----- What you'll Learn Making this App -----

1. How to use HTML Templates
2. How to Merge HTML Content using Multiple Files
3. Learn the URL, and Block Tags
4. How to Make a Basic Menu System
5. How to Perform Tests to Verify Your Site is Working

----- Install Virtual Environment & Create Project

```
mkdir tut2
cd tut2
pipenv install django==3.0
pipenv shell
django-admin startproject tut2 .
python manage.py startapp pages
```

----- Setup Settings & Define Template Location -----

Add app to settings

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'pages.apps.PagesConfig', <---- ADDED HERE
]
```

You can create a project level template directory
but you have to tell the settings.py file where
the template directory is in DIRS
Turn off the server : Ctrl-C

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Update the timezone
https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

```
TIME_ZONE = 'America/New_York'

# ----- Create Home HTML File -----

# Put an html file in the templates directory named home.html

Add <h1>Home</h1> to it


# ----- Update Views File -----

# Use the built-in TemplateView in pages/views.py
from django.views.generic import TemplateView

class HomePageView(TemplateView):
  template_name = 'home.html'

# ----- Update URLs File -----

# Update our project urls file
from django.contrib import admin
from django.urls import path, include # new

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('pages.urls')), # new
]

# ----- Setup Pages URLs File -----

# Create an app level urls.py file
from django.urls import path
from .views import HomePageView

# Since we are using a class view we add
# as_view()
urlpatterns = [
path('', HomePageView.as_view(), name='home')
]

# Run the server to see results
python manage.py runserver

# ----- Create an About Page -----

# Create about.html in templates folder

# Add <h1>About</h1> to it

# Add a view for the page in views.py
from django.views.generic import TemplateView

class HomePageView(TemplateView):
```

```
  template_name = 'home.html'

class AboutPageView(TemplateView):
  template_name = 'about.html'

# Connect the URL to our template in pages/urls.py
from django.urls import path
from .views import HomePageView, AboutPageView

# Since we are using a class view we add
# as_view()
urlpatterns = [
path('about', AboutPageView.as_view(), name='about'),
path('', HomePageView.as_view(), name='home'),
]

# ----- Reusing Content on Multiple Pages -----

# We can define html in one page and then use it in others

# 1. Create an HTML file that contains header info
# in templates/base.html

# Template tags look like this {% stuff %}
# I use a link template tag below. This tag creates
# links to pages by referring to the pages name which
# was defined in the file pages/urls.py

# This is a simple menu system

# The block tag can display content that is passed in
# the page template files home.html and about.html

<header>
  <a href="{% url 'home' %}">Home</a> | <a href="{% url 'about' %}">About</a>
</header>

{% block content %}
{% endblock content %}

# 2. Update the template files using base.html
# Extend pulls in the code from base.HTML

{% extends 'base.html' %}
{% block content %}
<h1>Homepage</h1>
{% endblock content %}

{% extends 'base.html' %}
{% block content %}
<h1>About Us</h1>
{% endblock content %}

# ----- TESTING -----
```

```
# Automated tests can be used to make sure our site
# is working as intended. We'll use tests.py to verify
# our response status code is 200 which means it succeeded

# As the site gets larger we'll create a test package
# that will split up tests into submodules

# tests.py
# SimpleTestCase is used when you aren't testing
# database queries. It can check for exceptions,
# forms, HTML responses, redirects, and if we are
# receiving expected results through HTML, XML,
# JSON and more.
from django.test import SimpleTestCase

# Testing if we receive successful responses from
# parts of the site
class LinkTests(SimpleTestCase):
    def test_home_status_code(self):
        response = self.client.get('/')
        self.assertEqual(response.status_code, 200)

    def test_about_status_code(self):
        response = self.client.get('/about')
        self.assertEqual(response.status_code, 200)
```