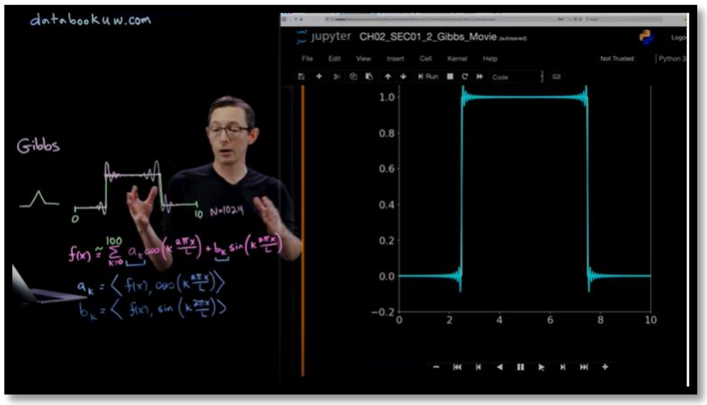
DAILY ASSESSMENT FORMAT

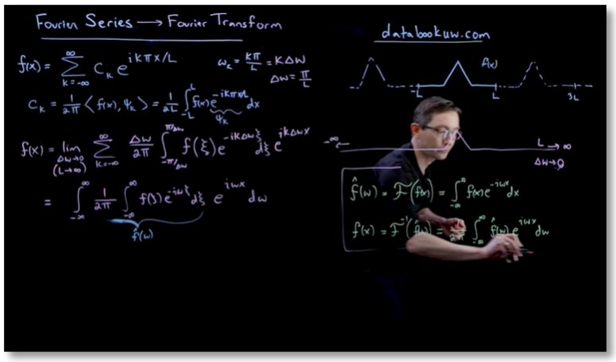
Date:	**26/05/2020**	**Name:**	**Davis S. Patel**
Course:	**DSP**	**USN:**	**4AL16EC045**
Topic:	➢ **Fourier Series &amp; Gibbs Phenomena using Python** ➢ **Fourier Transform** ➢ **Fourier Transform Derivatives** ➢ **Fourier Transform & Convolution** ➢ **Intuition of Fourier** ➢ **Transform and Laplace Transform** ➢ **Laplace Transform of First order** ➢ **Implementation of Laplace Transform using Matlab** ➢ **Applications of Z-Transform** ➢ **Find the Z-Transform of sequence using Matlab**	**Semester & Section:**	**8th - A**
GitHub Repository:	**Davis**		

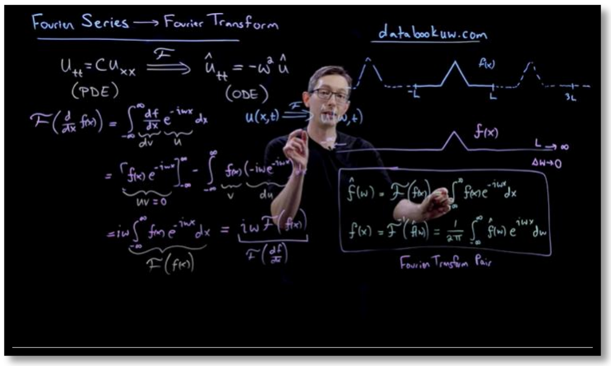
FORENOON SESSION DETAILS
Image of session 

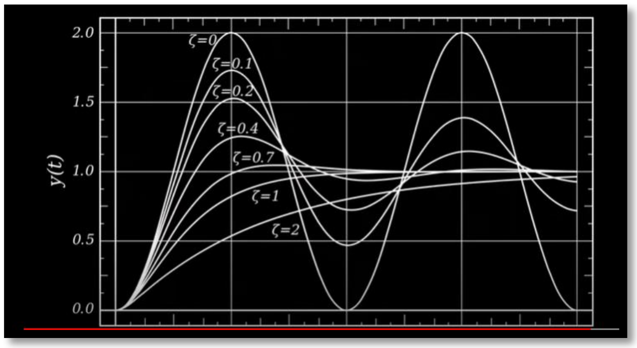
$f(x) \approx \sum_{k=0}^{100} a_k \cos\left(k\frac{2\pi x}{L}\right) + b_k \sin\left(k\frac{2\pi x}{L}\right)$

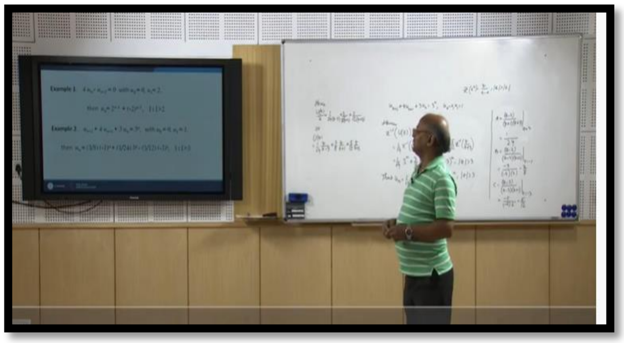
$a_k = \left\langle f(x), \cos\left(k\frac{2\pi x}{L}\right)\right\rangle$

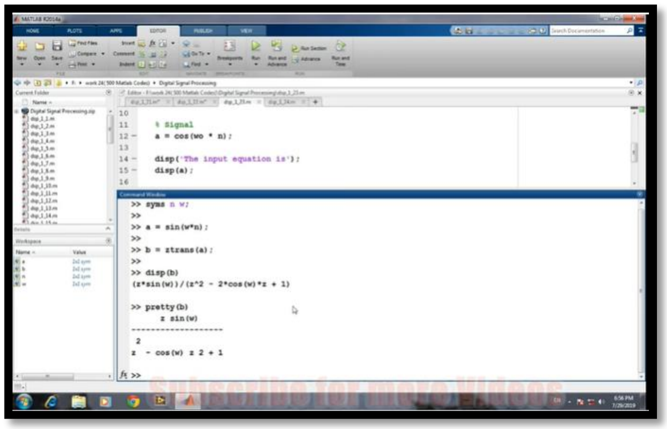
$b_k = \left\langle f(x), \sin\left(k\frac{2\pi x}{L}\right)\right\rangle$


Fourier Series ⟶ Fourier Transform
databookuw.com
$f(x)=\sum_{k=-\infty}^{\infty} c_k e^{ik\pi x/L}$
$\omega_k=\frac{k\pi}{L}=k\Delta\omega$
$\Delta\omega=\frac{\pi}{L}$
$\hat{f}(\omega)=\mathcal{F}(f(x))=\int_{-\infty}^{\infty} f(x)e^{-i\omega x}dx$
$f(x)=\mathcal{F}^{-1}(\hat{f}(\omega))=\frac{1}{2\pi}\int_{-\infty}^{\infty}\hat{f}(\omega)e^{i\omega x}d\omega$


Fourier Series ⟶ Fourier Transform
databookuw.com
$u_{tt}=c\,u_{xx}$ (PDE)
$\hat{u}_{tt}=-\omega^2\hat{u}$ (ODE)
$\hat{f}(\omega)=\mathcal{F}(f(x))=\int_{-\infty}^{\infty} f(x)e^{-i\omega x}dx$
$f(x)=\mathcal{F}^{-1}(\hat{f}(\omega))=\frac{1}{2\pi}\int_{-\infty}^{\infty}\hat{f}(\omega)e^{i\omega x}d\omega$
Fourier Transform Pair







```
% Signal
a = cos(wo * n);

disp('The input equation is');
disp(a);
```

```
>> syms n w;
>>
>> a = sin(w*n);
>>
>> b = ztrans(a);
>>
>> disp(b)
(z*sin(w))/(z^2 - 2*cos(w)*z + 1)

>> pretty(b)
      z sin(w)
-------------------
 2
z  - cos(w) z 2 + 1

fx >>
```

Report –

A Fourier series representation of a periodic square wave. If the periodic square wave is written as an odd function, then the Fourier series is

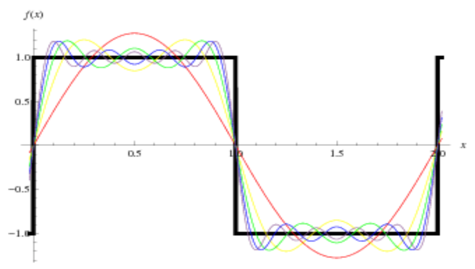
$$g(t) = \frac{1}{2} + \frac{2}{\pi}\sin t + \frac{2}{3\pi}\sin 3t + \frac{2}{5\pi}\sin 5t + \cdots$$

Since odd numbers can be written as 2n−1, where n is an integer, we can re-write this series as

$$g(t) = \frac{1}{2} + \sum_{n=1}^{\infty} \frac{2}{(2n-1)\pi}\sin(2n-1)t\ .$$

There are two important properties of periodic square waves for bench top researchers. The first is Gibb's phenomenon. Sharp transitions, such as the edge of the square wave, generate high frequency components in the Fourier series. Gibbs's phenomena produces artifacts in jpg images and also produces aliasing that makes the determination of power spectral descriptions of neural spike trains and the EKG of the heart beat difficult. The second important property concerns the relationship between square and sine waves. At first sight this does not seem to be that important. However, in electronics, communication signals are often transmitted as square wave pulses and on arrival to their destination are converted to sine waves. Moreover in an electronic circuit it is often easier to create a train of periodic square waves rather than to generate a sine wave.

The Gibbs phenomenon is an overshoot (or "ringing") of Fourier series and other Eigen function series occurring at simple discontinuities. It can be reduced with the Lanczos *sigma* factor. The phenomenon is illustrated below in the Fourier series of a square wave.



It is often necessary to evaluate a sum in a computer program, such as occurs in (??). In Python sums can be evaluated using a while loop. For example to determine y where

$$y = \sum_{i=1}^{10} i^2$$

we can write

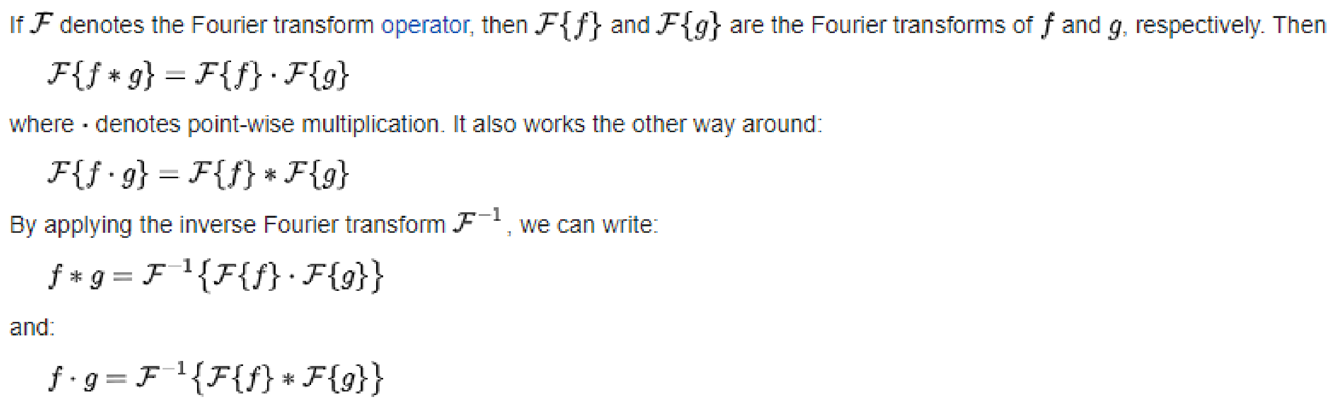
```
sum=0
n=1
while n <11:
    sum=sum+n^2
    n=n+1
print sum
```

It is possible to use a `for loop` in Python; however, a `while loop` construct is typically much faster.

<u>Relationship between convolution and Fourier transforms</u>

- It turns out that convolving two functions is equivalent to multiplying them in the frequency domain – One multiplies the complex numbers representing coefficients at each frequency
- In other words, we can perform a convolution by taking the Fourier transform of both functions, multiplying the results, and then performing an inverse Fourier transform

The convolution theorem states that under suitable conditions the Fourier transform of a convolution of two signals is the pointwise product of their Fourier transforms. In other words, convolution in one domain (e.g., time domain) equals point-wise multiplication in the other domain (e.g., frequency domain). Versions of the convolution theorem are true for various Fourier-related transforms.



If $\mathcal{F}$ denotes the Fourier transform operator, then $\mathcal{F}\{f\}$ and $\mathcal{F}\{g\}$ are the Fourier transforms of f and g, respectively. Then

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$

where $\cdot$ denotes point-wise multiplication. It also works the other way around:

$$\mathcal{F}\{f \cdot g\} = \mathcal{F}\{f\} * \mathcal{F}\{g\}$$

By applying the inverse Fourier transform $\mathcal{F}^{-1}$, we can write:

$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

and:

$$f \cdot g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} * \mathcal{F}\{g\}\}$$

Z transform is used in many applications of mathematics and signal processing. The lists of applications of z transform are:-

-Uses to analysis of digital filters.

-Used to simulate the continuous systems.

-Analyze the linear discrete system.

-Used to finding frequency response.

-Analysis of discrete signal.

-Helps in system design and analysis and also checks the systems stability.

-For automatic controls in telecommunication.

-Enhance the electrical and mechanical energy to provide dynamic nature of the system.

MATLAB program for Z-Transform of finite duration sequence

```
clc;
close all;
clear all;
syms 'z';
disp('If you input a finite duration sequence x(n), we will give you its z-transform');
nf=input('Please input the initial value of n = ');
nl=input('Please input the final value of n = ');
x= input('Please input the sequence x(n)= ');
syms 'm';
syms 'y';
```

```
f(y,m)=(y*(z^(-m)));
disp('Z-transform of the input sequence is displayed below');
k=1;
for n=nf:1:nl
    answer(k)=(f((x(k)),n));
   k=k+1;
end
disp(sum(answer));
```

Output

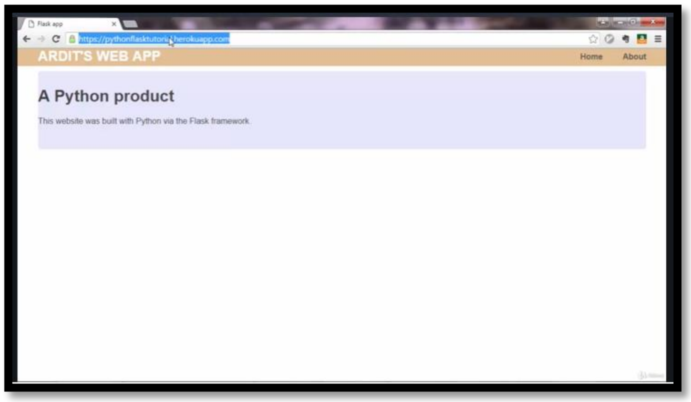
```
Please input the initial value of n = 0
Please input the final value of n = 4
Please input the sequence x(n)= [1 0 3 -1 2]
Z-transform of the input sequence is displayed below

3/z^2 - 1/z^3 + 2/z^4 + 1
```

DAILY ASSESSMENT FORMAT

Date:	**26/05/2020**	**Name:**	**Davis S. Patel**
Course:	**Python Course**	**USN:**	**4AL16EC045**
Topic:	**Application 4: Build a Personal Website with Python and Flask**	**Semester & Section:**	**8th - A**
GitHub Repository:	**Davis**		

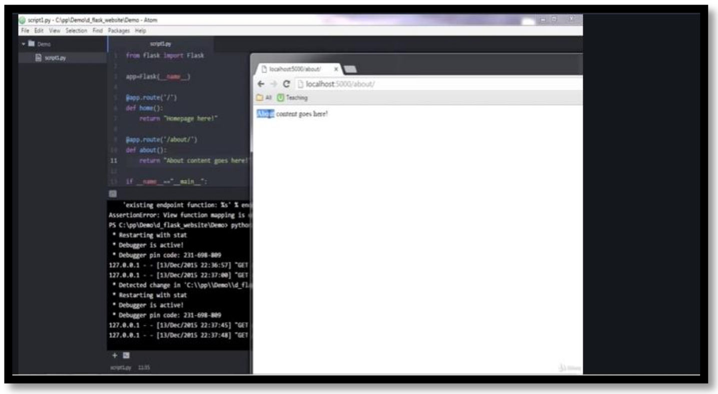
AFTERNOON SESSION DETAILS

Image of Session



A Python product

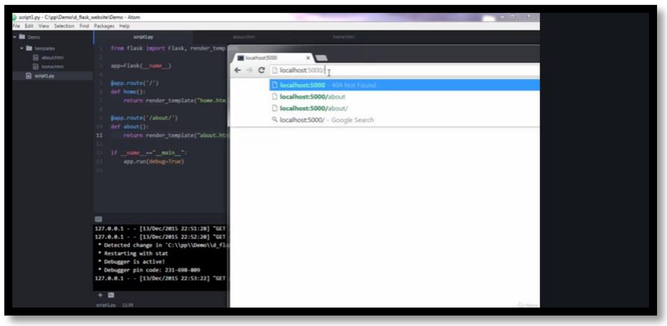
This website was built with Python via the Flask framework.


script1.py - C:\pp\Demo\d_flask_website\Demo - Atom
File Edit View Selection Find Packages Help
Demo
script1.py
localhost:5000/about/
All
Teaching

```
from flask import Flask

app=Flask(__name__)

@app.route('/')
def home():
    return "Homepage here!"

@app.route('/about/')
def about():
    return "About content goes here!"

if __name__=="__main__":
```

```
    'existing endpoint function: %s' % en
AssertionError: View function mapping is 
PS C:\pp\Demo\d_flask_website\Demo> pytho
 * Restarting with stat
 * Debugger is active!
 * Debugger pin code: 231-698-809
127.0.0.1 - - [13/Dec/2015 22:36:57] "GET 
127.0.0.1 - - [13/Dec/2015 22:37:00] "GET 
 * Detected change in 'C:\\pp\\Demo\\d_fl
 * Restarting with stat
 * Debugger is active!
 * Debugger pin code: 231-698-809
127.0.0.1 - - [13/Dec/2015 22:37:45] "GET 
127.0.0.1 - - [13/Dec/2015 22:37:48] "GET 
```

About content goes here!

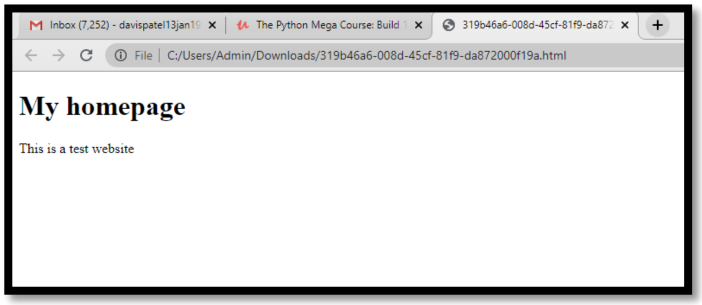


```python
from flask import Flask, render_temp

app=Flask(__name__)

@app.route('/')
def home():
    return render_template("home.htm

@app.route('/about/')
def about():
    return render_template("about.htm

if __name__=="__main__":
    app.run(debug=True)
```

```
127.0.0.1 - - [13/Dec/2015 22:51:28] "GET
127.0.0.1 - - [13/Dec/2015 22:52:20] "GET
 * Detected change in 'C:\\pp\\Demo\\d_fl
 * Restarting with stat
 * Debugger is active!
 * Debugger pin code: 231-698-809
127.0.0.1 - - [13/Dec/2015 22:53:22] "GET
```

localhost:5000/

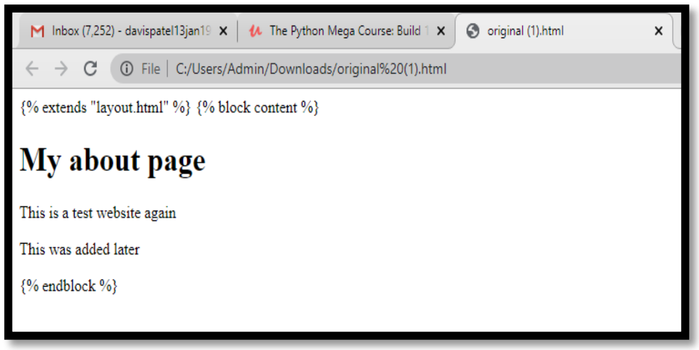
localhost:5000 - 404 Not Found
localhost:5000/about
localhost:5000/about/
localhost:5000/ - Google Search



C:/Users/Admin/Downloads/319b46a6-008d-45cf-81f9-da872000f19a.html

My homepage

This is a test website



C:/Users/Admin/Downloads/original%20(1).html

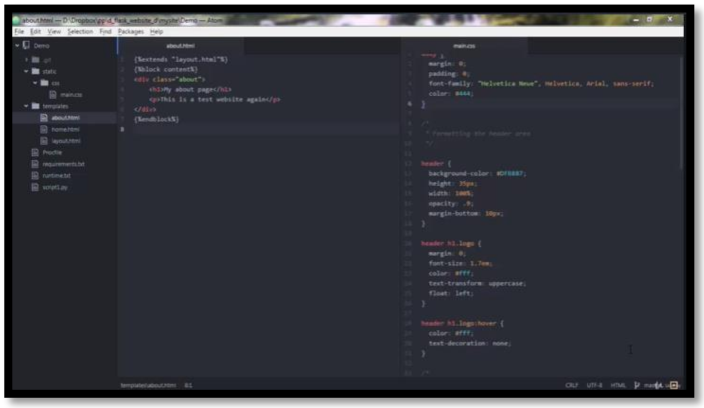
{% extends "layout.html" %} {% block content %}

My about page

This is a test website again

This was added later

{% endblock %}



```html
{%extends "layout.html"%}
{%block content%}
<div class="about">
    <h1>My about page</h1>
    <p>This is a test website again</p>
</div>
{%endblock%}
```

```css
  margin: 0;
  padding: 0;
  font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;
  color: #444;
}

/*
 * Formatting the header area
 */

header {
  background-color: #DF8887;
  height: 35px;
  width: 100%;
  opacity: .9;
  margin-bottom: 10px;
}

header h1.logo {
  margin: 0;
  font-size: 1.7em;
  color: #fff;
  text-transform: uppercase;
  float: left;
}

header h1.logo:hover {
  color: #fff;
  text-decoration: none;
}
```

Report –

Application 4: Build a Personal Website with Python and Flask

Flask makes the process of designing a web application simpler.
This code lets us run a basic web application that we can serve, as if it were a website.

```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def home():
   return "Hello, World!"
if __name__ == "__main__":
   app.run(debug=True)
```

This code is stored in our main.py. Here we are importing the Flask module and creating a Flask web server from the Flask module. **__name__ means this current file**. In this case, it will be main.py. This current file will represent web application.
We are creating an instance of the Flask class and calling it app. Here we are creating a new web application. It represents the default page.

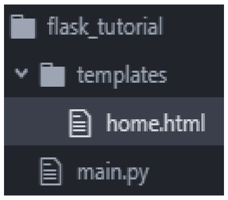
When the user goes to my website and they go to the default page (nothing after the slash), then the function below will get activated. When you run your Python script, Python assigns the name "__main__" to the script when executed.

If we import another script, the if statement will prevent other scripts from running. When we run main.py, it will change its name to __main__ and only then will that if statement activate.
This will run the application. Having debug=True allows possible Python errors to appear on the web page. This will help us trace the errors.

HTML, CSS, and Virtual Environments

HTML and Templates in Flask

First create a new HTML file. Name it as **home.html.** The Flask Framework looks for HTML files in a folder called **templates.** You **need to create a templates** folder and put all your HTML files in there.



Always keep the main.py outside of your templates folder. we need to change our main.py so that we can view the HTML file we created.

```
from flask import Flask, render_template
app = Flask(__name__)
@app.route("/")
def home():
    return render_template("home.html")
@app.route("/salvador")
def salvador():
    return "Hello, Salvador"
if __name__ == "__main__":
    app.run(debug=True)
```

We imported render_template () method from the flask framework. render_template () looks for a template (HTML file) in the templates folder. Then it will render the template for which you ask. Learn more about render_template () function.
We change the return so that now it returns render_template ("home.html"). This will let us view our HTML file.

We can use Flask to make the process of creating a navigation menu easier. First, let's create a template.html. This template.html will serve as a parent template. The two child templates will inherit code from it.

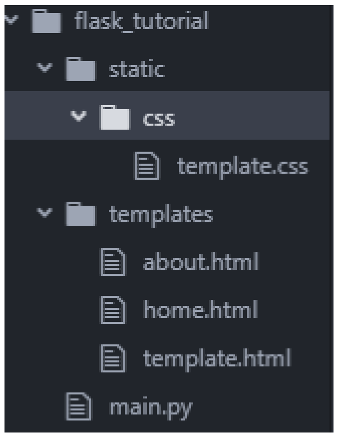
```
<html lang="en" dir="ltr">
 <head>
   <meta charset="utf-8">
   <title>Flask Parent Template</title>
   <link rel="stylesheet" href="{{ url_for('static',     filename='css/template.css') }}">
 </head>
 <body>
    <header>
      <div class="container">
        <h1 class="logo">First Web App</h1>
        <strong><nav>
          <ul class="menu">
            <li><a href="{{ url_for('home') }}">Home</a></li>
            <li><a href="{{ url_for('about') }}">About</a></li>
          </ul>
        </nav></strong>
      </div>
    </header>

    {% block content %}
    {% endblock %}

 </body>
</html>
```

We use the function called url_for ()**.** It accepts the name of the function as an argument. Right now we gave it the name of the function. More information on **url_for () function.** The two lines with the curly brackets will be **replaced by the content of home.html and about.html.** This will depend on the URL in which the user is browsing.

These changes allow the child pages (home.html and about.html) to connect to the parent (template.html). This allows us to not have to **copy the code for the navigation menu in the about.html and home.html.**

In the same way as we created a folder called **templates** to store all our HTML templates, we need a folder called **static**.

In **static**, we will store our CSS, JavaScript, images, and other necessary files. That is why it is important that you should create a **CSS folder to store your stylesheets.** After you do this, your project folder should look like this:



Our template.html is the one that links all pages. We can insert the code here and it will be applicable to all child pages. We may use Python for others projects besides web-development. Our projects might have different versions of Python installed, different dependencies and packages.

We use virtualenv to create an isolated environment for your Python project. This means that each project can have its own dependencies regardless of what dependencies every other project has.

Steps to deploy a static Flask website to Heroku

1. Create an account on www.heroku.com if you don't have one already.

2. Download and install Heroku Toolbelt from https://devcenter.heroku.com/articles/heroku-cli

3. Install gunicorn with "pip install gunicorn". Make sure you're using pip from your virtual environment if you have one.

4. Create a requirement.txt file in the main app directory where the main Python app file is located. You can create that file by running "pip freeze > requirements.txt" in the command line. Make sure you're using pip from your virtual environment if you have one. The requirement.txt file should now contain a list of Python packages.

5. Create a file named "Procfile" in the main app directory. The file should not contain any extension. Then type in this line inside: "web: gunicorn script1:app" where "script1" should be replaced with the name of your Python script and "app" with the name of the variable holding your Flask app.

6. Create a runtime.txt file in the main app directory and type "python-3.5.1" inside.

If you're using Python 2, you may want to type in "python-2.7.11" instead.

7. Open your computer terminal/command line to point to the directory where the Python file containing your app code is located.

8. Using the terminal, log in to Heroku with "heroku login"

9. Enter your Heroku email address

10. Enter your Heroku password

11. Create a new Heroku app with "heroku create myawesomeappname"

17. Initialize a local git repository with "git init"

18. Add your local application files to git with "git add ."

19. Tell git your email address with "git config --global user.email "myemail@hotmail.com"". Make sure the email address is inside quotes here.

20. Tell git your username (just pick whatever username) with "git config --global user.name "whateverusername"". The username should be in quotes.

21. Commit the changes with "git commit -m "first commit"". Make sure "first commit" is inside quotes.

22. Before pushing the changes to Heroku, tell heroku the name of the app you want to use with "heroku git:remote --app myawesomeappname"

23. Push the changes to Heroku with "git push heroku master"

26. That should do it. Go ahead and open your app with "heroku open".

Troubleshooting

If you deployed your website on Heroku but when you visit the website on the browser you see an error, you probably did something wrong during the deployment.

You can see what you did wrong by looking at the server logs. You can access the server logs by running the following in your terminal: `heroku logs`

This command will show a series of messages. Carefully read the logs to understand what went wrong.