , views.add),
    path('sub/<int:p>/<int:q>/' , views.sub),
    path('circle/<int:r>/' , views.area_of_circle),

# localhost:8000/add/10/20/
# localhost:8000/sub/10/20/
# localhost:8000/circle/10
```

==
==

# How add to static files in my project ??
=> css , js , images , vdo

# sty to use static files :=>

```
# {% %} ->  code block

<head> {% load static %} </head>

{% static 'folder/filename' %} # sty to add static data
```

steps:

1) right click on application folder and create a new folder with name of static.
2) right click on static folder and create a new folder with the name of css , images
3) right click on css folder and create a new css file
4) link css file with HTML file using link

```
<head>
        <link rel="stylesheet" href="{% static 'css/filename.css' %}">
</head>
```

==
==

# There are three import statement ways

Q- What is module ??
=> any .py or any python file is called as module.

eg of math module:->

1) import a module by using import keyword.

```
import module_name
module_name.function()

import math
math.pow(10 , 2) # 100

import math as m
m.pow(10 , 2) # 100
```

2) import module specific functions by using from keyword.

```
	from module_name import func1 , func2 , func3
	func1()
	func2()

	from math import pow , sqrt , pi
	pow(10 , 2) # 100
```

3) import all module specific functions.

```
	from moule_name import *
	func1()

	from math import *
	pow(10 , 2) # 100
```

# Task:->
	we have student data in dict and we need to display that data on screen using table

db = {'jay' : {'name' : 'jay' , 'marks': 88 , 'roll_num' : 11} , 'kiran' : {'name' : 'kiran' , 'marks': 77 , 'roll_num' : 22}}

----

jinja2 template tags :->
	use :-> if i want to execute python expression(sty) in HTML file then we can use jinja2 template tags.

# How to print data in python:
	print()

# How to print data in HTML web page :
	{{ }} ==> print()

# python if-else :

```
	if condition:
		if-body
	else:
		else-body
```

# HTML if-else :

```
	{% if condiion %}
		if-body
	{% else %}
		else-body
	{% endif %}
```

```
# for loop in python :
        for temp_var in sequence:
                body

# for loop in HTML web page:
        {% for temp_var in sequence %}
                body
        {% endfor %}
```

==
===

# Django administration :=>
 -> inbuilt

Q- How to use use ??
=> by using '/admin/' url

Q- How to craete admin user ??
=> by using createsuperuser commond

cmd:

```
        py manage.py migrate

        py manage.py createsuperuser
        username('pc name') : -----
        email :---------
        passowrd :-------
        confirm_password :-------

        superuser created successfully.
```

==
===

# Model :=> M => Model   => models.py => Database Logic => BackEnd

NOTE:

-> By default django will use sqlite3 database.
  -> database settings are present in settings.py file

```
                DATABASES = {
                        'default': {
                'ENGINE': 'django.db.backends.sqlite3',
                'NAME': BASE_DIR / 'db.sqlite3',
                          }
                        }
```

-> one python Model class is equals to one database table.
  -> one model class will create one database table.

-> one field is equals to one column in database table.
    -> one field class will create one column in database table.

-> one object of model class is equlas to one record/data row in database table.
    -> creation of one object of modelclass will insert data into database table.

-> by default for each database table django will craete id column as a primary key.

-> all concepts OOP for and object is same in django as well.

# How to craete table in django ??
=> open models.py file

sty:
```
	class ClassName(models.Model):
		col_name1 = models.FieldType()
		col_name2 = models.FieldType()
```

Expl:
1) Table name will be appname_classname
2) one FieldClass == one column in database table and col_name1 and col_name2 will be the name of column.

eg:

```
	class Student(models.Model):
		name = models.CharField(max_length = 50)
		marks = models.FloatField()
		roll_num = models.IntegerField()
```

cmd:
```
		py manage.py makemigrations # use to generate sql code from python code
		py manage.py migrate        # refelect sql code on database.
```

admin.py file:
```
		# register your model here
		from appname.models import ClassName
		admin.site.register(ClassName)
```

==

# ORM :=> Object Reletional Mapping

# django API :=> To execute orm query we can use django api

Q- How to open django API ??
=> cmd :=>

```
                py manage.py shell
```

steps:

1) open shell
2) import modelclass

        from appname.models import classname

3) How to add data to the database table ??

    => by creating object of modelclass and save this object by using save method.

    sty:

    ```
            obj_ref = ClassName(args)
            obj_ref.save()
    ```

4) How to fetch all data from database table ??

    => by using all() method

    NOTE :-> all() method will return list of all objects present in the given database table.

    sty:

    ```
            var_name = classname.objects.all()
    ```

5) how to get a single object from database table ??

    => by using get() method

    NOTE : It will return single object from database

    sty:

    ```
            var = classname.objects.get(column_name = value)
    ```

6) How to update the existing data of database table ??

    => First get the object that you want to update and then update that object.

    sty:

    ```
            var_name = classname.objects.get(column_name = value)

            var_name.ColumnName = updated_value
            var_name.ColumnName = updated_value
            var_name.save()
    ```

7) How to delete the object from database table ??

    => First get the object that you want to delete. and using delete method delete that object.

```
            sty:
                    var_name = classname.objects.get(column_name = value)
                    var_name.delete()
```

----

eg:

```
C:\Users\santosh\Desktop\django\django projects 6PM\modelproject>py manage.py shell
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit (AMD64)] on
win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>>
>>>
>>> from modelapp.models import Student
>>>
>>> data  = Student.objects.all()
>>>
>>> data
<QuerySet []>
>>>
>>> s1 = Student(name = 'jay', marks= 99 , roll_num = 11)
>>> s1.save()
>>>
>>> s2 = Student(name = 'kiran', marks= 88 , roll_num = 22)
>>> s2.save()
>>>
>>> s3 = Student(name = 'pavan', marks= 77 , roll_num = 33)
>>> s3.save()
>>>
>>> s4 = Student(name = 'nayan', marks= 66 , roll_num = 44)
>>> s4.save()
>>>
>>> data  = Student.objects.all()
>>> data
<QuerySet [<Student: jay>, <Student: kiran>, <Student: pavan>, <Student: nayan>]>
>>>
>>> data[0]
<Student: jay>
>>>
>>> data[1]
<Student: kiran>
>>> data[2]
<Student: pavan>
>>>
>>> for x in data:
...     print(x)
...
jay
```

```
kiran
pavan
nayan
>>>
>>>
>>> for x in data:
...     print(f'student name is {x.name} student marks are {x.marks} and roll number is
{x.roll_num}')
...     print('=='*50)
...
student name is jay student marks are 99.0 and roll number is 11
==================================================================================
==============================
student name is kiran student marks are 88.0 and roll number is 22
==================================================================================
==============================
student name is pavan student marks are 77.0 and roll number is 33
==================================================================================
==============================
student name is nayan student marks are 66.0 and roll number is 44
==================================================================================
==============================
>>>
>>>
>>> data
<QuerySet [<Student: jay>, <Student: kiran>, <Student: pavan>, <Student: nayan>]>
>>>
>>>
>>> data[0]
<Student: jay>
>>>
>>> data[0].name
'jay'
>>> data[0].marks
99.0
>>> data[0].roll_num
11
>>>
>>>
>>>
>>> data  = Student.objects.all()
>>> data
<QuerySet [<Student: jay>, <Student: kiran>, <Student: pavan>, <Student: nayan>]>
>>>
>>>
>>> data[0].id
5
>>> data[0].name
```

```
'jay'
>>> data[0].marks
99.0
>>> data[0].roll_num
11
>>>
>>>
>>> jay = Student.objects.get(pk = 5)
>>> jay
<Student: jay>
>>>
>>> jay = Student.objects.get(name = 'jay')
>>> jay
<Student: jay>
>>>
>>> jay = Student.objects.get(roll_num = 11)
>>>
>>> jay
<Student: jay>
>>>
>>> jay.name = 'jay baba'
>>> jay.marks = 999
>>> jay.roll_num = 111
>>>
>>> jay.save()
>>>
>>>
>>> data  = Student.objects.all()
>>>
>>> data
<QuerySet [<Student: jay baba>, <Student: kiran>, <Student: pavan>, <Student: nayan>]>
>>>
>>> for x in data:
...     print(f'student name is {x.name} student marks are {x.marks} and roll number is
{x.roll_num}')
...     print('=='*50)
...
student name is jay baba student marks are 999.0 and roll number is 111
====================================================================
==============================
student name is kiran student marks are 88.0 and roll number is 22
====================================================================
==============================
student name is pavan student marks are 77.0 and roll number is 33
====================================================================
==============================
student name is nayan student marks are 66.0 and roll number is 44
```

```
====================================================================================================
>>>
>>>
>>>
>>> jay = Student.objects.get(roll_num = 111)
>>> jay
<Student: jay baba>
>>>
>>> jay.delete()
(1, {'modelapp.Student': 1})
>>>
>>> data  = Student.objects.all()
>>>
>>> data
<QuerySet [<Student: kiran>, <Student: pavan>, <Student: nayan>]>
>>>
>>> for x in data:
...     print(f'student name is {x.name} student marks are {x.marks} and roll number is {x.roll_num}')
...     print('=='*50)
...
student name is kiran student marks are 88.0 and roll number is 22
====================================================================================================
student name is pavan student marks are 77.0 and roll number is 33
====================================================================================================
student name is nayan student marks are 66.0 and roll number is 44
====================================================================================================
>>>
>>>
>>>
>>> my_data = Student.objects.filter(marks__gt = 70)
>>> my_data
<QuerySet [<Student: kiran>, <Student: pavan>]>
>>>
>>>
>>> my_data = Student.objects.filter(marks__lt = 70)
>>> my_data
<QuerySet [<Student: nayan>]>
>>>
>>>
>>> my_data = Student.objects.filter(name__endswith = 'n')
>>> my_data
<QuerySet [<Student: kiran>, <Student: pavan>, <Student: nayan>]>
>>>
```

```
>>>
>>> my_data = Student.objects.filter(name__startswith = 'k')
>>> my_data
<QuerySet [<Student: kiran>]>
>>>
>>> my_data = Student.objects.filter(name__contains = 'an')
>>> my_data
<QuerySet [<Student: kiran>, <Student: pavan>, <Student: nayan>]>
>>>
>>> my_data = Student.objects.filter(name__contains = 'ay')
>>> my_data
<QuerySet [<Student: nayan>]>
>>>
>>>
```

==

# Forms in Django :=>

-> There are two types of forms in django

1) HTML forms
2) ModelForms

-> use of forms:-> To accept user input in django.

# HTML form:

sty:

```
<form action = '' method = ''>
        body
        <button type = 'submit'> name_of_button </button>
</form>
```

expl:

when we submit a from then given action(url) and method will get executed.

-> action attribute contains url that url will call given view function.

-> method attribute : ['GET' , 'POST']

-> GET is use when i want to get the data from database / screen.

-> POST is use when i want to post/ add data to the database or screen.

NOTE:

GET is a default method

NOTE:

-> with post method always we have to use {% csrf_token %}

-> {% csrf_token %} is use for security.

==

# CRUD operations using DJango HTML forms:=>

# create urls.py file at application level

steps:

1) right click on application folder and create a new file with the name of urls.py
2)

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


urlpatterns = [
        path('index/' , views.index)
]
```

3) we need inculde application level urls.py file in project level urls.py file.

```
from django.urls import path , include

urlpatterns = [
        path('urlname/' , include('appname.urls'))
]
```

4) localhost:8000/<root_url>/<app_url>/

root_url => project level url
app_url => application level url

==

# How link our website with bootstrap.
=> www.getbootstrap.com
=> bootstrap
=> copy CDN from bootstrap and add to the webpage.within the head tag.

==

# NOTE:

1) To create object in database we have to use form.
2) To update the existing object we have to use form
3) for update and delete operation we are required a object ID/PK.

==
===

# Django model form :=>
-> One ModeFormClass is equals to one HTML form

steps:
1) right click on application folder and create a new file with the name of forms.py.

2) open forms file and import forms
```
from django import forms
```

3) define ModelFormClass
```
sty:
    from appname.model import ModelClassName

    class ModelFormClassName(forms.ModelForm):
        class Meta:
            model = ModelClassName
            fields = ['colname1' , 'colname2', etc]
```

4) open views.py file and import ModelFormClassName

5) define a view function and create object of ModelFormClassName

```
def function_name(request) :
    form = ModelFormClassName()
    return render(request , 'template_name' , {'form' : form})
```

# To style django model forms add crispy forms
```
# cmd:
    pip install django-crispy-forms

# settings.py :
    # INSTALLED_APPS = [
        'crispy_forms',
    ]

    # CRISPY_TEMPLATE_PACK = 'bootstrap4'

# follow below link :->
    https://django-crispy-forms.readthedocs.io/en/latest/install.html

# HTML :=>
    <head>
        {% load cripsy_forms_tags %}
    </head>
```

```
                    <form>
                              {{form | crispy }}
                    </form>
```

==
=======================================
# User Registration System and Login Logout System IN DJango :=>

Q- How to register a user ??
=> by using ModelForm

steps:
1) open forms.py file and import UserCreationForm

```
                    from django.contrib.auth.forms import UserCreationForm
```

2) Create Object of UserCreationForm and send it to HTML file.

```
          def view_fun(request):
                    obj = UserCreationForm()
                    return render(requets , "template_name" , {'form' : obj})
```

3) define url pattern.

==
===================================
# Login Logout System In DJango :=>

1) open project level urls.py file

```
          from django.contrib.auth import views as auth_views
          urlpatterns = [
                    path("login/" ,
auth_views.LoginView.as_view(template_name="userapp/login.html")) ,
                    path("logout/" , auth_views.LogoutView.as_view()),

          ]
```

2) Login.html

```
   <form action="" method="POST">{% csrf_token %}
      {{ form.as_p }}
      <button type="submit"> LOGIN </button>
   </form>
```

3) settings.py

```
          LOGIN_REDIRECT_URL = '/user/home/'

          LOGOUT_REDIRECT_URL = "/login/"
```