**task=1**

**multithreading and multitasking :**

**1st=multi tasking**

multi-tasking is a general word executing several tasks simultaneously is the concept of multitasking ex; kitchen in mother 1st in prepare to cook on one side and another side cutting on vegetables and one side calling in a mobile and preparing to lunch box this is an so many tasks to prepare at a time this is called multi-tasking each task is a separate independent process

**task=2**

**multi threading :**

multiple independent parts of simultaneously which known as thread a thread is a unit of execution on concurrent programming where each task is an independent part of the same program ex; there is a 1 program

Thread1

Thread2

Thread3

is 1 program in 3 threads

how many programs are there in 1 program how many independent threads are there 3 where each task is one independent same program there was a python program in checked in line by line but threading is divided into 3 parts so the program is executed in less time what is the thread; a flow of execution is considered as a thread it is a python object most commonly used in the concept of application level multitasking concept at the operating concept level multi-threading concept is used in ; multimedia, lofing, animation, video games, the server implementation

3 ways of multi-threading

1st; Thread without using any class

2nd: thread by extending thread class

3rd: thread without extending thread class

**Import threading #threading .current- thread**

Import threading import #current thread

def display ():

for I in range (10):

print (‘child thread’)

t= thread () # by creating thread object

t= thread (target=display) #creation of thread object to execute display

t.start ()

for I in range (10) :

print (‘main thread’)

threading is a module name

thread is a class name

module is threading

main threading

**import threading**

print(“current executing threading : ,threading.current-thread () .get name ()”) there are 2 functions in the current thread (). Get name ()

from time import sleep

from threading import \*

class Hello(Thread):

def run(self):

for I in range(10):

print('Hello')

sleep(1)

class Hi(Thread):

def run(self):

for I in range(10):

print('Hi')

sleep(1)

t1=Hello()

t2=Hi()

t1.start()

sleep(0.5)

t2.start()

t1.join()

t2.join()

print('bye'...

**#from time import sleep**

**from threading import \***

**class Book(Thread):**

**def run(self):**

**for i in range(10):**

**print('Book')**

**sleep(1)**

**class Pen(Thread):**

**def run(self):**

**for i in range(10):**

**print('Pen')**

**sleep(0.5)**

**class Pencil(Thread) :**

**def run(self):**

**for i in range(10):**

**print('Pencil')**

**sleep(1)**

**t1 = Book()**

**t2 = Pen()**

**t3 = Pencil()**

**t1.start()**

**time.sleep()**

**t2.start()**

**t3.start()**

**t1.join()**

**t2.join()**

**t3.join()**

**print('bye')...**

we can not exactly output but the question is why means threading means is concatenation at a time. so threads are same time coming to the main thread

1. Multitasking users are allowed to perform many tasks by CPUMultithreading executes many threads simultaneously This is the main concept of multithreading

**2nd task Django**

Django is a web application development of the framework Using Django is a development complete web application by doing all the server-side programming And we run every application as a Django server Django is a python framework that makes it easier to create websites using python Django is especially helped fully for database-driven websites

**How does Django work**

Django follows the MVT design pattern M=MODEL V=VIEW

**T=TEMPLATE**

The model provides data from to database The common way to extract data from a database is SQL The models are usually 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 view is usually view .py

**templates**

templates are often HTML files with HTML CODE describing the layout of a web page but we will concentrate on HTML files the templates of an application are located in a folder named templates

**URLS**

Django also provides a way to navigate around the different pages in a websites When a user requests a URL. Django decides which view it will send it to this is done in a file called URLS.PY

**VIRTUAL ENVIRONMENT**

It is suggested to have a dedicated virtual environment for each Django project, and one way to manage a virtual environment is even, which is included in Python. With venv, you can create a virtual environment by typing this in the command prompt, remember to navigate to where you want to create your project: Install Django ; Finally, we can install Django. Remember to install Django while you are in the virtual environment! Django is installed using pip, with this command: Windows: (myproject) C:\Users\Your Name>py -m pip install Django Unix/macOS: (myproject) ... $ python -m pip install Django

**DJANGO PROJECT …..**

Introduction...

Getting started...

Create folder …

Create project…

Create app…

Next...

To views…

URLs… Setting Templates Models

TEMPLATES... Variables Template tags If else For loop Comment

STATIC FILES... Add CSS files Add js file Add image

QUERY SETS ….

**MODELS:**

Models we can define in class function Different types of variants in we can define This function is a python function this function changeable in SQL format JSON files used in this method this called method as makemigrations makemigrations changes to the format Migrate create a table format

1. Django receives the URL, checks the URLS.PY file, and calls the view that matches the URL.
2. The view, located in VIEW.PY, checks for relevant models.
3. The models are imported from the MODEL. PY, file.
4. The view then sends the data to a specified template in the TEMPLATE folder.
5. The template contains HTML and Django tags, and with the data, it returns finished HTML content back to the browser. Django can do a lot more than this, but this is basically what you will learn in this tutorial, and are the basic steps in a simple web application made with Django.

**My First Project ;**

Once you have come up with a suitable name for your Django project, like mine: MYWORLD, navigate to where in the file system you want to store the code (in the virtual environment) and run this command in the command prompt: django-admin startproject myworld Django creates an MY WORLD folder on my computer, with this content: my world

manage.py

my world/

\_\_init\_\_.py

asgi.py

settings.py

urls.py

wsgi.py

These are all files and folders with a specific meaning, you will learn about some of them later in this tutorial, but for now, it is more important to know that this is the location of your project and that you can start building applications in it.

**Run the Django Project**

Now that you have a Django project, you can run it, and see what it looks like in a browser. Navigate to the MY WORLD folder and execute this command in the command prompt:

**py manage.py runserver**

Which will produce this result: Watching for file changes with StatReloader Performing system checks...

The system check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, content types, and sessions. Run 'python manage.py migrate' to apply them. December 02, 2021 - 13:14:51 Django version 3.2.9, using settings 'my world.settings' Starting development server at http://127.0.0.1:8000/ Quit the server with CTRL-BREAK. Open a new browser window and type 127.0.0.1:8000 in the address bar. The result: The project is run in cmd or pycharm cmd :

Firstly chose the c drive d drive

Create virtual environment

For creating directory

For creating a folder

Then open the folder and create a new project

It means the project name

Then creating a project app

This means another project name

This is the project cmd mode process

and next move a Django project

project files create a Folder the Folder open to the shows in

This type of app in showing

**And the open in settings:**

App name is ;ex; shankar ; Shankar is open to setting in :” Shankar”

**And then next views :**

Import I Django From Django http import HTTP Response ; This means in HTTP is capital letters 2nd HTTP And creating in function anything Def display (request) : Return HTTPResoponce <h1>shankar<h2> ; Shankar's name is created in the app name

**And then moved on next URLs:**

URLs in import in views path From Django URLs. import path From app to import to views URLs pattern Path (‘hello’/.views. hello

**And next move on templates ;**

Open a new directory right click on the open template 2nd also opens a new template And right click on an HTML page And then configure HTML to (write a logic) And next to setting and open Import OS. Base DIR.

1. Dir (OS.path.join//base dir ‘shankar’)  
   Installed apps DIR template DIR

Next move on VIEWS.

Create path.

Then complete of process.

And moved on command

Python mange.py run server

Then open the code

The first project is complete

**MIDDLE WARE OF DJANGO :**

Middleware is a framework of hooks Django request/response processing It s a light, low-level “plugin” system for globally altering Django input or output Each middleware component is responsible for doing some specific functions for example Django includes a middleware component AUTHENTICATION MIDDLEWARE

**WHAT IS THE GET METHOD IN DJANGO:**

Get is an HTTP method in Django that encapsulates the data in a string and utilizes it to construct a URL . the URL includes the data keys and values as well as the address to which the data must be transmitted

**WHAT IS THE GET METHOD USED FOR :**

Get is used to request data from specified resources. some notes on get request: Get requests can be cached. get a request to remain in the browser history

**What is requested in Django:**

you will learn how to get data from getting requests in Django. when you send a request to the server, you can also send some parameters. generally, we use a get request to get some data from the server. we can send parameters. with the request to get some specific data.

**What is the get and POST method in Django:**

**Get and post:**

The Django login form is returned using the post method. in which the browser bundles up the form data, encodes it for transmission, sends it to the server, and then receives its response, GET, by contrast, bundles the submitted data into a string. and uses this to compose a URL

**DJANGO PROJECT STRUCTURE :**

. ├── apps

│ └── app\_1

│ ├── admin.py

│ ├── apps.py

│ ├── **init**.py

│ ├── migrations

│ │ └── **init**.py

│ ├── models.py

│ ├── tests.py

│ └── views.py

├── django\_project

│ ├── **init**.py

│ ├── **pycache**

│ │ ├── **init**.cpython-35.pyc

│ │ └── settings.CPython-35.pyc

│ ├── settings.py

│ ├── urls.py

│ └── wsgi.py

├── manage.py

├── media

├── static

└── templates

**MANAGE. PY**

This file is used basically as a command line utility and for deploying, debugging, or running our web application This file contains code for the run server, or makemigrations or migrations, etc that we use in the shell. anyway, we do not need to make any changes to the file.

**\*RUN SERVER :**

this command is used to run the server for our web application

**MIGRATION:**

This is used for applying the changes done to our models into the databases. That is if make any changes to our database then we use migrate command This is used for the first time we create the database

**MAKE MIGRATION:**

This is code done to apply new migrations that have been carried out due to the changes in the databases

**WHAT IS DEBUG TRUE AND FALSE IN DJANGO:**

I first assumed DBUG=TRUE turned on Django s logging capability and middleware for error reporting but now understand how the Django intrenal system works differently under the boolean settings Functionally, there are no differences. However, DEBUG defines whether the error message should be shown to the user at the browser level (DEBUG=True) v/s sends an email to admins (DEBUG=False with some settings.)

**FILTER AND GET DIFFERENCE IN DJANGO:**

Get is only 1 value returned If and get value is 0 and multiple values are in there the result is ERROR FILTER is an 0 and multiple values (2,3,4) is a returned

**3rd TASK**

**SERIALIZATION :**

**What is meant by serialization:**

Serialization is the process of converting a data object into a combination of code and data represented within a region of data storage The process of converting an object .from a python supporter form to a file support form is this called serialization

**DE SERIALIZATION :**

Deserialization means the file-supported form is converted into the python supported form this is called deserialization This is the reverse process of serialization

Employe:number=100 === this is EMPLOYE OBJECT

Employe : name=ram

Employe: salry=1000

Employe: id =112

write[RU1] ==== serialization & marshaling

pickling

THIS IS AN OBJECT FILE FORM OR NETWORK SUPPORT OBJECT

THIS IS PYTHON OBJECT  
READ == DESERILIZATION & UNMARSHILING ==

UNPICKLING

PYTHON OBJECT IS NOT CONVERTING IN JAVA APP AND MOBILE APP Python object is not which can not understand by java application and os applications We are required to convert files in python files to java files is a used for JSON, YAML, files JSON means java script object notation JSON advantage to understand java, mobile, applications

**What is JSON Django:**

in this article. we will see how to add JSON fields to our Django models. JSON is a simple format to store data in key and value format. it is written in curly braces. many a time on a developer website, we need to add developer data and JSON fields are useful in such cases. first, create a Django project and an app python object form to JSON object form converted into a called SERIALIZATION JSON object form to python object form converted to its called as DESERILIZATION

**PICKLE:**

Pickle is a module Pickle in python is primarily used in serializing and deserializing a python object structure in other words it s the process of converting a python object into a byte stream to store it in a file /database. maintain program state across the session or transport data over the network Pickle function we can perform serialization By using a function in DUMP(): Which using the function we can deserialization by using a function in LOAD(): We are known as unpickling E=employe (100,’ ram’,10000,’hyd’)

With open (‘emp. ser’ ‘wb’) as f:# writing binary

Pickle.dump(e,f) #serilization

With open (‘emp. ser’,’ be) as f :

Emo-boy = pickle.load (f)

Binary data=not readable form

**Import pickle**

CLASS EMPLOYE :

Def \_\_init \_\_ (self, eno ,name, esal,addr):

Self. no = eno

Self. name = name #instance variables

Self. esal = seal

Self. eaddr = eaddr

Def display (self):

Print (‘E NO : { } , E NAME : { } ,E SAL : { }, E ADDR : { } ‘ . FORMAT ( self. E no, self . ename ,self . esal , self. eaddr))

E = employee ( 100, ‘ ram ‘ , 10000, ‘ hyd ‘ )

With open (‘emp . ser,’ wb ‘) as f :

Pickle. dump ( e, f )

Print ( ‘ pickling of employee object completed ‘ )

[RU1]