

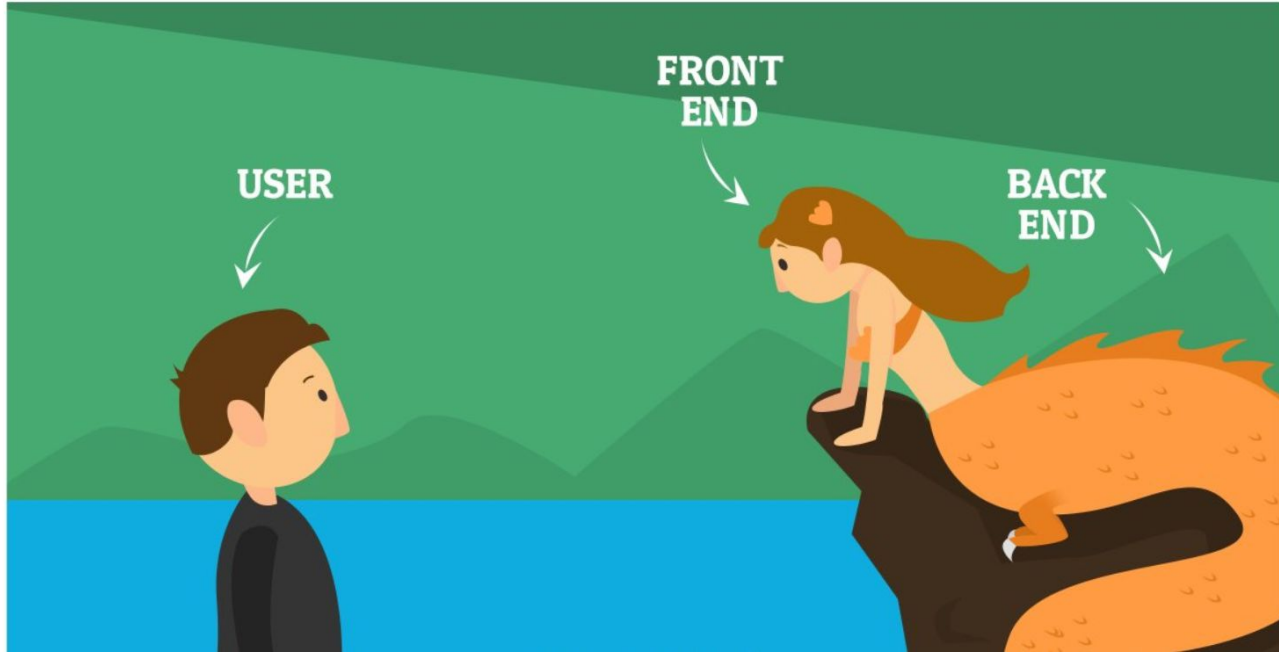
INTRODUCTION TO DJANGO

Computer Coding Club MNNIT

AGENDA

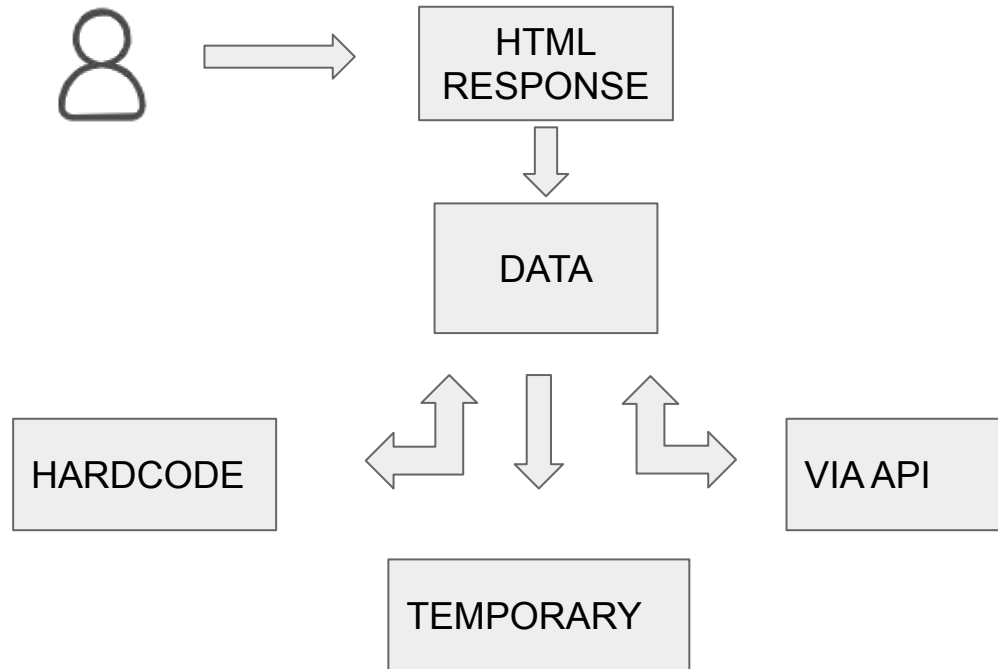
- Backend and why?
 - Static vs Dynamic vs Active Pages
 - Brief about Django
 - Python and VirtualEnv
 - Project Startup
 - Understanding file structure
 - Routing and Views
 - Templating
-

BACKEND?



Any logic/code that is hidden from client

WHY BACKEND?



LET'S ANALYSE

SAME URL BUT DIFFERENT RESPONSE

https://academics.mnnit.ac.in/new

Home Academic Calendar Downloads

Personal Details

NAME DIVYANSH UPADHYAY

REGISTRATION NO. [REDACTED]

FATHER'S NAME [REDACTED]

DATE OF BIRTH [REDACTED]

GENDER MALE

CATEGORY General

https://academics.mnnit.ac.in/new

TPO Dash ICAI Login

Home Academic Calendar Downloads

Upload

Personal Details

NAME ANSHUMAN CHOWDHARY

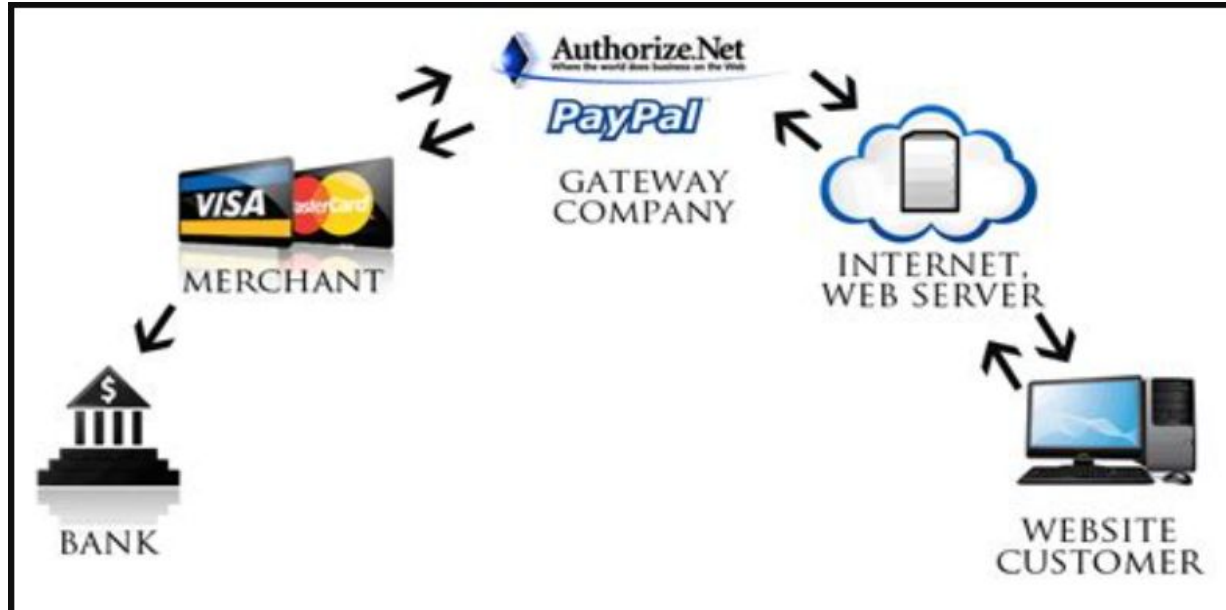
REGISTRATION NO. [REDACTED]

FATHER'S NAME [REDACTED]

DATE OF BIRTH [REDACTED]

GENDER MALE

LET'S ANALYSE



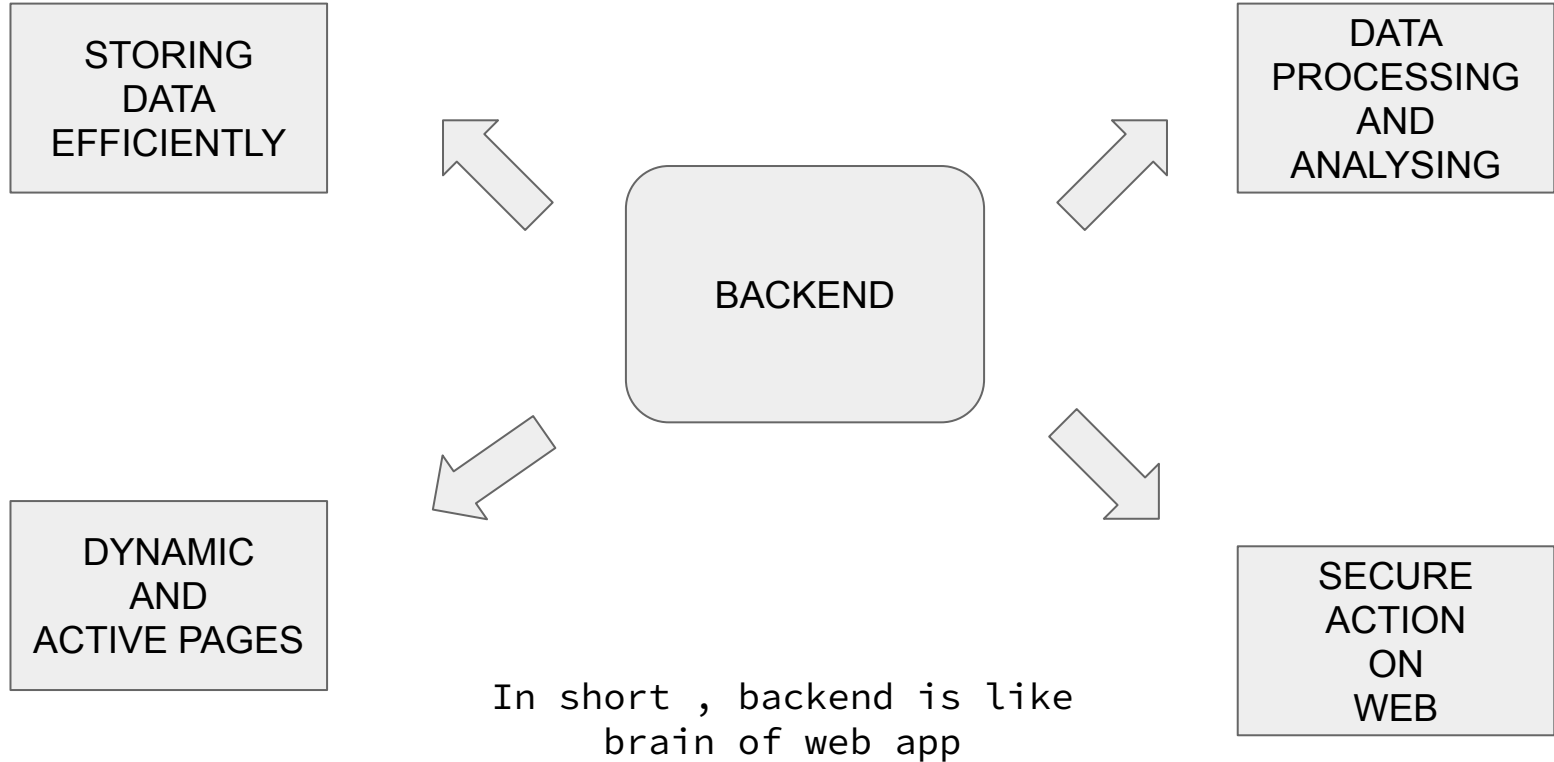
IDENTIFY CORRECT PARTY AND HAVE SECURE PAYMENT GATEWAY

LET'S ANALYSE



Millions of different results,
how is each html page is added
for so many queries?

WHY BACKEND?



STATIC WEB PAGES

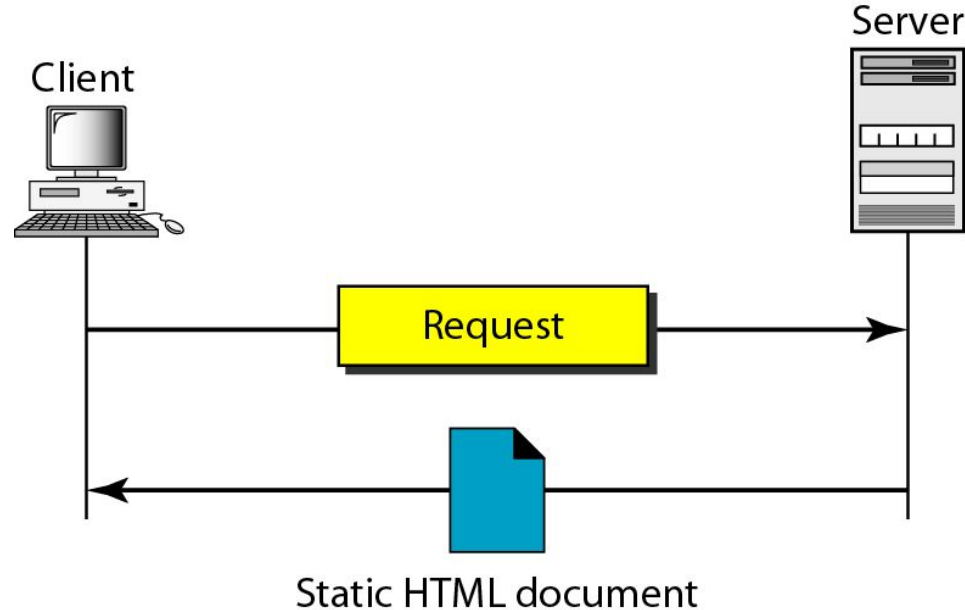
The contents of web page remains same for every client.
Suitable for never or rarely updated content.

Advantages :

- Quick and Cheap to develop.
- Cheap to host.
- Loads very quick.

Disadvantages :

- Requires a web developer to modify.



DYNAMIC WEB PAGES

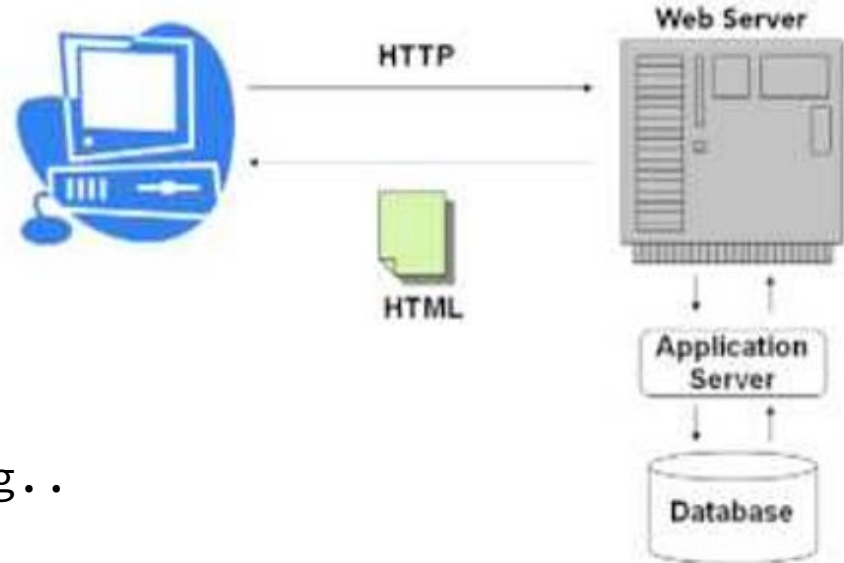
The contents are customized according the requesting user. The server creates a fresh **document** for each incoming request.

Advantages :

- Easy to restructure and manage.
- Use of database as a store.

Disadvantages :

- Take longer to load.
- More work and cost in designing..



ACTIVE WEB PAGES

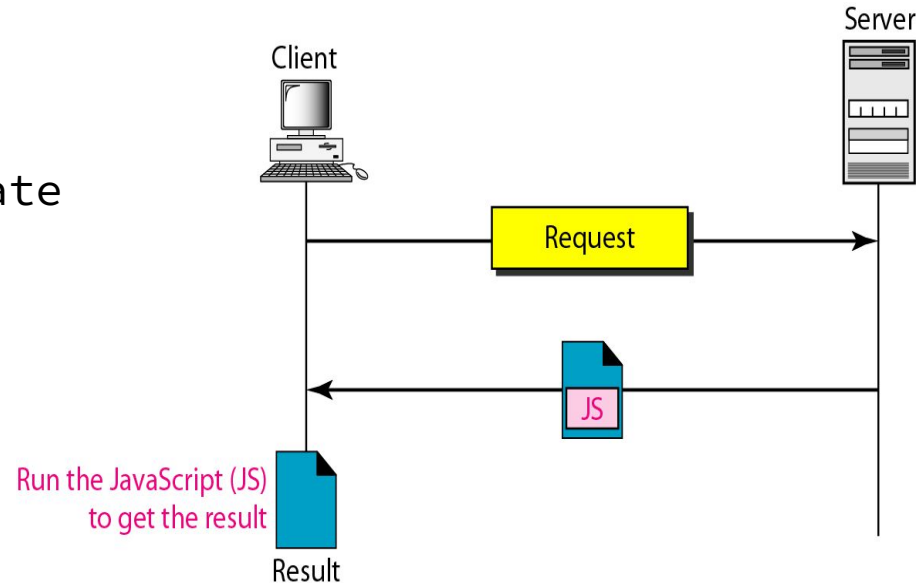
The computer program (JS or applet) is sent to the client browser to run locally, which interacts and changes display continuously.

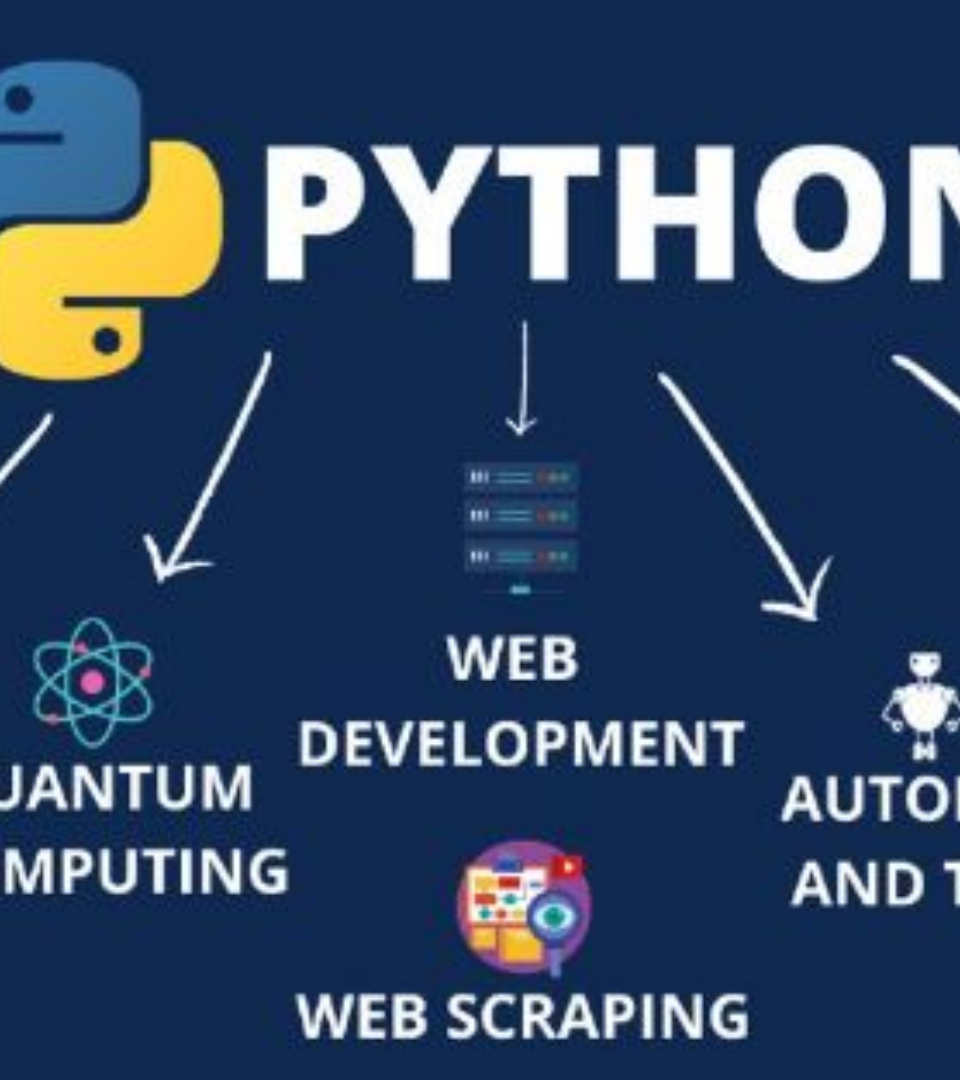
Advantages :

- Ability to access sources of information directly and update web page continuously.

Disadvantages :

- Program should be platform independent.
- Potential risk.






PYTHON

The python is a general purpose programming language and it's very versatile.

Vast number of libraries and vast community support

A close-up of Morpheus from the movie The Matrix, wearing his signature black sunglasses and a serious expression. The image is framed with a thick black border.

WHAT IF I TOLD YOU

PYTHON IS FUN

C++: Can not compare
float and int
Python:



Python is the
easier language
to learn.
No brackets,
no main.



You get errors
for writing an
extra space



**When you try to
define constants**



WHERE TO START?



- Go to [Python Website](#) install and setup python.
- Get GitHub Students Pack and install PyCharm Premium from [here](#).
- If are stuck somewhere , you can refer to [it](#).
- To check whether python is installed or not open cmd/terminal and type in “python -version”.

```
Microsoft Windows [Version 10.0.22000.978]
(c) Microsoft Corporation. All rights reserved.

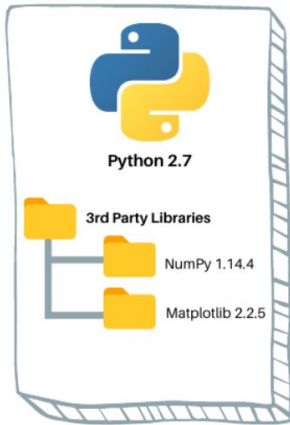
C:\Users\leolo>python --version
Python 3.10.1
```

VIRTUAL ENVIRONMENT

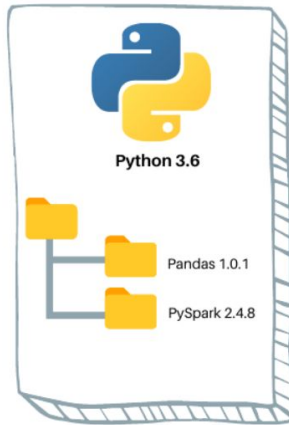
WHAT IS VIRTUAL ENV?

A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them.

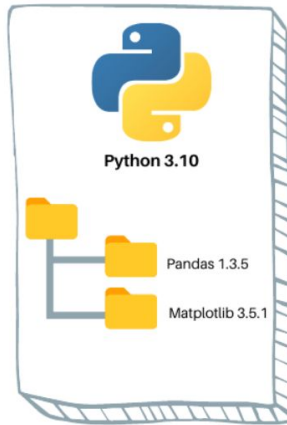
Virtual Environment 1



Virtual Environment 2



Virtual Environment 3



VIRTUAL ENVIRONMENT

CREATING VIRTUAL ENV

- `pip install virtualenv`
- Windows: `python -m virtualenv [venv]`
- Linux: `virtualenv [venv]`

ACTIVATING VIRTUAL ENV

- Windows: `./[venv]/Scripts/activate`
- Linux: `source /[venv]/bin/activate`

DEACTIVATING VIRTUAL ENV

- Windows: `deactivate`
- Linux: `deactivate`



VIRTUAL ENVIRONMENT



Why do we need
it?

- Creates an isolated python environment with its own site package
- Install whatever you want without fouling anything else up
- Maintain dependencies for different projects
- It creates a folder which contains all the necessary executables to use the packages that a python project needs.

INTRODUCING...DJANGO

Django is a high-level Python web **framework**.that encourages rapid development and clean, pragmatic design.

FEATURES

- Ridiculously fast
- Reassuringly secure
- Exceedingly scalable
- Open Source
- Lots of extras cool stuffs like - middleware, csrf protections, sessions, caching, ORM, authentication etc.

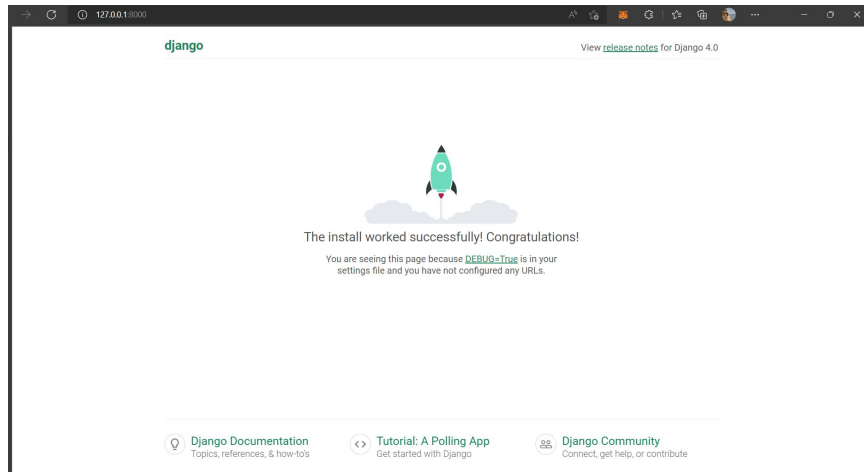
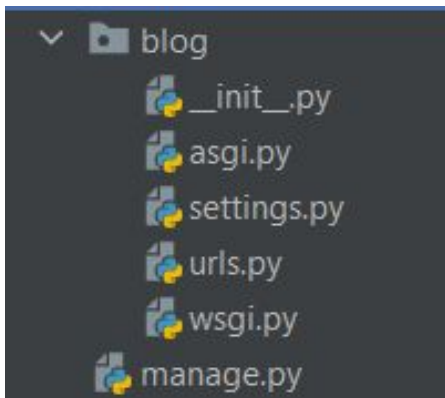
DJANGO IN ACTION

Can you name them all?



SETTING UP PROJECT

- Activate your virtual environment
- Run “pip install django” in cmd/terminal
- Run “django-admin startproject [project-name]”
- Open the project folder in an IDE
- If you are able to see the following file structure, then it's awesome. Now run python manage.py runserver and open up browser



WHAT IS GENERATED?

- `settings.py` -> Contains configuration of the project, secrets, middleware, apps etc.
- `manage.py` -> Scripts to interact with Django via terminal.
- `urls.py` -> file managing url mapping of the project
- `wsgi.py` -> used for deployment sync
- `asgi.py` -> used for deployment sync and async
- `init.py` -> to make Python treat directories containing them as packages.

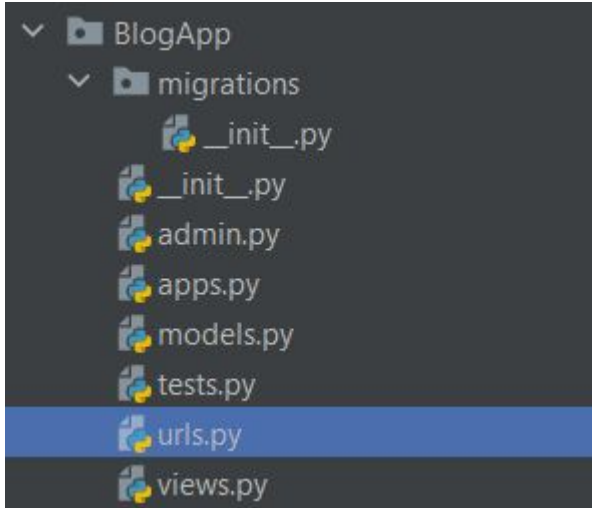
APPS IN DJANGO

An application is an web application that does something or carries out a specific task.

Used to break down the project into smaller manageable modules.

Run “python manage.py startapp [appname]” to create an app

INTO AN APP



- migrations-> Is a way of making changes into database schema(covered in next class).
- admin.py-> used to display models on admin panel.
- apps.py-> to include any configuration of the application.
- models.py-> for creating schema in ORM fashion
- tests.py-> unit and integration tests are written here
- urls.py-> url mapping for the application
- views.py -> contains logic which is responsible for rendering template

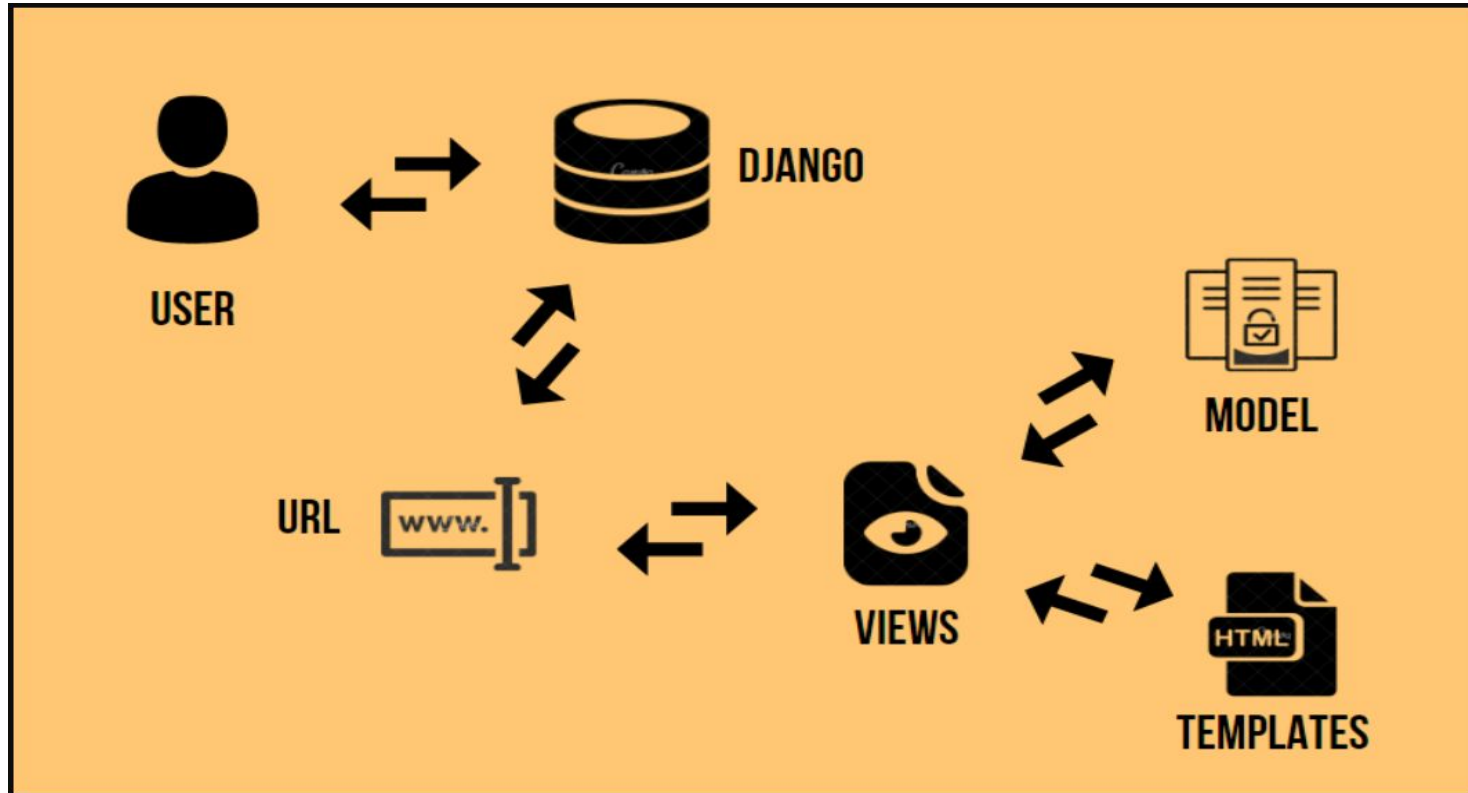
**urls.py to be created manually

MVT-ARCHITECTURE

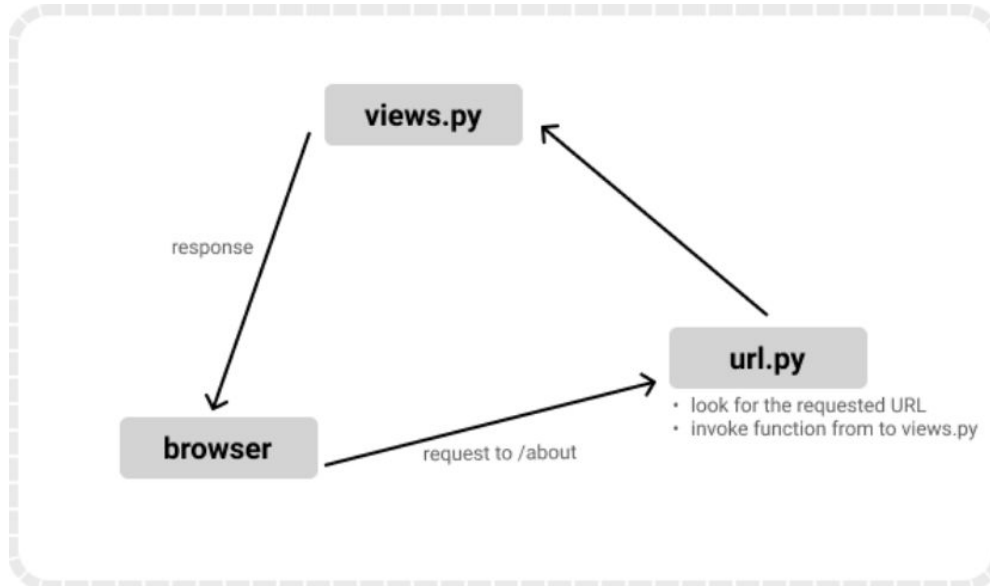
MVT is a design pattern in which a software is divided into 3 components - Models, View and Templates.

- Model-> The model acts as interface of your data. It is responsible for maintaining the data.
- View-> Responsible for executing logic and interact with model to render a template.
- Template->Responsible for user interface i.e. what we see on browser when resources are loaded.

MVT-ARCHITECTURE



ROUTING



Example of a path: `books/harrypotter/volume1`

ROUTING

Which HTML page should load on `academic.mnnit.ac.in/new?`

Which HTML page should load on `academic.mnnit.ac.in/about?`

This is handled via routing in Django.

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

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

ROUTING

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

This means that there is an app `goals`, which has a file called `urls.py`, whose `urls` need to be included whenever we are hitting the endpoint `'goals/'`.

DYNAMIC ROUTING

```
urlpatterns = [  
    path('product/<int>', views.product),  
]
```

An integer stands at the end of the endpoint. This can be used to provide unique information about the web page requested.

For example, on an E-commerce website, we have different products displayed. On clicking on one of them, it can be directed to the URL - amazon.com/product/1. Now for all 'product/<int>' type URL, we will have the same view handling it.

VIEWS

Django views are Python functions that takes http requests and returns http response, like HTML documents.

Views are written in a file called views.py.

Example:

```
from django.http import HttpResponseRedirect

def hello(request):
    text = '<p> Welcome to the first class of Django </p>'
    return HttpResponseRedirect(text)
```

EXAMPLES OF RESPONSES

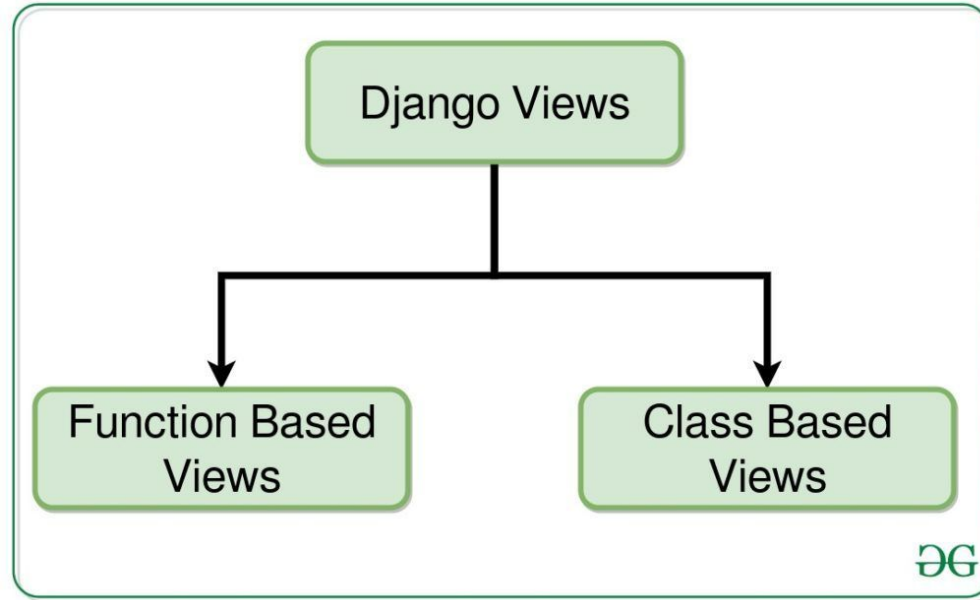
```
1.def myrandomtext(request):  
    return HttpResponse("Random - " + str(random()))
```

This would display an HTML page with content “Random – ndfkjmfkmkf”.

```
2.def myrandomtemplate(request):  
    return render(request, 'game.html')
```

This will load an html page defined in the file/template: game.html.

VIEWS



We have discussed Function based views.
Class based views is your homework:)

TEMPLATES

- A convenient way to generate dynamic HTML pages.
- A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.
- Promotes the Concept of DRY(don't repeat yourself).

TEMPLATES

- Variables

- `{{ variable }}`
- evaluates variable and replaces with result
- alphanumeric characters and underscore only
- dot notation to access attributes of a variable

TEMPLATES

- Filters
 - modify variables for display by using filters
 - apply filters with a pipe (|), filters can be chained
 - examples:
 - `{{ list|join:", " }}`
 - `{{ value|length }}`
 - `{{ value|striptags }}`
 - `{{ name|lower }}`

TEMPLATES

- Tags
 - More Complex than variables
 - Can be used for useful tasks
 - General Syntax `{% tag %}`
 - examples:
 - `{% for %} {% endfor %}`
 - `{% extends %}`

TEMPLATES

- Tags-> Conditionals
 - For Conditionals-> `{% if %}{% elif %}{% endif %}`

```
{% if athlete_list %}
    Number of athletes: {{ athlete_list|length }}
{% elif athlete_in_locker_room_list %}
    Athletes should be out of the locker room soon!
{% else %}
    No athletes.
{% endif %}
```

TEMPLATES

- Tags-> Looping
 - For loop -> {% for %}{% endfor %}

```
<ul>
  {% for athlete in athlete_list %}
    <li>{{ athlete.name }}</li>
  {% endfor %}
</ul>
```

TEMPLATES

- Tags-> Extends and Include
 - `{% extends 'template' %}`-> It inherits the code from template. Generally used to define child parent relationship.
 - `{% include 'template' %}`->It imports a specific html snippet into your file. Used for breaking html code into reusable components.

I'M LEARNING DJANGO

A man with a serious expression, wearing a brown fur hat and a dark coat with a fur collar, stands in a doorway. The image is framed by a thick black border.

**SEND ME ALL YOUR
RESOURCES**

THANK YOU



That's all Folks!