1. Template View in Django :

views.py > ...

```python
from django.http import HttpResponse
from django.shortcuts import render, redirect
from book.forms import BookStoreForm
from book.models import BookStoreModel
from django.views.generic import TemplateView, ListView, DetailView
from django.views.generic.edit import FormView, CreateView, UpdateView, DeleteView
from django.urls import reverse_lazy
from django.http import HttpResponse
```

```python
urlpatterns = [
    # Normal Functional View :
    # path('', home, name="homepage"),

    # Shortcut View :
    path('', views.TemplateView.as_view(template_name='home.html'), name='homepage'),

    # Class Based View :
    path('<int:roll>/', views.MyTemplateView.as_view(template_name='home.html'), {'author': "Muktadir"}, name='homepage'),
    # List View
    path('show_books/', views.BookListView.as_view(), name='show_books'),
    # Detail View
    path('show_books/book_details/<int:id>', views.BooksDetailView.as_view(), name='book_details'),
    # Form View
    path('store_new_book/', views.BookFormView.as_view(), name='store_book'),
    # Update View
    path('edit_book/<int:pk>', views.BookUpdateView.as_view(), name='edit_book'),

    # path('store_new_book/', views.store_book, name='store_book'),
    # path('edit_book/<int:pk>', views.edit_book, name='edit_book'),
    path('delete_book/<int:pk>', views.DeleteBookView.as_view(), name='delete_book'),
]
```

```python
# Class Based View
class MyTemplateView(TemplateView):
    template_name = 'home.html'
    def get_context_data(self, **kargs):
        context = super().get_context_data(**kargs)
        context = {'name': 'Muktadir', 'age': 23}
        context.update(kargs) # Updating Dictionary
        print(kargs)
        print(context)
        return context
```

2. ListView in Django :

```python
# Class Based List View --->
class BookListView(ListView):
    model = BookStoreModel
    template_name = 'show_books.html'
    context_object_name = 'data' # data from the specific model
    ordering = ['id']

    # Sorting Data : -- >
    # def get_queryset(self):
    #     return BookStoreModel.objects.filter(category='Thriller')

    # Passing With External Context : -->
    # def get_context_data(self, **kargs):
    #     context = super().get_context_data(**kargs)
    #     context = {'Rahim': BookStoreModel.objects.all().order_by('book_name')}
    #     return context

    # Overwrite Templte : -->
    def get_template_names(self):
        if self.request.user.is_superuser:
            template_name = 'superuser.html'
        elif self.request.user.is_staff:
            template_name = ''
        else:
            template_name = self.template_name

        return [template_name]
```

3. DetailView in Django :

```python
# Class Based Detail View --->
class BooksDetailView(DetailView):
    model = BookStoreModel
    template_name = 'book_details.html'
    context_object_name = "item" # for accessing the data in template
    pk_url_kwarg = 'id'
```

4. FormView in Django :

```python
#Class Based Form View --->
class BookFormView(FormView):
    template_name = 'store_book.html'
    form_class = BookStoreForm
    success_url = '/show_books/'
    success_url = reverse_lazy('show_books') # from url name

    def form_valid(self, form):
        print(form.cleaned_data)
        form.save() # One Way for save
        return redirect('show_books')
        # return HttpResponse('<h1> Form Submitted </h1>')
```

5. CreateView in Django :

```python
# Class Based Create View --->
class BookFormView(CreateView):
    model = BookStoreModel
    template_name = 'store_book.html'
    form_class = BookStoreForm
    success_url = reverse_lazy('show_books') # AUTO SAVE + REDIRECT
```

6. UpdateView and DeleteView in Django :

```python
# Edit / Update View :
class BookUpdateView(UpdateView):
    model = BookStoreModel
    template_name = 'store_book.html'
    form_class = BookStoreForm
    success_url = reverse_lazy('show_books')
```

7. Introduction to Cookie in Django :

```python
# COOKIE ------> > > > >
def home(request):
    response = render (request, 'home.html')
    response.set_cookie('name', 'Rahim')
    response.set_cookie('name', 'Muktadir', max_age=60*3) # 3 Min
    response.set_cookie('name', 'Muk74dir', expires=datetime.utcnow()+timedelta(days=7)) # 7 Days
    return response

def get_cookie(request):
    name_from_cookie = request.COOKIES.get('name', 'Guest') # Default Value Guest
    print(request.COOKIES)
    print(name_from_cookie)
    return render(request, 'get_cookie.html', {'name': name_from_cookie})

def delete_cookies(request):
    response = render(request, 'delete_cookie.html')
    response.delete_cookie('name')
    return response
```

8. Working with Session Framework :

Session vs Cookie

```python
def set_session(request):
    data = {
        'name' : 'Muktadir',
        'age' : 23,
        'language' : 'Bangla'
    }
    print(request.session.get_session_cookie_age())
    print(request.session.get_expiry_date())
    request.session.update(data)
    return render(request, 'home.html')

def get_session(request):
    name = request.session.get('name')
    age = request.session.get('age')
    return render(request, 'get_session.html', {'name' : name, 'age': age})

def delete_session(request):
    # del request.session['name']
    # del request.session['age'] # Delete Specific key-value from a session ID
    request.session.flush() # Delete Whole session ID
    return render(request, 'delete_session.html')
```

9. How to implement Session Expired Feature :

```python
def get_session(request):
    if 'name' in request.session:
        name = request.session.get('name')
        age = request.session.get('age')
        request.session.modified = True
        return render(request, 'get_session.html', {'name' : name, 'age': age})

    else:
        return HttpResponse( """
                <h1> Your Session has been expired. Kindly Log in Again. </h1>
            """ )
def delete_session(request):
    # del request.session['name']
    # del request.session['age'] # Delete Specific key-value from a session ID
    request.session.flush() # Delete Whole session ID
    return render(request, 'delete_session.html')
```