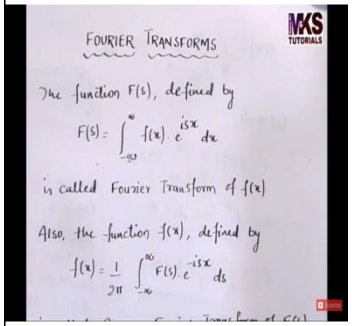
DAILY ASSESSMENT FORMAT

Date:	27-May-2020	Name:	Shilpa S
Course:	TCS iON	USN:	4AL14EC078
Topic:	DSP	Semester & Section:	8th sem & 'A' section
Github Repository:	Shilpa_online		

MORNING SESSION DETAILS
Image of session


FOURIER TRANSFORMS

The function F(s), defined by

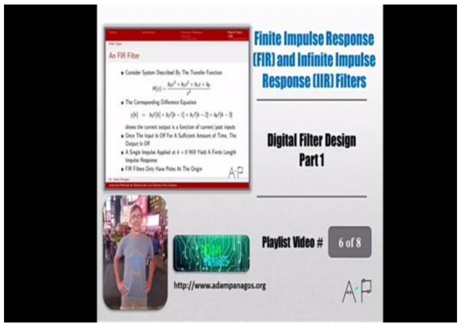
$$F(s) = \int_{-\infty}^{\infty} f(x) \cdot e^{isx} \, dx$$

is called Fourier Transform of f(x)

Also, the function f(x), defined by

$$f(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(s) \cdot e^{-isx} \, ds$$

MKS TUTORIALS


Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) Filters

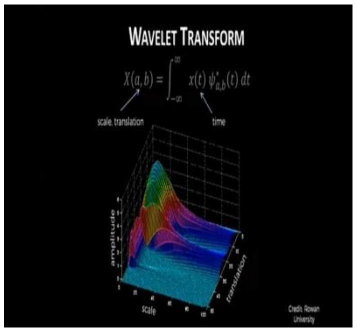
Digital Filter Design Part 1

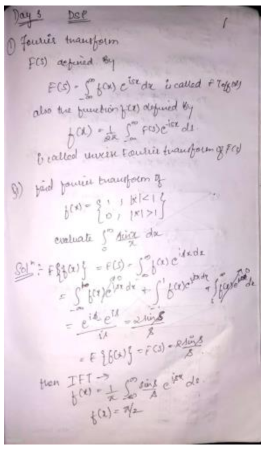
Playlist Video # 6 of 8

http://www.adampanagos.org

An FIR Filter

- Consider System Described By The Transfer Function
- The Corresponding Difference Equation
- shows the current output is a function of current/past inputs
- Once The Input Is Off For A Sufficient Amount of Time, The Output Is Off
- A Single Impulse Applied at k = 0 Will Yield A Finite Length Impulse Response
- FIR Filters Only Have Poles At The Origin





Day 3 DSP

1

① Fourier transform

F(s) defined by

$$F(s) = \int_{-\infty}^{\infty} f(x)\, e^{isx}\, dx$$ is called F T of f(x)

also the function f(x) defined by

$$f(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(s)\, e^{-isx}\, ds$$

is called inverse Fourier transform of F(s)

Q) find fourier transform of

$$f(x) = \begin{cases} 1, & |x|<1 \\ 0, & |x|>1 \end{cases}$$

evaluate $\int_{0}^{\infty} \frac{\sin x}{x}\, dx$

Soln :- $F\{f(x)\} = F(s) = \int_{-\infty}^{\infty} f(x)\, e^{isx} dx$

$$= \int_{-\infty}^{-1} \overset{0}{f(x)} e^{isx} dx + \int_{-1}^{1} f(x) e^{isx} dx + \int_{1}^{\infty} \overset{0}{f(x)} e^{isx} dx$$

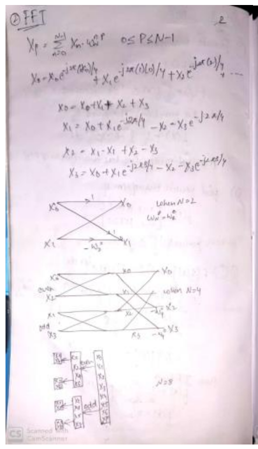
$$= \frac{e^{is} - e^{-is}}{is} = \frac{2\sin s}{s}$$

$$= F\{f(x)\} = F(s) = \frac{2\sin s}{s}$$

then IFT →

$$f(x) = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{\sin s}{s} e^{isx} ds$$

$$f(x) = \pi/2$$



② FFT

2

$$X_p = \sum_{n=0}^{N-1} x_n \cdot W_N^{np} \quad 0 \le P \le N-1$$

$$X_0 = x_0 e^{-j2\pi(0)(0)/4} + x_1 e^{-j2\pi(1)(0)/4} + x_2 e^{-j2\pi(2)/4} + \cdots$$

$$X_0 = x_0 + x_1 + x_2 + x_3$$

$$X_1 = x_0 + x_1 e^{-j2\pi/4} - x_2 - x_3 e^{-j2\pi/4}$$

$$X_2 = x_1 - x_1 + x_2 - x_3$$

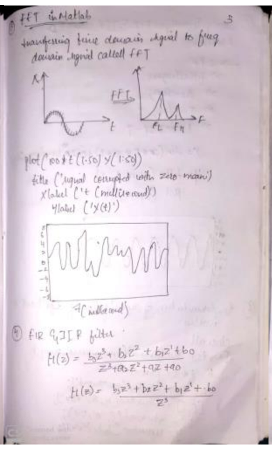
$$X_3 = x_0 + x_1 e^{-j2\pi 3/4} - x_2 - x_3 e^{-j2\pi 3/4}$$

when N=2

$W_N^0 = W_2^0$

when N=4

N=8



③ FFT in Matlab

3

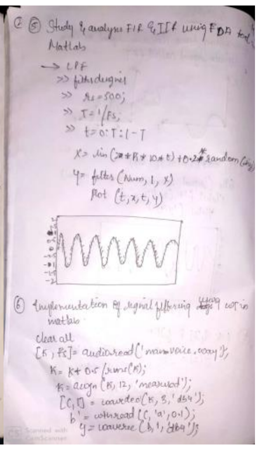
transferring time domain signal to freq domain signal called FFT

```
plot(1000*t(1:50), y(1:50))
title('signal corrupted with zero mean')
xlabel('t (milliseconds)')
ylabel('y(t)')
```

④ FIR & IIR filter

$$H(z) = \frac{b_3z^3 + b_2z^2 + b_1z^1 + b_0}{z^3 + a_2z^2 + a_1z + a_0}$$

$$H(z) = \frac{b_3z^3 + b_2z^2 + b_1z^1 + b_0}{z^3}$$



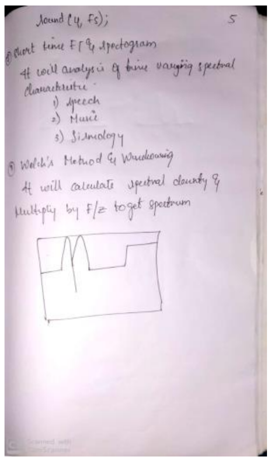
② ⑤ Study & analyse FIR & IIR using FDA tool Matlab

→ LPF

```
>> filterdesigner
>> Fs=500;
>> T=1/Fs;
>> t=0:T:1-T
x= sin(2*Pi*10*t)+0.2*randn(size(t))
y= filter(Num,1,x)
plot (t,x,t,y)
```

⑥ Implementation of signal filtering using DWT in matlab

```
clear all
[x, Fs]= audioread('manvoice.wav');
x= x+ 0.5/rms(x);
x= awgn(x,12,'measured');
[C,L] = wavedec(x,3,'db4');
b' = wthresh(C,'a',0.1);
y= waverec(b,L,'db4');
```



```
sound(y, Fs);
```

5

⑦ Short time FT & spectogram

It will analysis of time varying spectral characteristic

1) speech
2) Music
3) Sismology

⑧ Welch's Method & Windowing

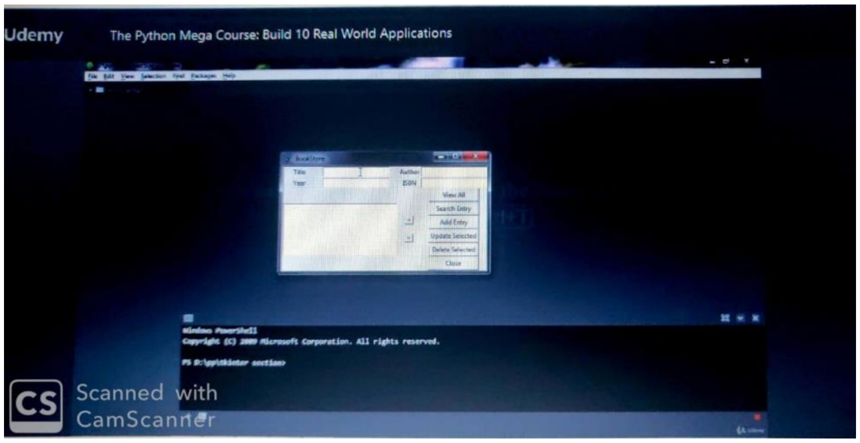
It will calculate spectral density & multiply by F/2 to get spectrum

DAILY ASSESSMENT FORMAT

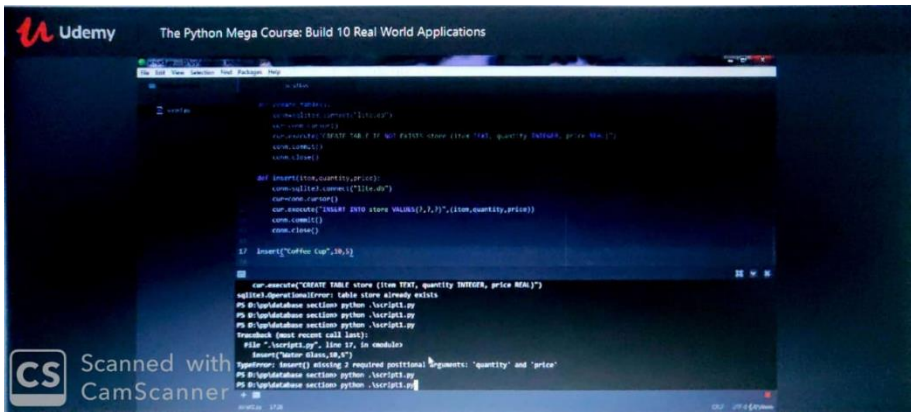
Date:	**27-5-2020**	**Name:**	**Shilpa S**
Course:	**Python programming**	**USN:**	**4AL14EC078**
Topic:	**Graphical user interface with tkinter,Interacting with data base**	**Semester & Section:**	**8th A**
Github Repository:	**Shilpa_online**		

AFTERNOON SESSION DETAILS

Image of session



```
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS D:\pp\tkinter section>
```


Udemy
The Python Mega Course: Build 10 Real World Applications
File Edit View Selection Find Packages Help
Scanned with CamScanner

```
def create_table():
    conn=sqlite3.connect("lite.db")
    cur=conn.cursor()
    cur.execute("CREATE TABLE IF NOT EXISTS store (item TEXT, quantity INTEGER, price REAL)")
    conn.commit()
    conn.close()

def insert(item,quantity,price):
    conn=sqlite3.connect("lite.db")
    cur=conn.cursor()
    cur.execute("INSERT INTO store VALUES(?,?,?)",(item,quantity,price))
    conn.commit()
    conn.close()

insert("Coffee Cup",10,5)
```

```
    cur.execute("CREATE TABLE store (item TEXT, quantity INTEGER, price REAL)")
sqlite3.OperationalError: table store already exists
PS D:\pp\database section> python .\script1.py
PS D:\pp\database section> python .\script1.py
PS D:\pp\database section> python .\script1.py
Traceback (most recent call last):
  File ".\script1.py", line 17, in <module>
    insert("Water Glass,10,5")
TypeError: insert() missing 2 required positional arguments: 'quantity' and 'price'
PS D:\pp\database section> python .\script1.py
PS D:\pp\database section> python .\script1.py
```

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

- **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- **wxPython** − This is an open-source Python interface for wxWindows http://wxpython.org.
- **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local

There are many other interfaces available, which you can find them on the net.

Tkinter Programming:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

- Import the *Tkinter* module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Example:

```
#!/usr/bin/python

import Tkinter
top = Tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

Standard attributes:

Let us take a look at how some of their common attributes.such as sizes, colors and fonts are specified.

- Dimensions
- Colors
- Fonts
- Anchors
- Relief styles
- Bitmaps
- Cursors

Geometry Management

All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area. Tkinter exposes the following geometry manager classes: pack, grid, and place.

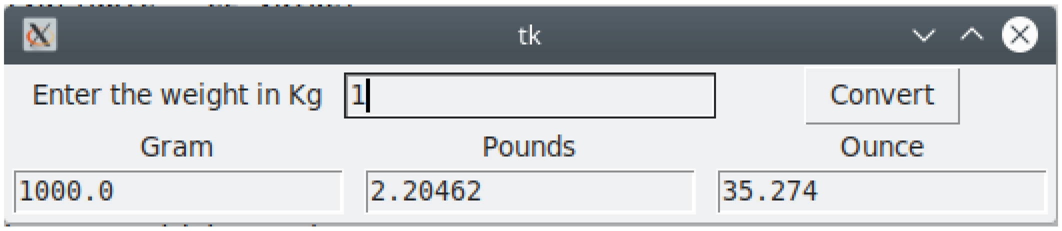
- The *pack()* Method − This geometry manager organizes widgets in blocks before placing them in the parent widget.
- The *grid()* Method − This geometry manager organizes widgets in a table-like structure in the parent widget.
- The *place()* Method − This geometry manager organizes widgets by placing them in a specific position in the parent widget.

Weight Conversion GUI using Tkinter:

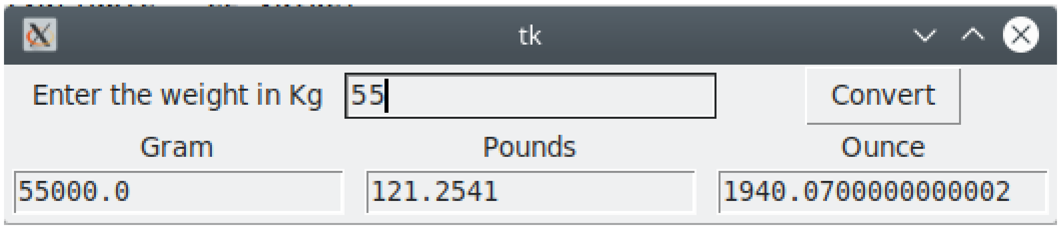
```
# Create a GUI window
window = Tk()

# Function to convert weight
# given in kg to grams, pounds
# and ounces
def from_kg():

    # convert kg to gram
    gram = float(e2_value.get())*1000

    # convert kg to pound
    pound = float(e2_value.get())*2.20462

    # convert kg to ounce
    ounce = float(e2_value.get())*35.274

    # Enters the converted weight to
    # the text widget
    t1.delete("1.0", END)
    t1.insert(END,gram)

    t2.delete("1.0", END)
    t2.insert(END,pound)

    t3.delete("1.0", END)
    t3.insert(END,ounce)

# Create the Label widgets
e1 = Label(window, text = "Enter the weight in Kg")
e2_value = StringVar()
e2 = Entry(window, textvariable = e2_value)
e3 = Label(window, text = 'Gram')
e4 = Label(window, text = 'Pounds')
e5 = Label(window, text = 'Ounce')

# Create the Text Widgets
t1 = Text(window, height = 1, width = 20)
t2 = Text(window, height = 1, width = 20)
t3 = Text(window, height = 1, width = 20)

# Create the Button Widget
b1 = Button(window, text = "Convert", command = from_kg)

# grid method is used for placing
# the widgets at respective positions
# in table like structure
e1.grid(row = 0, column = 0)
e2.grid(row = 0, column = 1)
e3.grid(row = 1, column = 0)
e4.grid(row = 1, column = 1)
e5.grid(row = 1, column = 2)
```

```
t1.grid(row = 2, column = 0)
t2.grid(row = 2, column = 1)
t3.grid(row = 2, column = 2)
b1.grid(row = 0, column = 2)

# Start the GUI
window.mainloop()
```

Output:



```
Enter the weight in Kg  1
Gram        Pounds      Ounce
1000.0      2.20462     35.274
```



```
Enter the weight in Kg  55
Gram        Pounds      Ounce
55000.0     121.2541    1940.0700000000002
```

Database Programming in Python:

From a construction firm to a stock exchange, every organisation depends on large databases. These are essentially collections of tables, and' connected with each other through columns. These database systems support SQL, the Structured Query Language, which is used to create, access and manipulate the data. SQL is used to access data, and also to create and exploit the relationships between the stored data. Additionally, these databases support database normalisation rules for avoiding redundancy of data. The Python programming language has powerful features for database programming. Python supports various databases like MySQL, Oracle, Sybase, PostgreSQL, etc. Python also supports Data Definition Language (DDL), Data Manipulation Language (DML) and Data Query Statements. For database programming, the Python DB API is a widely used module that provides a database application programming interface.

Benefits of Python for database programming:

There are many good reasons to use Python for programming database applications:

- Programming in Python is arguably more efficient and faster compared to other languages.
- Python is famous for its portability.
- It is platform independent.
- Python supports SQL cursors.
- In many programming languages, the application developer needs to take care of the open and closed connections of the database, to avoid further exceptions and errors. In Python, these connections are taken care of.
- Python supports relational database systems.
- Python database APIs are compatible with various databases, so it is very easy to migrate and port database application interfaces.

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as −

- GadFly
- mSQL
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase

Here is the list of available Python database interfaces: Python Database Interfaces and APIs. You must download a separate DB API module for each database you need to access. For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following −

- Importing the API module.
- Acquiring a connection with the database.
- Issuing SQL statements and stored procedures.
- Closing the connection

We would learn all the concepts using MySQL, so let us talk about MySQLdb module.

What is MySQLdb?

MySQLdb is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.

How do I Install MySQLdb?

Before proceeding, you make sure you have MySQLdb installed on your machine. Just type the following in your Python script and execute it −

```
#!/usr/bin/python

import MySQLdb
```

If it produces the following result, then it means MySQLdb module is not installed −

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    import MySQLdb
ImportError: No module named MySQLdb
```

To install MySQLdb module, use the following command −

For Ubuntu, use the following command -

```
$ sudo apt-get install python-pip python-dev libmysqlclient-dev
```

For Fedora, use the following command -

```
$ sudo dnf install python python-devel mysql-devel redhat-rpm-config gcc
```

Database Connection:

Before connecting to a MySQL database, make sure of the followings −

- You have created a database TESTDB.
- You have created a table EMPLOYEE in TESTDB.
- This table has fields FIRST_NAME, LAST_NAME, AGE, SEX and INCOME.
- User ID "testuser" and password "test123" are set to access TESTDB.
- Python module MySQLdb is installed properly on your machine.
- You have gone through MySQL tutorial to understand MySQL Basi