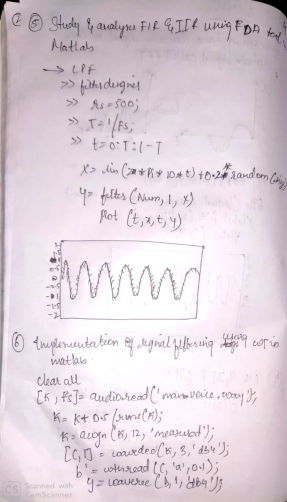
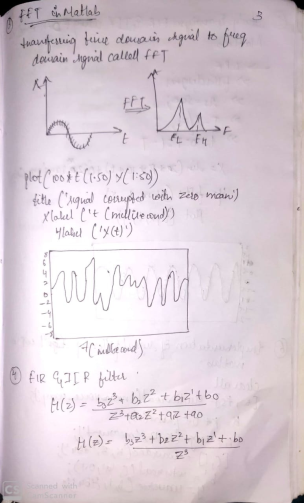
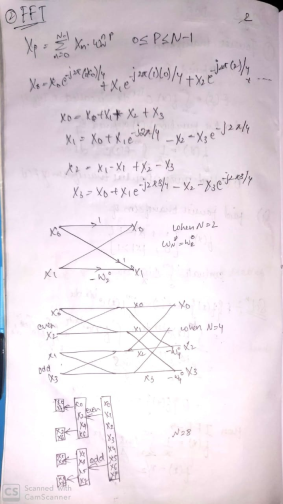
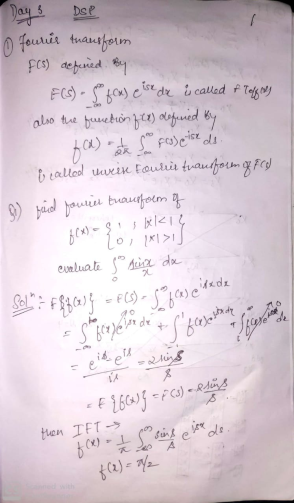
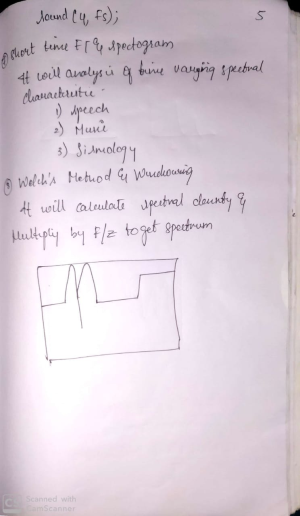
**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **27-May-2020** | **Name:** | **Russell D’souza** |
| **Course:** | **TCS iON** | **USN:** | **4AL15EC023** |
| **Topic:** | **DSP** | **Semester & Section:** | **8th sem & ‘A’ section** |
| **Github Repository:** | **Russell1005** |  |  |

|  |
| --- |
| **MORNING SESSION DETAILS** |
| **Image of session** |

**DAILY ASSESSMENT FORMAT**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | **Date:** | **27-5-2020** | **Name:** | **Russell D’souza** | | **Course:** | **Python programming** | **USN:** | **4AL15EC023** | | **Topic:** | **Graphical user interface with tkinter,Interacting with data base** | **Semester & Section:** | **8th A** | | **Github Repository:** | **Russell1005** |  |  | |
|  |
|  |
| **AFTERNOON SESSION DETAILS** | |
| **Image of session** | |
| 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](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](https://www.tutorialspoint.com/python/tk_dimensions.htm) * [Colors](https://www.tutorialspoint.com/python/tk_colors.htm) * [Fonts](https://www.tutorialspoint.com/python/tk_fonts.htm) * [Anchors](https://www.tutorialspoint.com/python/tk_anchors.htm) * [Relief styles](https://www.tutorialspoint.com/python/tk_relief.htm) * [Bitmaps](https://www.tutorialspoint.com/python/tk_bitmaps.htm) * [Cursors](https://www.tutorialspoint.com/python/tk_cursors.htm)  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](https://www.tutorialspoint.com/python/tk_pack.htm) − This geometry manager organizes widgets in blocks before placing them in the parent widget. * [The *grid()* Method](https://www.tutorialspoint.com/python/tk_grid.htm) − This geometry manager organizes widgets in a table-like structure in the parent widget. * [The *place()* Method](https://www.tutorialspoint.com/python/tk_place.htm) − 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:**  python-tkinter  python-tkinter 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](https://electronicsforu.com/resources/cool-stuff-misc/collection-51-free-ebooks-python-programming) language has powerful features for database programming. [Python](https://opensourceforu.com/2017/06/python-user-friendly-language-coding/) 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](http://wiki.python.org/moin/DatabaseInterfaces). 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](https://www.tutorialspoint.com/mysql/index.htm) | |