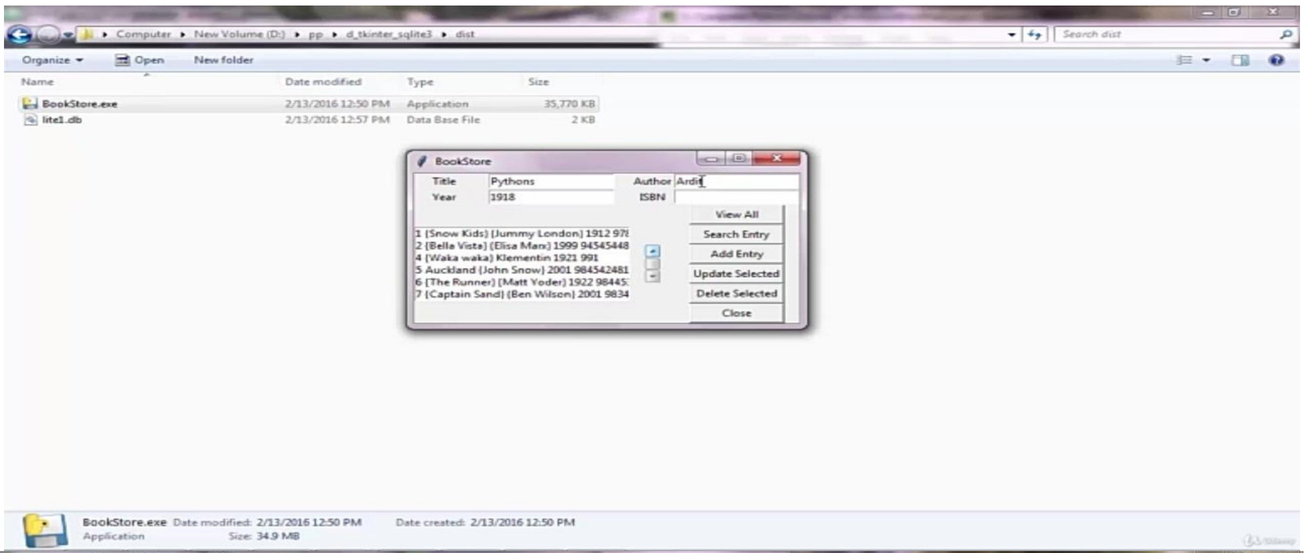
# DAILY ASSESSMENT FORMAT

| | | | |
|---|---|---|---|
| **Date:** | 27th May 2020 | **Name:** | Soundarya NA |
| **Course:** | UDEMY | **USN:** | 4AL16EC077 |
| **Topic:** | PYTHON:<br>Application 5: Build a Desktop Database Application | **Semester & Section:** | 8th - B |

**FORENOON SESSION DETAILS**

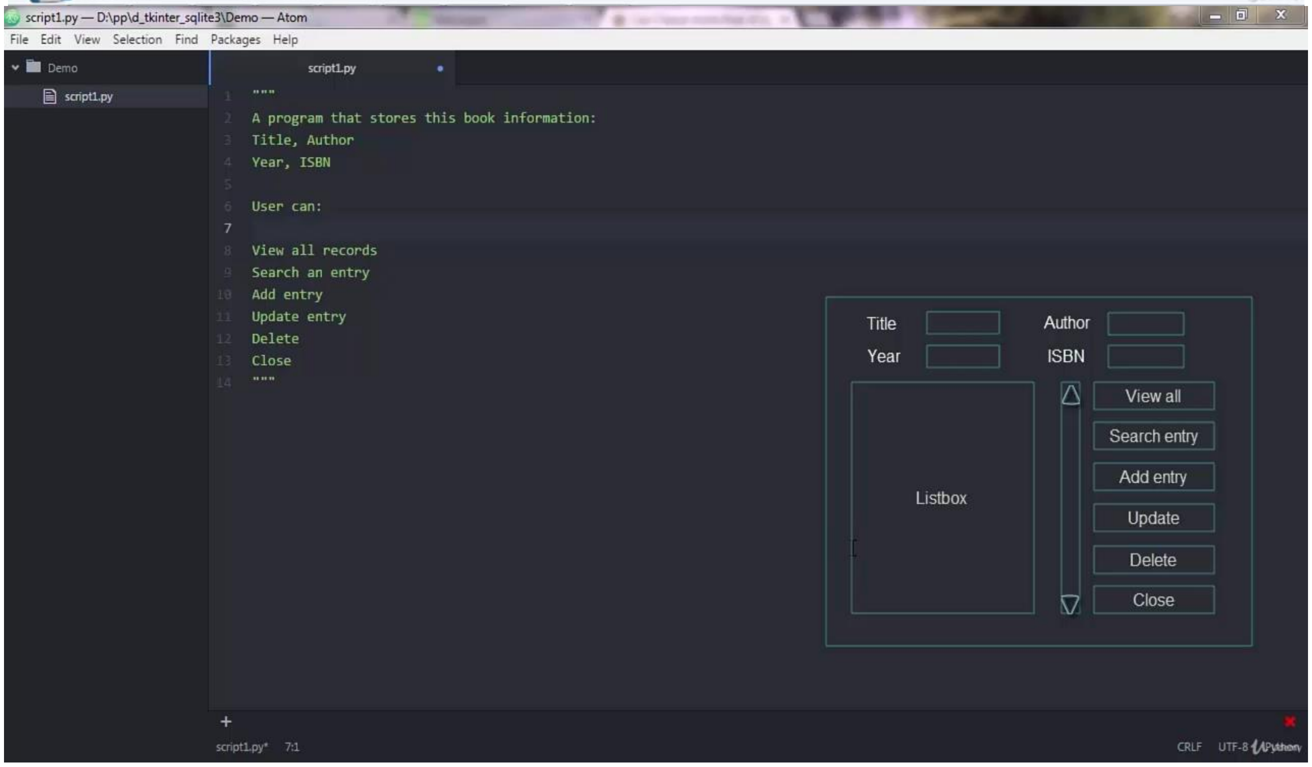
**Image of session**



| Name | Date modified | Type | Size |
|---|---|---|---|
| BookStore.exe | 2/13/2016 12:50 PM | Application | 35,770 KB |
| lite1.db | 2/13/2016 12:57 PM | Data Base File | 2 KB |

Title: Pythons
Year: 1918
Author: Ardi

```
1 (Snow Kids) (Jummy London) 1912 97
2 (Bella Vista) (Elisa Marx) 1999 94545448
4 (Waka waka) Klementin 1921 991
5 Auckland (John Snow) 2001 984542481
6 (The Runner) (Matt Yoder) 1922 98445
7 (Captain Sand) (Ben Wilson) 2001 9834
```



script1.py — D:\pp\d_tkinter_sqlite3\Demo — Atom
File Edit View Selection Find Packages Help
Demo
script1.py
Title
Author
Year
ISBN
Listbox
View all
Search entry
Add entry
Update
Delete
Close

```
"""
A program that stores this book information:
Title, Author
Year, ISBN

User can:

View all records
Search an entry
Add entry
Update entry
Delete
Close
"""
```

**Report:**

**Build a Desktop Database Application:**

This course will help you build a Python GUI application step-by-step with Tkinter, SQLite. and the free Coin Market Cap API. Throughout the course, you will learn more about Python and Tkinter, including:

• Understanding API and its usage.

• Extracting cryptocurrency coin data and working with it.

• Building portfolio logic on the command line then focusing on the GUI

• Getting started with Tkinter and Python GUI

• Merging your command-line application with Tkinter and completing the.py version.

• Converting the.py application to an executable .exe app.

• Understanding the basics of SQLite3 with Python.

**Features:**

- Check different Cryptocurrency coin prices with a real API
- Build Portfolio logic and functionality from scratch
- Convert the.py application to an executable .exe app that can be used in real life

**Frontend Interface:**

The part of a website that user interacts with directly is termed as front end. It is also referred to as the 'client side' of the application. It includes everything that users experience directly: text colors and styles, images, graphs and tables, buttons, colors, and navigation menu. HTML, CSS, and Javascript are the languages used for Front End development. The structure, design, behavior, and content of everything seen on browser screen when websites, web applications, or mobile apps are opened up, is implemented by front End developers. Responsiveness and performance are two main objectives of the front End. The developer must ensure that the site is responsive i.e. it appears correctly on devices of all sizes no part of the website should behave abnormally irrespective of the size of the screen.

**Front end Languages:** The frontend portion is built by using some languages which are discussed below:

**HTML:** HTML stands for Hyper Text Markup Language. It is used to design the front end portion of web pages using markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. The markup language is used to define the text documentation within tag which defines the structure of web pages.

**CSS:** Cascading Style Sheets fondly referred to as CSS is a simply designed language intended to simplify the process of making web pages presentable. CSS allows you to apply styles to web pages. More importantly, CSS enables you to do this independent of the HTML that makes up each web page.

**JavaScript:** JavaScript is a famous scripting language used to create the magic on the sites to make the site interactive for the user. It is used to enhancing the functionality of a website to running cool games and web-based software.

**Front End Frameworks and Libraries:**

**AngularJS:** Angular is a JavaScript open source front-end framework that is mainly used to develop single page web applications (SPAs). It is a continuously growing and expanding framework which provides better ways for developing web applications. It changes the static HTML to dynamic HTML. It is an open source project which can be freely used and changed by anyone. It extends HTML attributes with Directives, and data is bound with HTML.

**React.js:** React is a declarative, efficient, and flexible JavaScript library for building user interfaces. ReactJS is an open-source, component-based front-end library responsible only for the view layer of the application. It is maintained by Facebook.

**Bootstrap:** Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first web sites.

**jQuery:** jQuery is an open source JavaScript library that simplifies the interactions between an HTML/CSS document, or more precisely the Document Object Model (DOM), and JavaScript. Elaborating the terms, jQuery simplifies HTML document traversing and manipulation, browser event handling, DOM animations, Ajax interactions, and cross-browser JavaScript development.

**SASS:** It is the most reliable, mature and robust CSS extension language. It is used to extend the functionality of an existing CSS of a site including everything from variables, inheritance, and nesting with ease.

Some other libraries and frameworks are: Semantic-UI, Foundation, Materialize, Backbone.js, Express.js, Ember.js etc.

**Backend:**

Backend is server side of the website. It stores and arranges data, and also makes sure everything on the client-side of the website works fine. It is the part of the website that you cannot see and interact with. It is the portion of software that does not come in direct contact with the users. The parts and characteristics developed by backend designers are indirectly accessed by users through a front-end application. Activities, like writing APIs, creating libraries, and working with system components without user interfaces or even systems of scientific programming, are also included in the backend.

**Back end Languages:** The back-end portion is built by using some languages which are discussed below:

**PHP:** PHP is a server-side scripting language designed specifically for web development. Since PHP code executed on the server side so it is called server-side scripting language.

**C++:** It is a general-purpose programming language and widely used now a days for competitive programming. It is also used as backend language.

**Java:** Java is one of the most popular and widely used programming language and platform. It is highly scalable. Java components are easily available.

**Python:** Python is a programming language that lets you work quickly and integrate systems more efficiently.

**JavaScript:** Javascript can be used as both (front end and back end) programming languages.

**Node.js:** Node.js is an open source and cross-platform runtime environment for executing JavaScript code outside of a browser. You need to remember that NodeJS is not a framework and it's not a programming language. Most of the people are confused and understand it's a framework or a programming language. We often use Node.js for building back-end services like APIs like Web App or Mobile App. It's used in production by large companies such as Paypal, Uber, Netflix, Wallmart and so on.

**Back End Frameworks:**

- The list of back end frameworks are: Express, Django, Rails, Laravel, Spring, etc.
- The other back end program/scripting languages are: C#, Ruby, REST, GO etc.

**Difference between Frontend and Backend:**

Frontend and backend development are quite different from each other, but still, they are two aspects of the same situation. The frontend is what users see and interact with and backend is how everything works.

- Frontend is the part of the website users can see and interact with such as the graphical user interface (GUI) and the command line including the design, navigating menus, texts, images, videos, etc. Backend, on the contrary, is the part of the website users cannot see and interact with.
- The visual aspects of the website that can be seen and experienced by users are frontend. On the other hand, everything that happens on the background can be attributed to the backend.
- Languages used for front end are HTML, CSS, Javascript while those used for backend include Java, Ruby, Python, .Net.

**Connecting the Frontend to the Backend:**

**Code:**

**“Frontend”**

```
<!DOCTYPE html>
<script type="text/javascript" src="http://code.jquery.com/jquery-latest.min.js"></script>
<script src="javaScript.js"></script>

<html>

<div id="liveOutputTitle">
<h1><span>Live Output</span></title>
</div>

<div id="liveOutput">
Value 1 From Python: <input type="text" name="Value1" id="Value1" value="0"><br>

Value 2 From Python: <input type="text" name="Value2" id="Value2" value="0"><br>

Value 3 From Python: <input type="text" name="Value3" id="Value3" value="0"><br>
```

```
</div>
```

**"Sever side"**

```
#!/usr/bin/env python

def getValueOne():
   //gets the value from wherever
   valueOne = 1

def getValueTwo():
   //gets the value from wherever
   valueTwo = 2

def getValueThree():
   //gets the value from wherever
   valueThree = 3
```

**"URL associated to view in backend"**

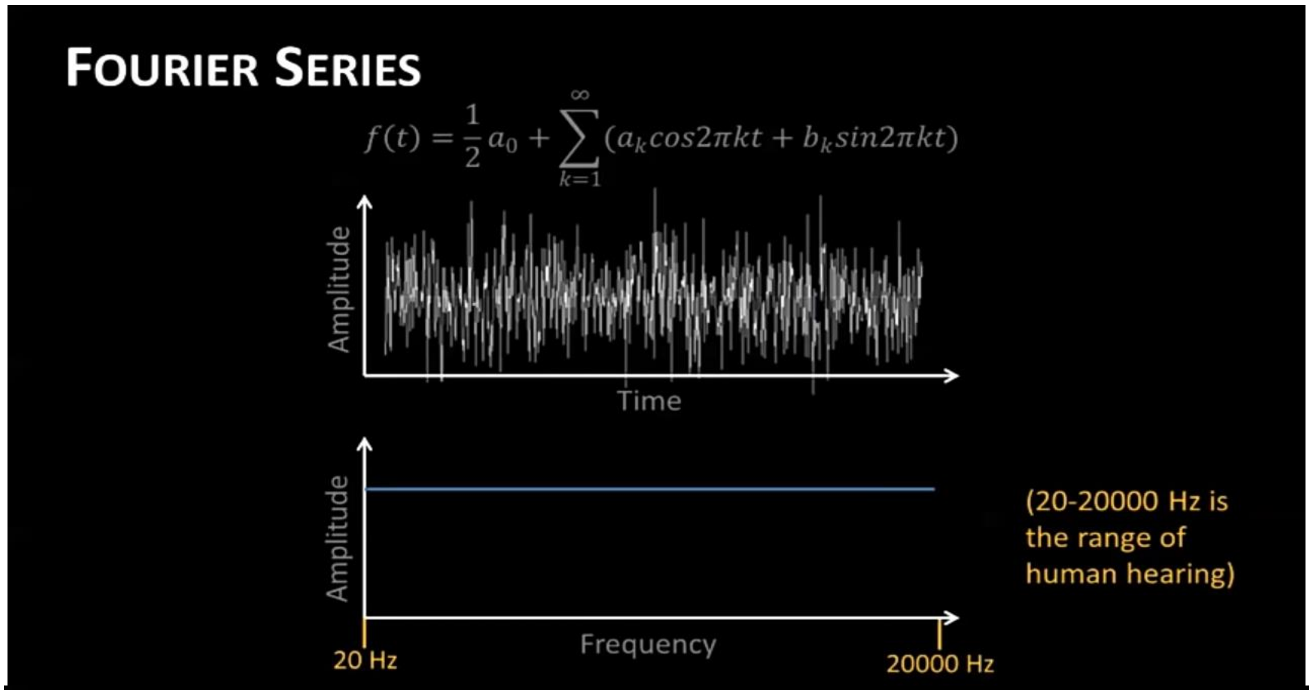
```
import json
def getUsers():
   users = db.get_users() # just an example
   return json.dumps(users)
```

**Creating a Standalone Executable Version of the Program:**

```
from distutils.core import setup
 import py2exe

 from distutils.filelist import findall
 import matplotlib

 setup(
      console=['PlotMemInfo.py'],
  options={
          'py2exe': {
          'packages' : ['matplotlib'],
     'dll_excludes': ['libgdk-win32-2.0-0.dll',
                      'libgobject-2.0-0.dll',
           'libgdk_pixbuf-2.0-0.dll']
                    }
          },
  data_files = matplotlib.get_py2exe_datafiles()
)
```

|  |  |  |  |
|---|---|---|---|
| **Date:** | 27th May 2020 | **Name:** | Soundarya NA |
| **Course:** | DSP | **USN:** | 4AL16EC077 |
| **Topic:** | Digital Signal Processing | **Semester & Section:** | 8th - B |

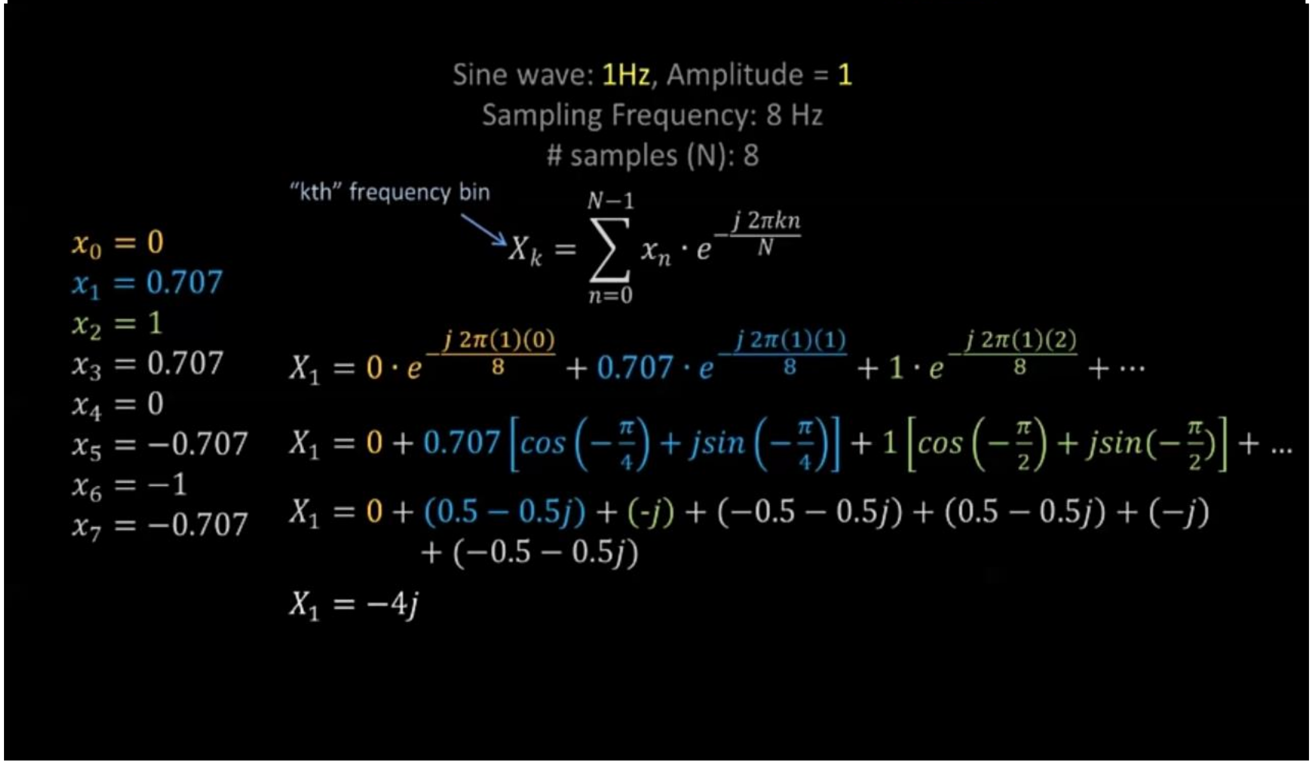
**Image:**



## Fourier Series

$$f(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty}(a_k cos2\pi kt + b_k sin2\pi kt)$$

(20-20000 Hz is the range of human hearing)



Sine wave: 1Hz, Amplitude = 1

Sampling Frequency: 8 Hz

# samples (N): 8

"kth" frequency bin

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{j\,2\pi kn}{N}}$$

$x_0 = 0$
$x_1 = 0.707$
$x_2 = 1$
$x_3 = 0.707$
$x_4 = 0$
$x_5 = -0.707$
$x_6 = -1$
$x_7 = -0.707$

$$X_1 = 0 \cdot e^{-\frac{j\,2\pi(1)(0)}{8}} + 0.707 \cdot e^{-\frac{j\,2\pi(1)(1)}{8}} + 1 \cdot e^{-\frac{j\,2\pi(1)(2)}{8}} + \cdots$$

$$X_1 = 0 + 0.707\left[cos\left(-\frac{\pi}{4}\right) + jsin\left(-\frac{\pi}{4}\right)\right] + 1\left[cos\left(-\frac{\pi}{2}\right) + jsin(-\frac{\pi}{2})\right] + \ldots$$

$$X_1 = 0 + (0.5 - 0.5j) + (-j) + (-0.5 - 0.5j) + (0.5 - 0.5j) + (-j) + (-0.5 - 0.5j)$$

$$X_1 = -4j$$

**Report:**

**Fourier Transforms:**

The Fourier transform of a function f is traditionally denoted f, by adding a circumflex to the symbol of the function. There are several common conventions for defining the Fourier transform of an integrable function f: R->C One of them is

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\, dx,$$

**Properties of Fourier Transform:**

**Linearity:**

For any complex numbers a and b, if $h(x) = af(x) + bg(x)$, then $\hat{h}(\xi) = a \cdot \hat{f}(\xi) + b \cdot \hat{g}(\xi)$.

**Translation / time shifting:**

For any real number $x_0$, if $h(x) = f(x - x_0)$, then $\hat{h}(\xi) = e^{-2\pi i x_0 \xi} \hat{f}(\xi)$.

**Modulation / frequency shifting:**

For any real number $\xi_0$, if $h(x) = e^{2\pi i x \xi_0} f(x)$, then $\hat{h}(\xi) = \hat{f}(\xi - \xi_0)$.

**Time scaling:**

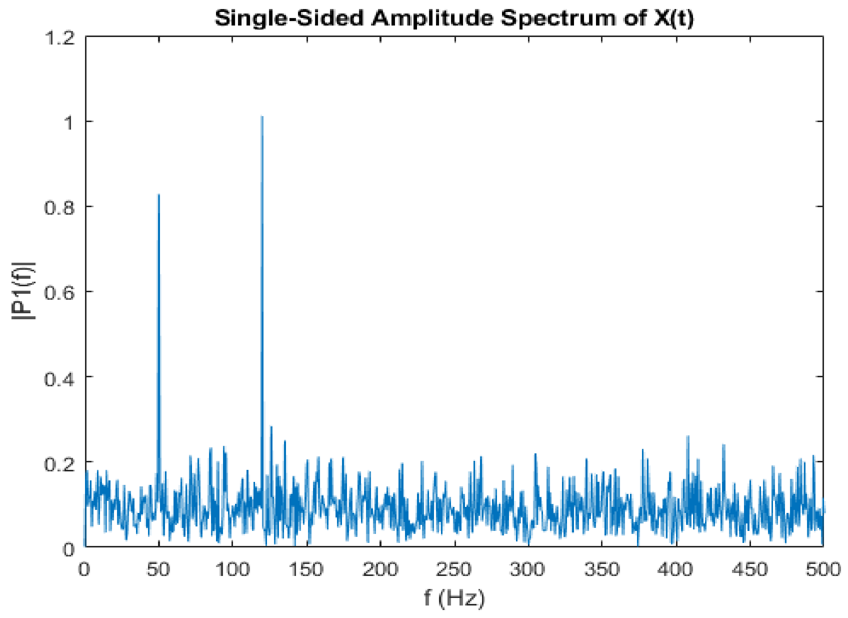
For a non-zero real number $a$, if $h(x) = f(ax)$

**Conjugation:**

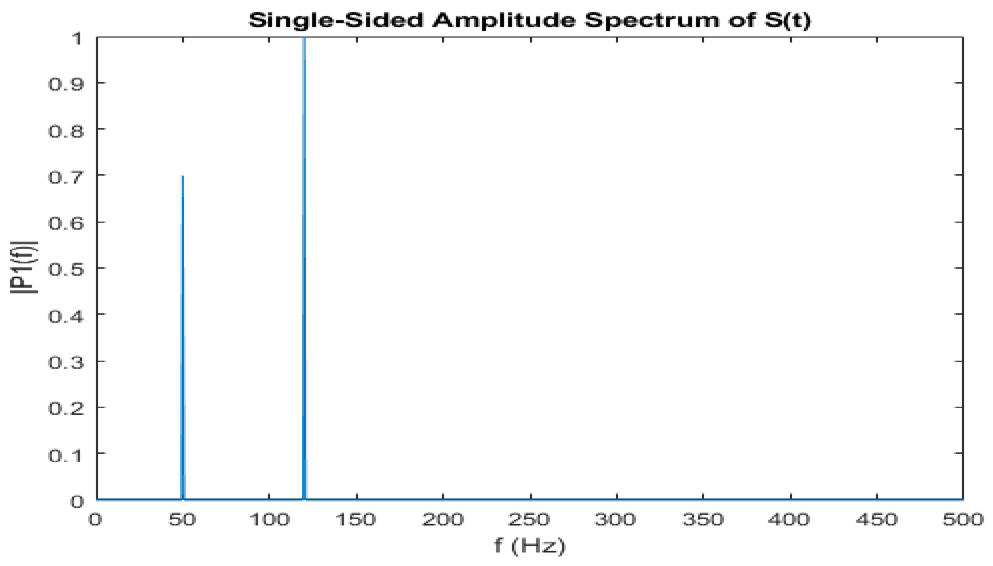
h(x) = f (x)

**FFT Fast Fourier Transform MATLAB:**

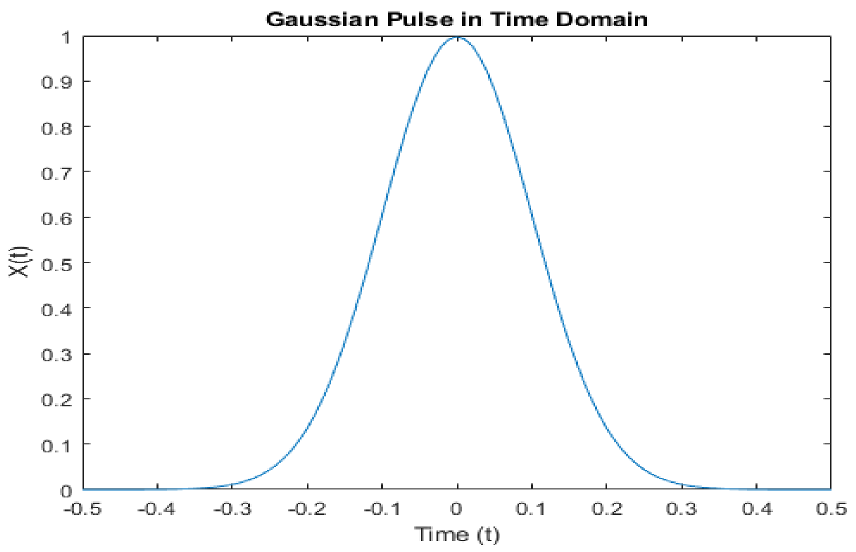
```
f = Fs*(0:(L/2))/L;

plot(f,P1)

title('Single-Sided Amplitude Spectrum of X(t)')

xlabel('f (Hz)')

ylabel('|P1(f)|')
```



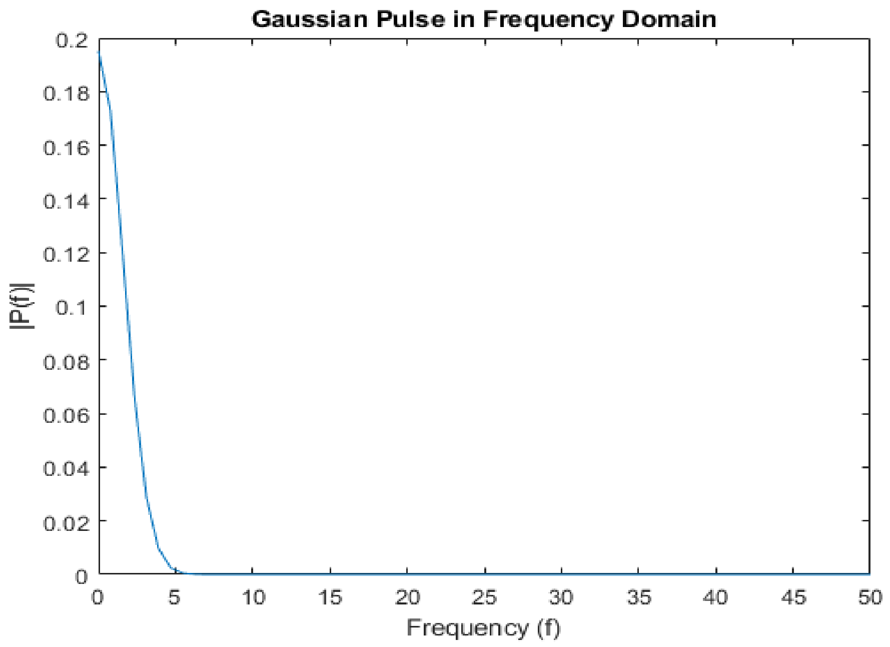
```
Y = fft(S);

P2 = abs(Y/L);

P1 = P2(1:L/2+1);

P1(2:end-1) = 2*P1(2:end-1);

plot(f,P1)

title('Single-Sided Amplitude Spectrum of S(t)')

xlabel('f (Hz)')

ylabel('|P1(f)|')
```



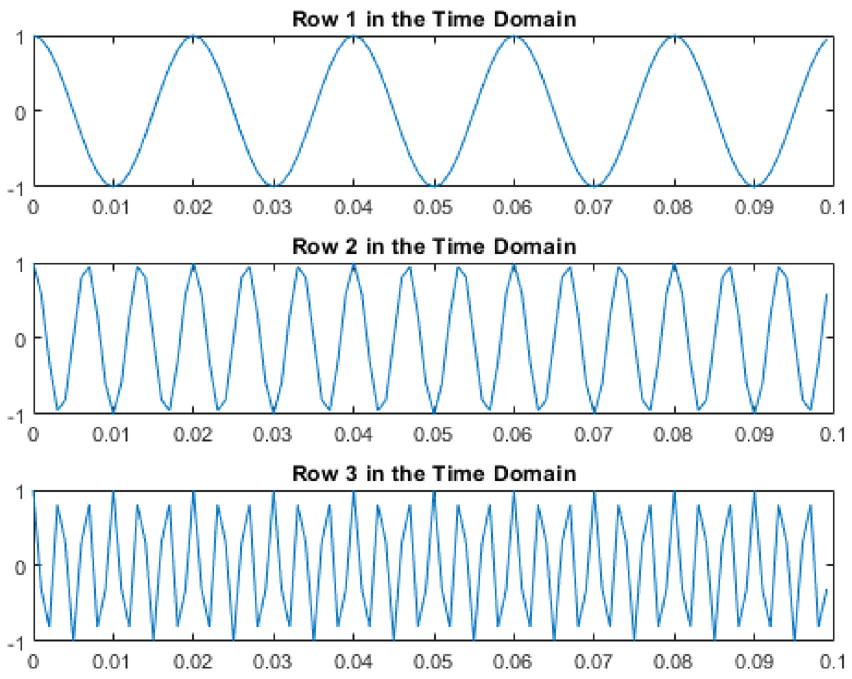
```
Fs = 100;          % Sampling frequency

t = -0.5:1/Fs:0.5;  % Time vector

L = length(t);      % Signal length

X = 1/(4*sqrt(2*pi*0.01))*(exp(-t.^2/(2*0.01)));

plot(t,X)

title('Gaussian Pulse in Time Domain')

xlabel('Time (t)')

ylabel('X(t)')
```



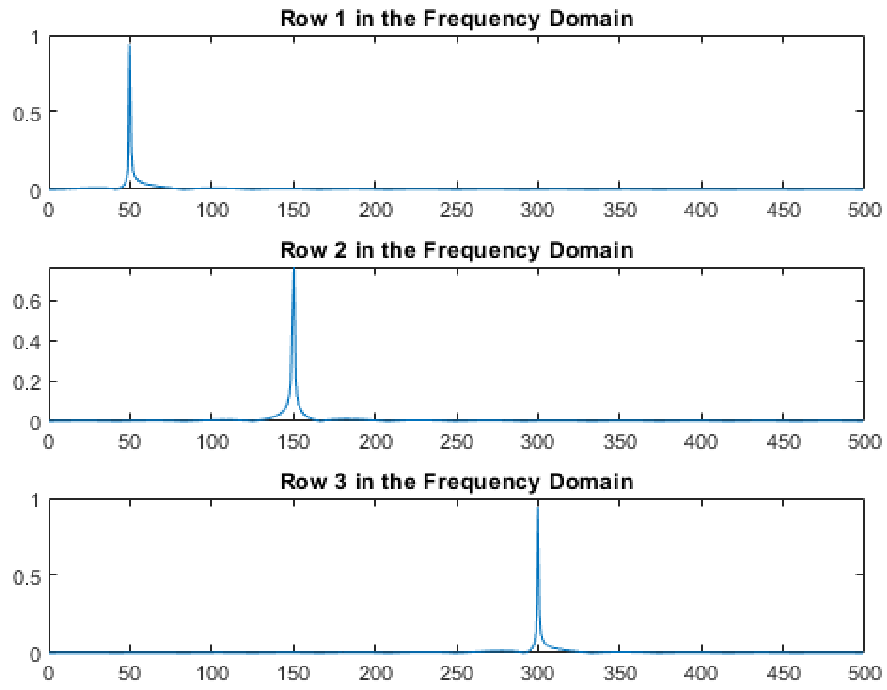
```
f = Fs*(0:(n/2))/n;

P = abs(Y/n);


plot(f,P(1:n/2+1))

title('Gaussian Pulse in Frequency Domain')

xlabel('Frequency (f)')

ylabel('|P(f)|')
```



```
Fs = 1000;                    % Sampling frequency
T = 1/Fs;                    % Sampling period
L = 1000;                    % Length of signal
t = (0:L-1)*T;               % Time vector
x1 = cos(2*pi*50*t);         % First row wave
x2 = cos(2*pi*150*t);        % Second row wave
x3 = cos(2*pi*300*t);        % Third row wave

X = [x1; x2; x3];
for i = 1:3
   subplot(3,1,i)
   plot(t(1:100),X(i,1:100))
   title(['Row ',num2str(i),' in the Time Domain'])
end
```



```
n = 2^nextpow2(L);
dim = 2;
Y = fft(X,n,dim);
P2 = abs(Y/L);
P1 = P2(:,1:n/2+1);
P1(:,2:end-1) = 2*P1(:,2:end-1);
for i=1:3
   subplot(3,1,i)
   plot(0:(Fs/n):(Fs/2-Fs/n),P1(i,1:n/2))
   title(['Row ',num2str(i),' in the Frequency Domain'])
end
```



**Introduction to WT:**
**Code:**

```
#include <Wt/WApplication.h>

#include <Wt/WBreak.h>

#include <Wt/WContainerWidget.h>

#include <Wt/WLineEdit.h>

#include <Wt/WPushButton.h>

#include <Wt/WText.h>


class HelloApplication : public Wt::WApplication

{

public:

  HelloApplication(const Wt::WEnvironment& env);


private:

  Wt::WLineEdit *nameEdit_;

  Wt::WText *greeting_;

};
```

```
HelloApplication::HelloApplication(const Wt::WEnvironment& env)
  : Wt::WApplication(env)
{
  setTitle("Hello world");

  root()->addWidget(std::make_unique<Wt::WText>("Your name, please? "));
  nameEdit_ = root()->addWidget(std::make_unique<Wt::WLineEdit>());
  Wt::WPushButton *button = root()->addWidget(std::make_unique<Wt::WPushButton>("Greet
me."));
  root()->addWidget(std::make_unique<Wt::WBreak>());
  greeting_ = root()->addWidget(std::make_unique<Wt::WText>());
  auto greet = [this]{
    greeting_->setText("Hello there, " + nameEdit_->text());
  };
  button->clicked().connect(greet);
}

int main(int argc, char **argv)
{
  return Wt::WRun(argc, argv, [](const Wt::WEnvironment& env) {
    return std::make_unique<HelloApplication>(env);
  });
}
```

**Welch's method and windowing:**

```
pxx = pwelch(x)
pxx = pwelch(x,window)
pxx = pwelch(x,window,noverlap)
pxx = pwelch(x,window,noverlap,nfft)
[pxx,w] = pwelch(___)
```

```
[pxx,f] = pwelch(___,fs)

[pxx,w] = pwelch(x,window,noverlap,w)

[pxx,f] = pwelch(x,window,noverlap,f,fs)

[___] = pwelch(x,window,___,freqrange)

[___] = pwelch(x,window,___,trace)

[___,pxxc] = pwelch(___,'ConfidenceLevel',probability)

[___] = pwelch(___,spectrumtype)

pwelch(___)
```