**Django**

**Explain how a request is processed in Django?**

In case some user requests a page from some Django powered site, the system follows an algorithm that determines which Python code needs to be executed. Here are the steps that sum up the algorithm:

1. Django first determines which root URLconf or URL configuration module is to be used
2. Then, that particular Python module is loaded and then Django looks for the variable urlpatterns
3. These URL patterns are then run by Django, and it stops at the first match of the requested URL
4. Once that is done, the Django then imports and calls the given view
5. In case none of the URLs match the requested URL, Django invokes an error-handling view

## ****How to use file-based sessions?****

In order to make use of file-based sessions, you will need to set the SESSION\_ENGINE setting to “django.contrib.sessions.backends.  
file”.

## ****Give the exception classes present in Django.****

* AppRegistryNotReady
* ObjectDoesNotExist
* EmptyResultSet
* FieldDoesNotExist
* MultipleObjectsReturned

## ****Does Django support multiple-column Primary Keys?****

No. Django only supports single-column Primary Keys.

Pros & Cons of Function-Based Views:

some pros and cons about function-based views and class-based views.

**Pros:**

* Simple to implement
* Easy to read
* Explicit code flow
* Straightforward usage of decorators

**Cons:**

* Hard to extend and reuse the code
* Handling of HTTP methods via conditional branching

Class-Based Views

**Pros:**

* Can be easily extended, reuse code
* Can use O.O techniques such as mixins (multiple inheritance)
* Handling of HTTP methods by separate class methods
* Built-in generic class-based views.

**Cons:**

* Harder to read
* Implicit code flow
* Hidden code in parent classes, mixins
* Use of view decorators require extra import, or method override

**Django Mixin:**

t is a type of multiple inheritance which allows classes in Python to share methods between any class that inherits from that mixin. It is used when we want to implement a specific functionality in different classes.

**Class based view:**

<https://docs.djangoproject.com/en/3.0/topics/class-based-views/intro/>

**Django RestFramework**

**View Sets:**

Django REST framework allows you to combine the logic for a set of related views in a single class, called a ViewSet. In other frameworks you may also find conceptually similar implementations named something like 'Resources' or 'Controllers'.

A ViewSet class is simply **a type of class-based View, that does not provide any method handlers** such as .get() or .post(), and instead provides actions such as .list() and .create().

The method handlers for a ViewSet are only bound to the corresponding actions at the point of finalizing the view, using the .as\_view() method.

Typically, rather than explicitly registering the views in a viewset in the urlconf, you'll register the viewset with a router class, that automatically determines the urlconf for you.

There are two main advantages of using a ViewSet class over using a View class.

* Repeated logic can be combined into a single class. In the above example, we only need to specify the queryset once, and it'll be used across multiple views.
* By using routers, we no longer need to deal with wiring up the URL conf ourselves.

**ViewSet Action:**

The default routers included with REST framework will provide routes for a standard set of **create/retrieve/update/destroy** style actions,