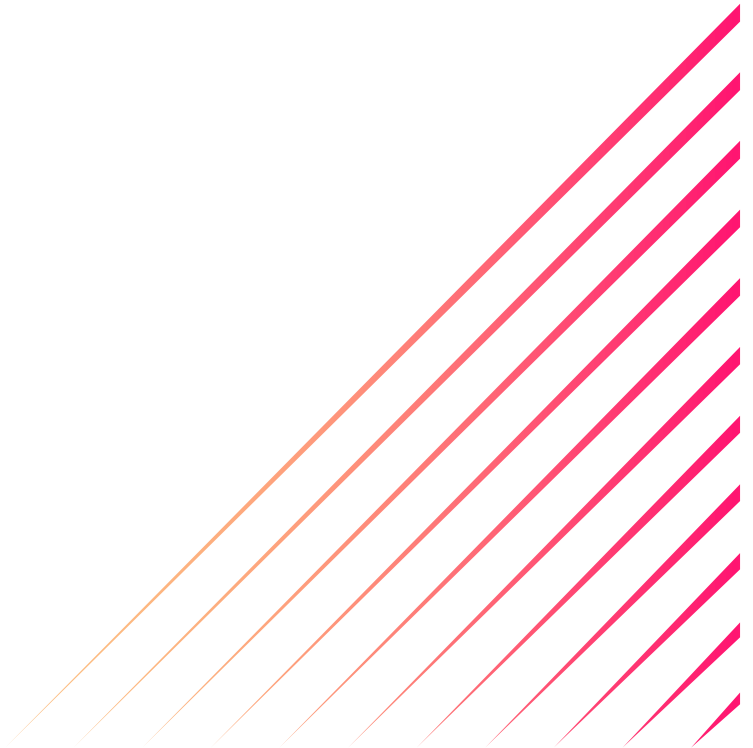




Django Task 3



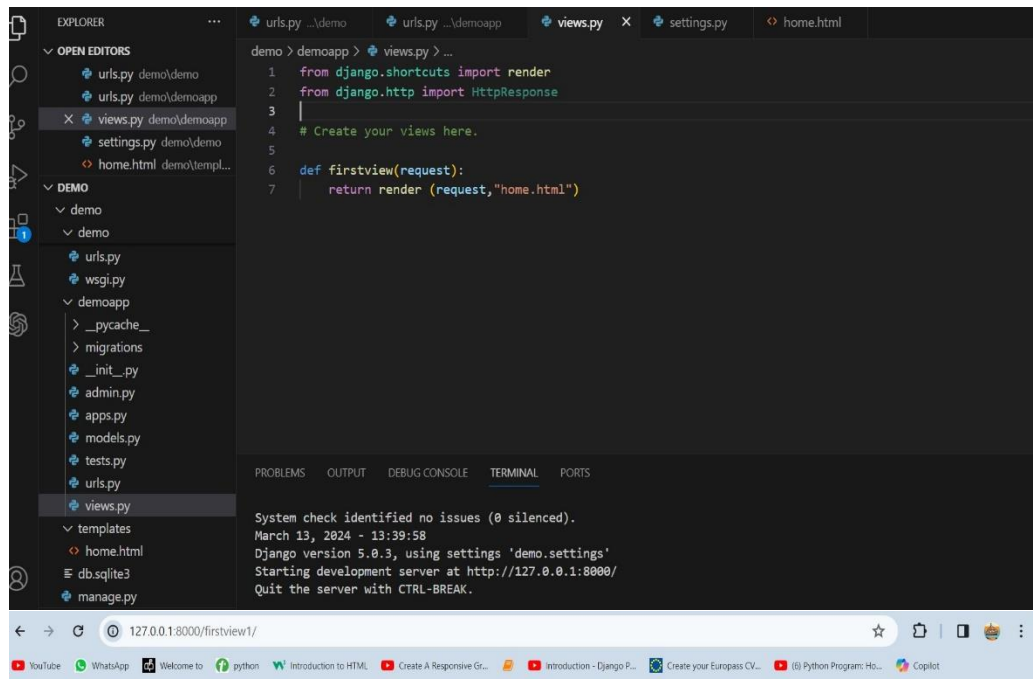
What are the key features of the Django Template language, and how do they contribute to its advantages in web development?

Django Template Language (DTL) is a potent tool facilitating dynamic web page creation within Django, a robust Python web framework. Crafted to promote efficient development while maintaining a clear separation of concerns between presentation and business logic layers, DTL boasts several key features pivotal to its advantages in web development:

1. **Straightforward Syntax:** DTL employs a simple, HTML-like syntax, fostering easy comprehension and maintenance of templates. This familiarity combined with Python logic expedites prototyping and development, lowering the learning curve for developers.
2. **Template Inheritance:** Supporting template inheritance, DTL empowers developers to create base templates with standardized layouts and structures. Child templates can then extend or override specific blocks, fostering code reusability and ensuring consistency across the web application's interface.
3. **Versatile Template Tags:** DTL offers an array of built-in template tags facilitating various actions within templates. These encompass control structures, template inclusion, data filtering, and more. Template tags provide a concise means to manipulate data and control content flow, enhancing template flexibility.
4. **Dynamic Filters:** DTL supports filters enabling developers to modify variables within templates dynamically. Filters can format dates, truncate text, capitalize strings, and perform other operations, enhancing template flexibility without cluttering views or models with presentation logic.
5. **Automatic Escaping:** DTL automatically escapes variables to thwart Cross-Site Scripting (XSS) attacks, mitigating security vulnerabilities. Developers retain manual control over variable safety, marking them as safe or unsafe based on context, thus managing escaping behaviour effectively.
6. **Internationalization and Localization Support:** DTL embeds built-in support for internationalization (i18n) and localization (l10n) of web applications. Developers can effortlessly mark translatable strings in templates, leveraging Django's translation framework to render content in the user's preferred language, enabling easier creation of multilingual websites.
7. **Custom Template Tags and Filters:** Django empowers developers to craft custom template tags and filters, extending DTL's functionality. This capability encapsulates complex logic into reusable components, facilitating template integration and promoting code organization, modularity, and maintainability.
8. **Seamless ORM Integration:** DTL seamlessly integrates with Django's Object-Relational Mapping (ORM) system, enabling direct access and display of database objects within templates. This tight integration streamlines development by reducing the necessity for additional code in views to pass data to templates.

Create a Django view using both function-based and class-based approaches. Utilize different Django Template Language (DTL) tags within these views. (attach screenshots of SITE

USING FUNCTION



```
demo > demoapp > views.py > ...
1 from django.shortcuts import render
2 from django.http import HttpResponse
3
4 # Create your views here.
5
6 def firstview(request):
7     return render (request,"home.html")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

System check identified no issues (0 silenced).
March 13, 2024 - 13:39:58
Django version 5.0.3, using settings 'demo.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

127.0.0.1:8000/firstview1/

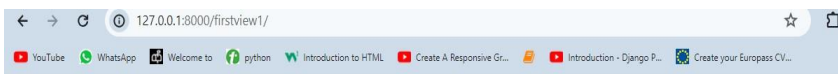
BLAIR'S NEW WEB PAGE

this is done using h2 tag

USING CLASS AND DTL TAGS

```
demo > templates > home copy.html
1  {% extends "home.html" %}
2
3  {% block content %}
4
5  <h1> title : {{name}} </h1>
6  <h2> year : {{year}} </h2>
7  💡
8  {% endblock %}
```

```
demo > demoapp > views.py > firstview1 > get
1  from django.shortcuts import render
2  from django.http import HttpResponse
3  from django.views import View
4  # Create your views here.
5
6  def firstview(request):
7      return render (request,"home.html")
8
9  class firstview1(View):
10     def get (self,request,*args,**kwargs):
11         company={
12             "name" : "microsoft",
13             "year" : "2022"
14         }
15         return render(request,"home copy.html",company)
```



BLAIR'S NEW WEB PAGE

title : microsoft

year : 2022