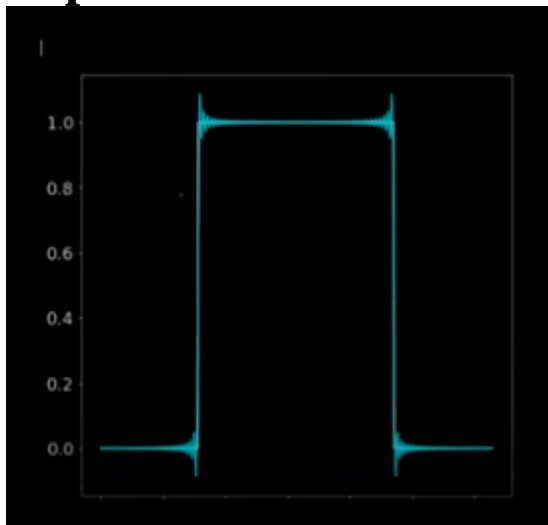


## DAILY ASSESSMENT

<b>Date:</b>	26/05/2020	<b>Name:</b>	Chandana.R
<b>Course:</b>	DSP	<b>USN:</b>	4AL16EC017
<b>Topic:</b>	Fourier series , transformer, laplus transform, basic programming in matlab, Z-tran, application of Z-tran	<b>Semester &amp; Section:</b>	8(A)
<b>Github Repository:</b>	Chandana-shaiva		

### FORENOON SESSION DETAILS

#### Report:-



#### ➤ Fourier Series and Gibbs Phenomena [Python]:

- The Fourier theorem to construct 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 the Gibbs phenomenon that arises when we increase the number of 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.
- An example in biology arose in the measurement of the Bode plot for the hyperosmolar response of yeast cells we see that a square wave can be thought of as a sine wave to which terms with higher frequencies have been added.
- Thus an attractive idea to generate a sine wave is to low-pass filter the train of periodic square waves to remove the higher frequency terms, leaving the pure sine wave component behind.

## Gibbs phenomenon:

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



### ➤ The Fourier Transform:

- The Fourier transform of a function of time is a complex-valued function of frequency,
- whose magnitude (absolute value) represents the amount of that frequency present in the original function, and whose argument is the phase offset of the basic sinusoid in that frequency.
- The Fourier transform is not limited to functions of time, but the domain of the original function is commonly referred to as the *time domain*.
- There is also an *inverse Fourier transform* that mathematically synthesizes the original function from its frequency domain representation, as proven by the Fourier inversion theorem.

Here we assume  $f(x)$ ,  $g(x)$  and  $h(x)$  are *integrals functions*: Lebesgue-measurable on the real line satisfying:

$$\int_{-\infty}^{\infty} |f(x)| dx < \infty.$$

We denote the Fourier transforms of these functions as  $\hat{f}(\xi)$ ,  $\hat{g}(\xi)$  and  $\hat{h}(\xi)$  respectively.

### ➤ The Fourier Transform and Derivatives

The Fourier transform is a generalization of the complex Fourier series in the limit as  $L = \infty$

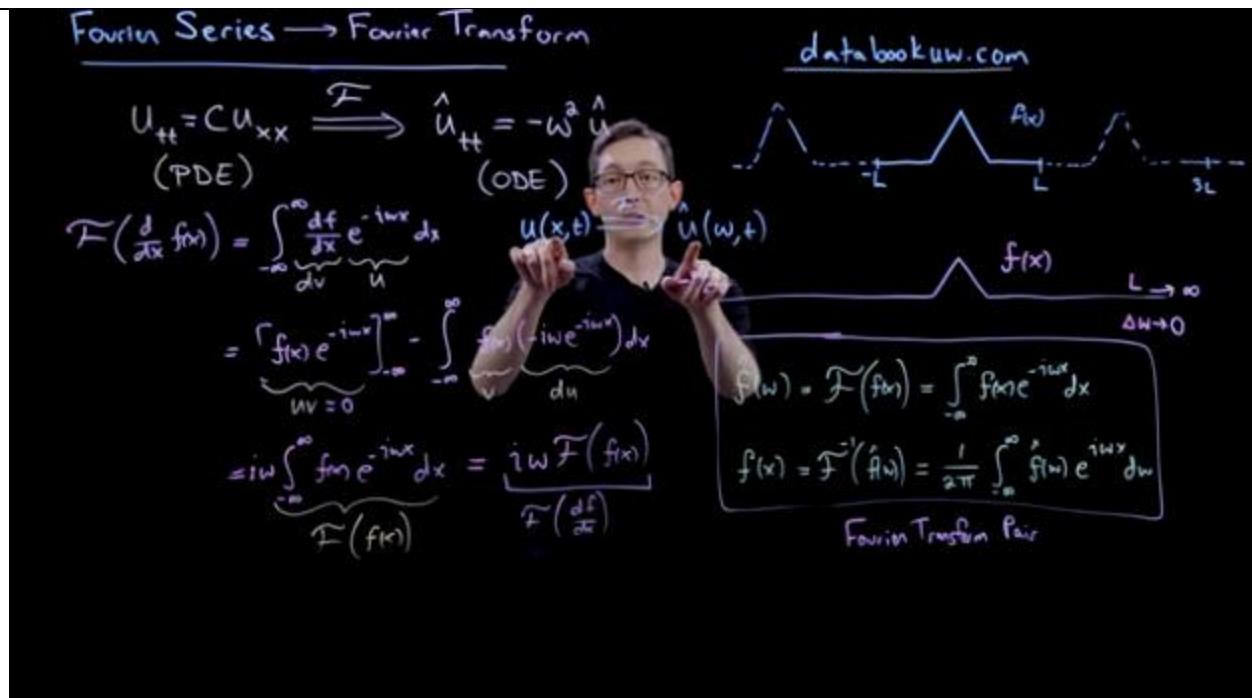
$$f(x) = \sum_{n=-\infty}^{\infty} A_n e^{i(2\pi nx/L)}$$

$$A_n = \frac{1}{L} \int_{-L/2}^{L/2} f(x) e^{-i(2\pi nx/L)} dx.$$

$$f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi i k x} dk$$

$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx.$$

$$F(k) = \mathcal{F}[f(x)] = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$



## The Fourier Transform and Convolution Integrals

$$w(t) = u(t)v(t) \Leftrightarrow W(f) = U(f) * V(f)$$

Convolution Theorem:

$$w(t) = u(t)v(t) \Leftrightarrow W(f) = U(f) * V(f)$$

$$w(t) = u(t) * v(t) \Leftrightarrow W(f) = U(f)V(f)$$

Convolution in the time domain is equivalent to multiplication in the frequency domain and vice versa

Proof of second line:

Given  $u(t)$ ,  $v(t)$  and  $w(t)$

$$\text{satisfying } w(t) = u(t)v(t) \Leftrightarrow W(f) = U(f) * V(f)$$

define dual waveforms  $x(t)$ ,  $y(t)$  and  $z(t)$  as follows:

$$x(t) = U(t) \Leftrightarrow X(f) = u(-f)$$

$$y(t) = V(t) \Leftrightarrow Y(f) = v(-f)$$

$$z(t) = W(t) \Leftrightarrow Z(f) = w(-f)$$

Now the convolution property becomes:

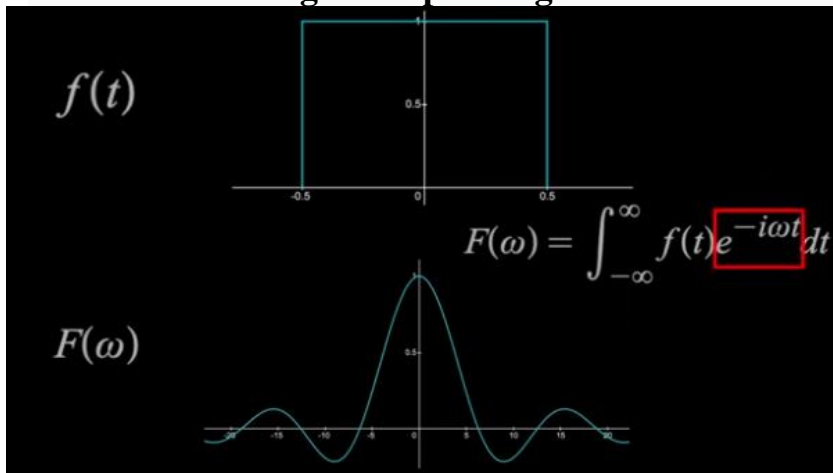
$$w(-f) = u(-f)v(-f) \Leftrightarrow W(f) = U(f) * V(f)$$

$$[\text{sub } t \leftrightarrow \pm f]$$

$$Z(f) = X(f)Y(f) \Leftrightarrow z(t) = x(t) * y(t)$$

The intuition behind Fourier and Laplace transforms

there are some fundamental aspects of the Fourier and Laplace transforms which can be quickly communicated, however, the "intuition" will depend on the background of who is doing the explaining.



$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t} dt$$

$$e^{-i\omega t} = \cos(\omega t) - i \sin(\omega t)$$

$$F(\omega) = \int_{-\infty}^{\infty} f(t)(\cos \omega t - i \sin \omega t) dt$$

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cos(\omega t) dt - i \int_{-\infty}^{\infty} f(t) \sin(\omega t) dt$$

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t} dt$$

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cos(\omega t) dt - i \int_{-\infty}^{\infty} f(t) \sin(\omega t) dt$$

magnitude

$\int_{-\infty}^{\infty} f(t) \cos(\omega t) dt$

$\int_{-\infty}^{\infty} f(t) \sin(\omega t) dt$

Courant and Hilbert go on to take a function  $f(x)$  represented by a Fourier series in the interval  $-1 < x < 1$  as follows:

$$f(x) = \sum_{k=-\infty}^{\infty} a_k e^{ik \frac{\pi}{l} x}$$

$$a_k = \frac{1}{2l} \int_{-l}^l f(t) e^{-ik \frac{\pi}{l} x} dt$$

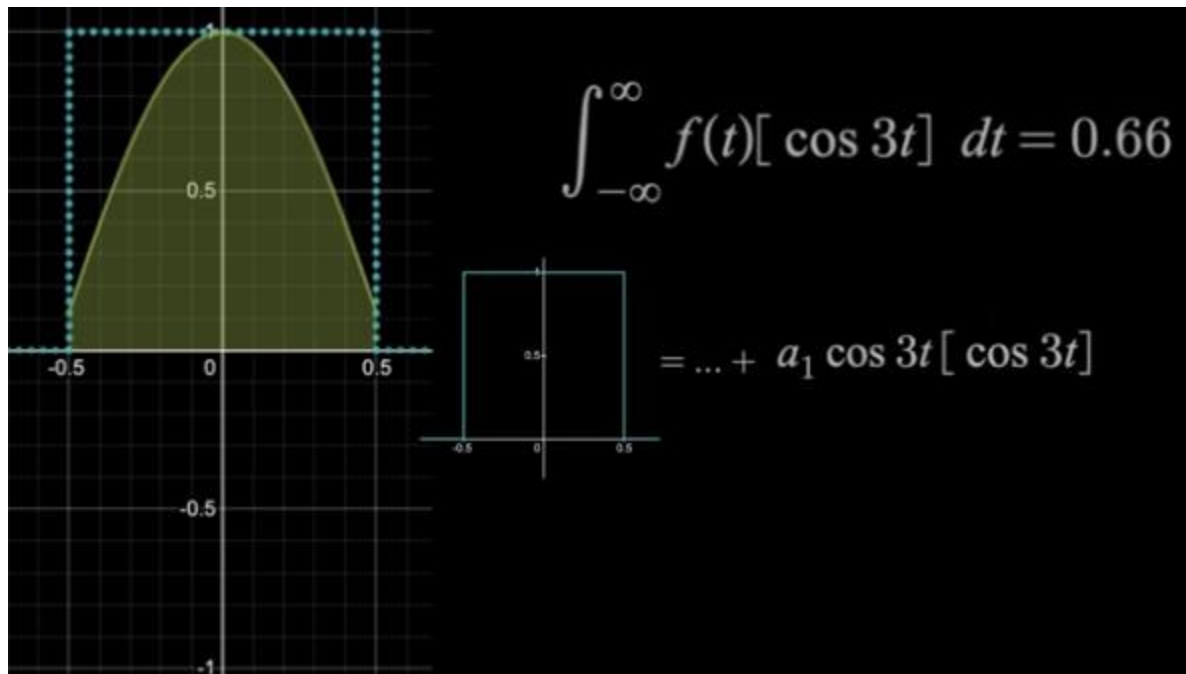
and they say "it seems desirable to let  $l$  go to  $\infty$ , since then it is no longer necessary to require that  $f$  be continued periodically; thus one may hope to obtain a representation for a non-periodic function defined for all real  $x$ . On this basis it is proved in every Fourier theory course that  $f(x)$  can be represented as follows:

$$f(x) = \frac{1}{\pi} \int_{-\infty}^{\infty} f(t) \cos u(t-x) dt$$

So here is the high level intuition which is actually rigorous (subject to various caveats):

$f \rightarrow$  Weierstrass approximation

$$f \rightarrow \text{Weierstrass approximation} = \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k \sin kx + b_k \cos kx) \rightarrow \text{Fourier integral}$$



## Laplace Transform: First Order Equation

The time domain equation that describes the behaviour of a first order system is given as follows:

$$\tau \frac{d\theta_o}{dt} + \theta_o = K\theta_i$$

Many different systems can be modeled in this manner. For example, filling a tank with water, heating a tank, charging a capacitor, measuring temperature with a thermometer are all examples of first order systems that can be modeled to give a time constant and a static gain

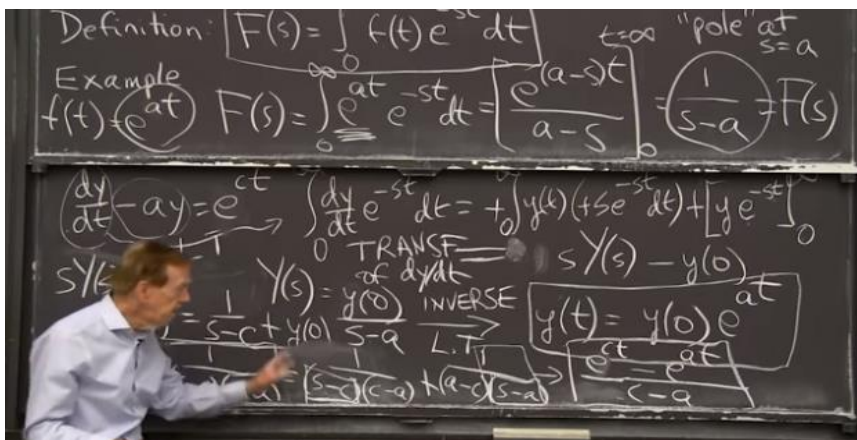
Applying the Laplace transform to this equation gives:

$$\tau s \theta_o(s) + \theta_o(s) = K\theta_i(s)$$

$$\tau s \theta_o(s) + \theta_o(s) = K\theta_i(s)$$

$$\Rightarrow \theta_o(s)[1 + \tau s] = K\theta_i(s)$$

$$\Rightarrow \frac{\theta_o(s)}{\theta_i(s)} = \frac{K}{1 + \tau s}$$



## ➤ Basics of Programming using MATLAB:

MATLAB is a programming language developed by Math Works. It started out as a matrix programming language where linear algebra programming was simple. It can be run both under interactive sessions and as a batch job. This tutorial gives you aggressively a gentle introduction of MATLAB programming language. It is designed to give students fluency in MATLAB programming language. Problem-based MATLAB examples have been given in simple and easy way to make your learning fast and effective.

Deep learning is getting a lot of attention these days, and for good reason. It's achieving unprecedented levels of accuracy—to the point where deep learning algorithms can outperform humans at classifying images and can beat the world's best GO player.

If you are interested in using deep learning technology for your project, but you've never used it before, where do you begin?

Should you spend time using deep learning models or can you use machine learning techniques to achieve the same results? Is it better to build a new neural network or use an existing pretrained network for image classification

### MATLAB Code: Key Parts

```
% Define Parameters and Initial Conditions
param.g = 9.81;
u0 = 35*cos(pi/

[tOut, XOut] = ode45(@bal

plot(XOut,
```

Comment

Assignment

(Math) Expression

Calling a function





## How to calculate Z-Transform in Matlab:

`ztrans(f)` finds the Z-Transform of `f`. By default, the independent variable is `n` and the transformation variable is `z`. If `f` does not contain `n`, `ztrans` uses `symvar`.

example

`ztrans(f,transVar)` uses the transformation variable `transVar` instead of `z`.

example

`ztrans(f,var,transVar)` uses the independent variable `var` and transformation variable `transVar` instead of `n` and `z`, respectively.

Compute the Z-transform of `sin(n)`. By default, the transform is in terms of `z`.

```
syms n
f = sin(n);
ztrans(f)
ans =
(z*sin(1))/(z^2 - 2*cos(1)*z + 1)
```

Compute the Z-transform of `exp(m+n)`. By default, the independent variable is `n` and the transformation variable is `z`.

```
syms m n
f = exp(m+n);
ztrans(f)
ans =
(z*exp(m))/(z - exp(1))
```

Specify the transformation variable as `y`. If you specify only one variable, that variable is the transformation variable. The independent variable is still `n`.

```
syms y
ztrans(f,y)
ans =
(y*exp(m))/(y - exp(1))
```

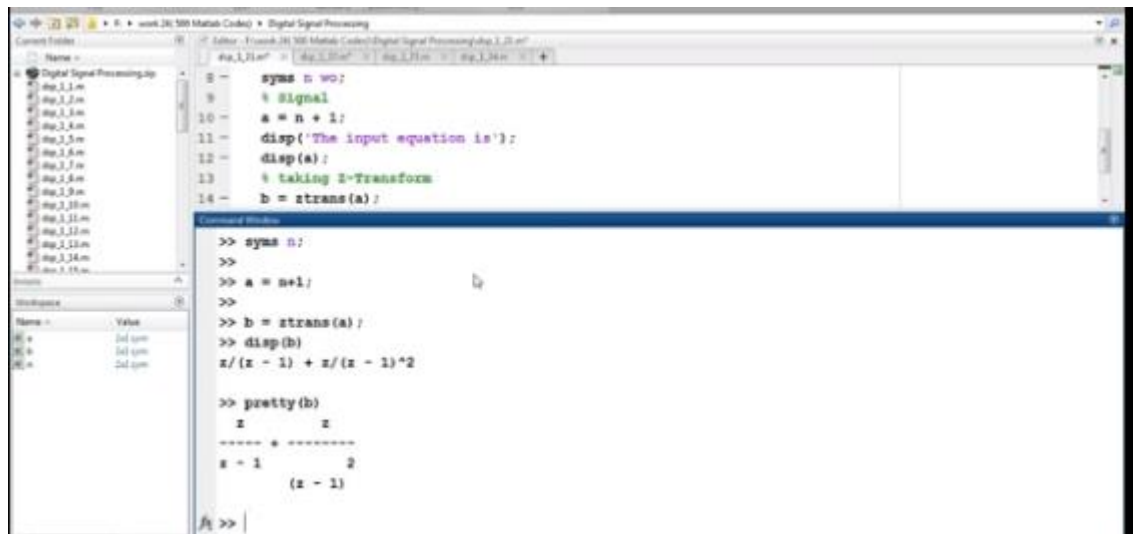
Specify both the independent and transformation variables as `m` and `y` in the second and third arguments, respectively.

```
ztrans(f,m,y)
ans =
(y*exp(n))/(y - exp(1))
```

Compute the Z-transform of the Heaviside function and the binomial coefficient.

```
syms n z
ztrans(heaviside(n-3),n,z)
ans =
(1/(z - 1) + 1/2)/z^3
ztrans(nchoosek(n,2))
```

ans =  
 $z/(z - 1)^3$



The image shows a MATLAB script editor window titled "Digital Signal Processing" with a script file named "ans\_1\_21.m". The script contains the following code:

```
8 - syms n wo;  
9 - % signal  
10 - a = n + 1;  
11 - disp('The input equation is');  
12 - disp(a);  
13 - % taking Z-Transform  
14 - b = ztrans(a);
```

The Command Window shows the execution of the script:

```
>> syms n;  
>>  
>> a = n+1;  
>>  
>> b = ztrans(a);  
>> disp(b)  
z/(z - 1) + z/(z - 1)^2  
  
>> pretty(b)  
      z      z  
----- + -----  
z - 1      (z - 1)^2
```

The Workspace window shows the following variables:

Name	Value
a	2nd sym
b	2nd sym
n	2nd sym

## DAILY ASSIGEMENTS DETAILS:

<b>Date:</b>	<b>26/05/2020</b>	<b>Name:</b>	<b>CHANDANA.R</b>
<b>Course:</b>	Python	<b>USN:</b>	<b>4AL16EC017</b>
<b>Topic:</b>	Steps of the flask	<b>Semester &amp; Section:</b>	<b>8(A)</b>
<b>Github Repository:</b>	<b>Chandana-shaiva</b>		

### AFTERNOON SESSION

#### Report:

#### Step #1: Start with Flask

Flask is a new source, followed a tutorial to get started but the general idea is to create a simple `myapp.py`

```
from flask import Flask, render_template
```

```
app = Flask(__name__)
```

```
@app.route("/")
```

```
def home():
```

```
    return
```

```
render_template('home.html')
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

You also want to create a very basic

`home.html`

template:

```
<!DOCTYPE html>
```

```
<html>
```

```
<head></head>
```

```
<body>
<div>
<h1>
Jane Doe
</h1>
<p>
Hello, I'm a super girl.
</p>
</div>
</body>
</html>
```

make it fancier, i.e. pass variables via `render_template` and use them in the html file, reference an external CSS file instead of some controversial in-line CSS, and so on. However, if you go ahead and run `python myapp.py`, you should see that the website is up and running on your local host — if this is too minimal, you can always go back and improve it incrementally later.

## Step #2: Static-ify with Frozen-Flask

If you think once again about it, what we want to build is somewhere where to display our bio and some links, and really nothing more. In other words, our end goal is to make a static website, and we can leverage Frozen-Flask to freeze a Flask application into a set of static files.

Frozen-Flask is only about deployment: instead of installing Python and Flask on your server, you can use Frozen-Flask to freeze your application and only have static html files on your server.

It's magic!

(Ok, not really magic, but that's quite impressive, right?)

I found this article quite clear and helpful, but in a nutshell, after installing the Frozen-Flask package, go ahead and create a `freeze.py` that looks something like this

```
From myapp import app
Freezer = Freezer ( app )
If __name__ == '__main__':
Freezer. Freeze ()
```

Running this script via `python freeze.py` will create a build directory containing your application's content, frozen into static files, which is exactly what we want.

## Step #3: Deploy with Netlify

The final step is to host the website somewhere on the internet — and not only on our local host. To do so, we can leverage Netlify, whose main purpose is indeed to build, deploy, and manage modern web projects. As suggested by the official documentation, it's usually a good idea to create a `requirements.txt` and a `runtime.txt`.

The first will tell Netlify which packages are needed by your application, and can be created via:

```
pip freeze > requirements.txt
```

The second file is simply a record of the Python version that you want to run

When you're ready to deploy your website, head over to the Netlify website, sign up, and choose "create a new site from GitHub". The Deploy settings are especially interesting and might not be the most straightforward ones to fill out: Deploy settings that I used for my personal website Pay attention in particular to:

- Build command, a.k.a. the command we want Netlify to run. In our case, since we want to simply freeze our html so that it creates a static website for us, let's set it to `python freeze.py`
- Public directory, a.k.a. the directory with the static content of the website. We set this to Build because that's where Frozen-Flask puts all the static content after generating it.
- After saving these settings, you can finally trigger the build of your website, and ta-da! After few seconds, your personal page will be available at `<site-name>.netlify.com`, but if you want, you can also buy your own domain if you want to make it look even fancier.
- 

## CSS STYLING:

```
body {  
    margin: 0;  
    padding: 0;  
    font-family: "Helvetica Neue", Helvetica, Arial, sans-serif;  
    color: #060;  
}
```

```
/*
```

```
* Formatting the header area
*/
```

```
header {
  background-color: #DFB887;
  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;
}
```

```
/*
* Center the body content
*/
```

```
.container {
  width: 1200px;
  margin: 0 auto;
}
```

```
div.home {
  padding: 10px 0 30px 0;
  background-color: #E6E6FA;
  -webkit-border-radius: 6px;
  -moz-border-radius: 6px;
  border-radius: 6px;
}
```

```
div.about {
```

```
padding: 10px 0 30px 0;
background-color: #E6E6FA;
-webkit-border-radius: 6px;
-moz-border-radius: 6px;
border-radius: 6px;
}
```

```
h2 {
font-size: 3em;
margin-top: 40px;
text-align: center;
letter-spacing: -2px;
}
```

```
h3 {
font-size: 1.7em;
font-weight: 100;
margin-top: 30px;
text-align: center;
letter-spacing: -1px;
color: #999;
}
```

```
.menu {
float: right;
margin-top: 8px;
}
```

```
.menu li {
display: inline;
}
```

```
.menu li + li {
margin-left: 35px;
}
```

```
.menu li a {
color: #444;
text-decoration: none;
}
```

## Steps to deploy a static Flask website to Heroku:

- Create an account on [www.heroku.com](http://www.heroku.com) if you don't have one already.
- Download and install Heroku Toolbelt from <https://devcenter.heroku.com/articles/heroku-cli>
- Install gunicorn with "pip install gunicorn". Make sure you're using pip from your virtual environment if you have one.
- 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.
- 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.
- 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.
- Open your computer terminal/command line to point to the directory where the Python file containing your app code is located.
- Using the terminal, log in to Heroku with "heroku login"
- Enter your Heroku email address
- Enter your Heroku password
- Create a new Heroku app with "heroku create myawesomeappname"
- Initialize a local git repository with "git init"
- Add your local application files to git with "git add ."



- Tell git your email address with "git config --global user.email \"myemail@hotmail.com\"". Make sure the email address is inside quotes here.
- Tell git your username (just pick whatever username) with "git config --global user.name \"whateverusername\"". The username should be in quotes
- Commit the changes with "git commit -m \"first commit\"". Make sure "first commit" is inside quotes.
- Before pushing the changes to Heroku, tell heroku the name of the app you want to use with "heroku git:remote --app myawesomeappname"
- Push the changes to Heroku with "git push heroku master"
- That should do it. Go ahead and open your app with "heroku open".