INT 213 CA1 [Project Based]

Roll No. B51 Reg No. 11901731

Morse Code Converter

<My Contribution<Raman Sharma>>

GitHub Repository : https://github.com/RamanSharma100/morse-code-converter-python-project

I Contributed in the following portions :-

1. **Signup Module** :- GUI + Database + validation [With Regex]
2. **Text To Morse Module** :- GUI+ Database + validation [with Logic]
3. **Logics** :- For Both **Text To Morse** And **Morse To Text**
4. **Database Designing** :- Database creation + Tables creation [Automatically done by the application if table does not exists in the database]

Modules Working in Details With GUI, Database And Documentation:-

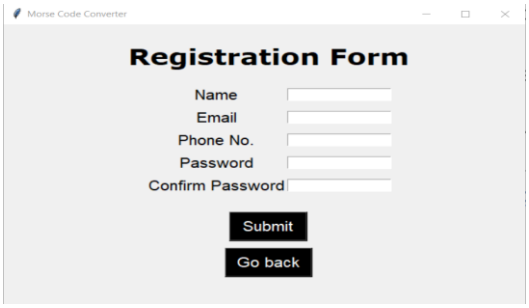
1. **Signup Module :-**

1.1. Introduction:-

- In this module there is a signup form where user can register into our application and that will store the data in the **pythonProject** database **users** table.
- This module will automatically create the database table **users** after getting all fields correct with no error.
- This module has many validations that all are done with the help you **Regex** [validations without database] and have validations that are interacting with the **Database** .

1.2. GUI [Graphic User Interface]:-

This is the GUI of the signup form:-

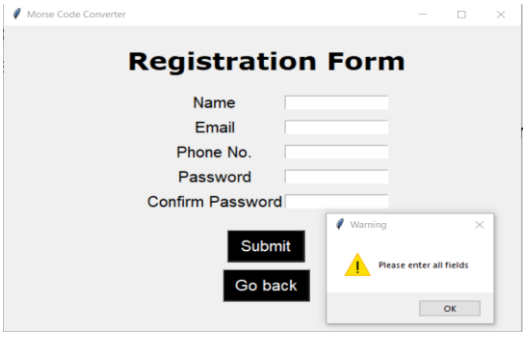


1.3. Validations :-

Here 7 types of validations are applied:-

1. All fields empty:-

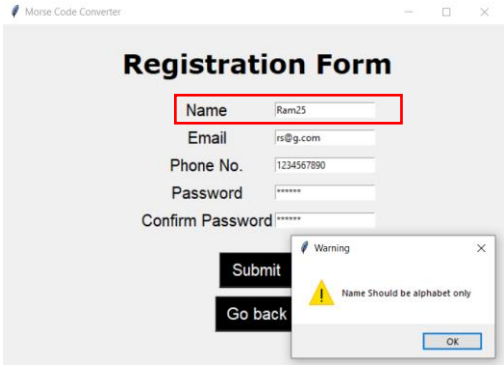
If all fields are empty then this will show the error like this:-



Validations done by Regex:-

2. Enter Valid name:-

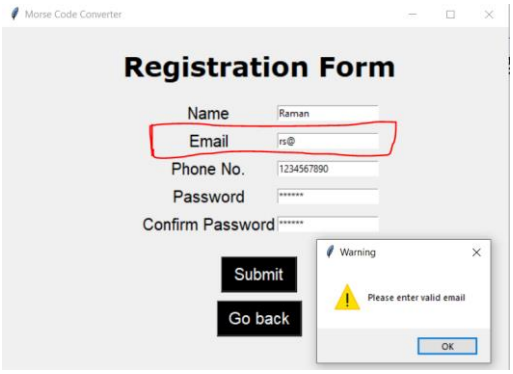
If in the name field user has entered any numerical value then it will show the name validation error



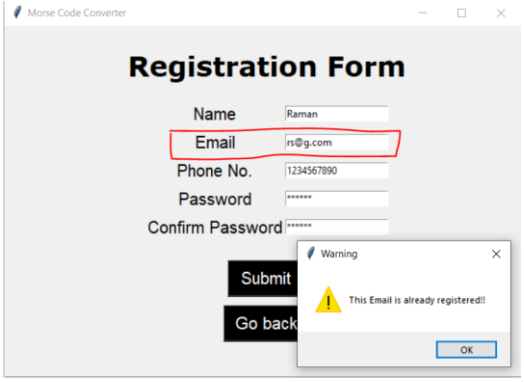
3. Email validation :-

The email validation consist of two parts:-

a. first to check if the email sequence is correct , means is it valid or not, this validation is done by regex

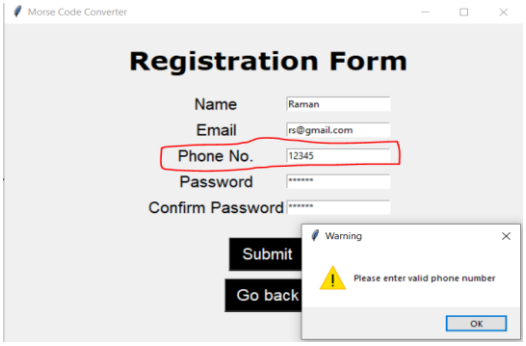


b. second time it will check in the database , if the email is already present or not because email will act like a username for the user



4. Phone number validation:-

Here this is one validation for the phone that if the phone number is less than 10 numbers then it will show the phone number validation error

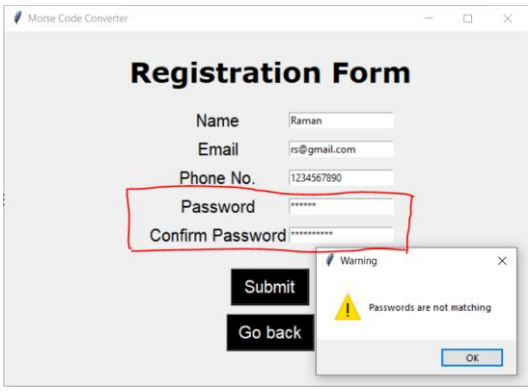


5. Password validation:-

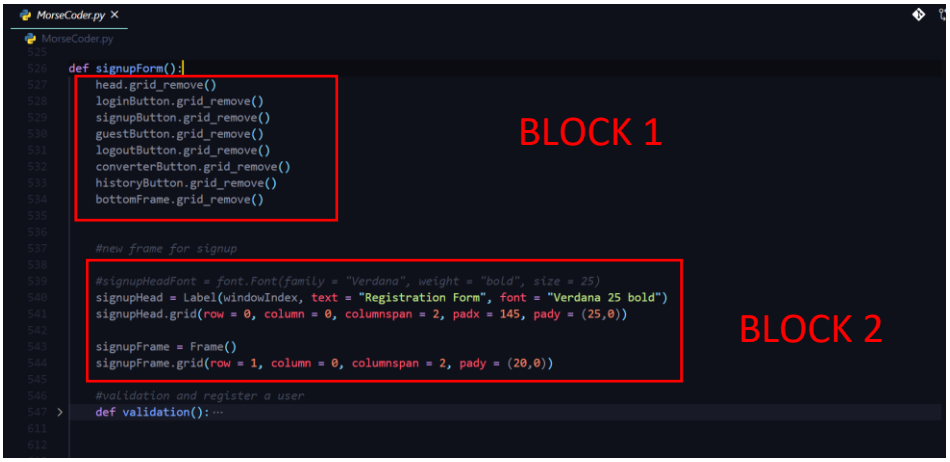
The password validation is applied that it should not be less than 6 characters

6. Password and Confirm Password not matching validation:-

If the password dad confirm password donot matched then the Passwords donot match error will display



1.4. Code Documentation:-



Block 1:-

In this code of snippet there when the user will click on signup button, then signupForm() function will be execute.
Here in the block 1 snippet will remove the previous visible frames and buttons. This code will make our gui empty with no Frame.

Block 2:-

Now, all frames has been removed and empty frame is there so the first two line of codes in block 2 will add a label or heading to the frame , i.e, Registration Form. The last two lines of code in block 2 will add an empty frame named **signupFrame** Below the Heading or Label.

```
MorseCoder.py ×
MorseCoder.py > signupForm
#declaration
name = Label(signupFrame, text = "Name", font = "12")
email = Label(signupFrame, text = "Email", font = "12")
phonenumber = Label(signupFrame, text = "Phone No.", font = "12")
password = Label(signupFrame, text = "Password", font = "12")
confirmpass = Label(signupFrame, text = "Confirm Password", font = "12")

#packing declarations
name.grid(row = 1,column = 0, pady = (0,5))
email.grid(row = 2, column = 0, pady = (0,5))
phonenumber.grid(row = 3, column = 0, pady = (0,5))
password.grid(row = 4, column = 0, pady = (0,5))
confirmpass.grid(row = 5, column = 0, pady = (0,5))

#variables
namevalue = StringVar()
emailvalue = StringVar()
phonenumbervalue = StringVar()
passwordvalue = StringVar()
confirmpassvalue = StringVar()

#Entries
nameentry = Entry(signupFrame, textvariable = namevalue)
emailentry = Entry(signupFrame, textvariable = emailvalue)
phonenumberentry = Entry(signupFrame, textvariable = phonenumbervalue)
passwordentry = Entry(signupFrame,show = "*", textvariable = passwordvalue)
confirmpassentry = Entry(signupFrame, show = "*", textvariable = confirmpassvalue)
```

BLOCK 3

BLOCK 4

BLOCK 5

BLOCK 6

```
MorseCoder.py ×
MorseCoder.py > signupForm
confirmpassentry = Entry(signupFrame, show = "*", textvariable = confirmpassvalue)

#packing entries
nameentry.grid(row = 1, column = 1, pady = (0,5))
emailentry.grid(row = 2, column = 1, pady = (0,5))
phonenumberentry.grid(row = 3, column = 1, pady = (0,5))
passwordentry.grid(row = 4, column = 1, pady = (0,5))
confirmpassentry.grid(row = 5, column = 1, pady = (0,5))

#     f.write(f"{namevalue.get(), phonenumbervalue.get(), emailvalue.get(), passwordvalue.get(), termsvalue.get()}\n")

def goBack():
    signupHead.grid_remove()
    signupFrame.grid_remove()
    head.grid()
    loginButton.grid()
    signupButton.grid()
    guestButton.grid()
    bottomFrame.grid()

#buttonFont = font.Font(size=10)
submit = Button(signupFrame, text = "Submit", font = "10", bg = '#000000', fg = '#fff', command = validation)
submit.grid(row = 6, column = 0, columnspan = 2, ipadx = 8, ipady = 2, pady = (20,0))
goBack = Button(signupFrame, text = "Go back", font = "10", bg = '#000000', fg = '#fff', command = goBack)
goBack.grid(row = 7, column = 0, columnspan = 2, ipadx = 6, ipady = 2, pady = (10,0))

# logout the user
```

BLOCK 7

BLOCK 8

BLOCK 9

Block 3:-

The code snippets in block 3 are used to create the labels for the form entries and that labels are stored in the respective variables.

Block 4 :-

In the block 4 code snippet the labels that we created in block 3 are placed in the grid in 5 rows respectively and in first column.

Block 5 :-

In the block 5 some textvariables are created for the entries of the labels that we created and places in block 3 and 4 to get the data of the entries that are created in the block 6 for our validation and database purpose.

Block 6 :-

Here in this block 6 we are creating some entries for out labels in which users will enter the values by seeing the labels text as shown in GUI block above and stored in respective variables to place in the frame.

Block 7 :-

In the block 7 the entries that we are created in block 6 are placed parallel to the labels that we place in the block 4 in 5 rows but in second column.

Block 8 :-

This block 8 snippet code is the go back button which is used to go back the user to the home screen.

Here this will first remove all this frame and then it will place the home frame and place its buttons.

Block 9 :-

This code snippets in the block 9 are the signup button and go back button.

This signup button will call the validation function that we will see in next block.

The goback button will call the goback function that we saw in previous block.

```
MorseCoder.py
def validation():
    errors = 0
    if(namevalue.get() == '' or emailvalue.get() == '' or phonenumbervalue.get() == '' or passwordvalue.get() == '' or
    confirmpassvalue.get() == ''):
        messagebox.showwarning("Warning", "Please enter all fields")
        errors = errors + 1
    else:
        #validating name field
        if(not re.search("^[a-zA-Z].*",namevalue.get()) or re.search("[0-9]",namevalue.get())):
            messagebox.showwarning("Warning", "Name Should be alphabet only")
            errors = errors + 1
        #validating email
        if(not re.search("^[a-z0-9]+[\._]?[a-z0-9]+[@]\w+[.]\w{2,3}$",emailvalue.get().lower())):
            messagebox.showwarning("Warning", "Please enter valid email")
            errors = errors + 1
        #validating phone number
        if(not re.search("^[0-9].*$",phonenumbervalue.get()) or len(phonenumbervalue.get()) < 10):
            messagebox.showwarning("Warning", "Please enter valid phone number")
            errors = errors + 1
        #validating password
        if( passwordvalue.get() == '' or len(passwordvalue.get()) < 6 ):
            messagebox.showwarning("Warning", "Password should be alteast of 6 characters")
            errors = errors + 1
        #confirming password
        if(passwordvalue.get() != confirmpassvalue.get()):
            messagebox.showwarning("Warning", "Passwords are not matching")
            errors = errors + 1
```

Here in the validation function and in this code of snippet :-

First, There errors variable is declared and it is initialized as 0 means it is having no error.

Now there is first if condition, which will check the first validation that I mentioned in the validations part, i.e., if any of the one value is empty then show the warning that please enter all the fields and increment the error by 1.

Now, if there is no error then it will go to the else condition and now it will validate with the help of regex respectively as I mentioned in the validations part and with each error it will increment the errors value.

```
#checking if errors are there or not
if(errors == 0):
    # if there is no error connect to the database
    db = mysql.connector.connect(host='localhost',user='root',password='Krantivir456@',database='pythonProject')
    # checking if database is connected or not
    if(db):
        # creating database cursor
        myCursor = db.cursor()
        # creating database first if it is not created
        myCursor.execute("CREATE TABLE IF NOT EXISTS users(userId int(255) PRIMARY KEY AUTO_INCREMENT,name varchar
        (255) NOT NULL,email TINYTEXT NOT NULL,phonenumber int(11) NOT NULL,pass varchar(255) NOT NULL)")
        # checking if the email is already present or not
        emailCheckSql = r"SELECT * FROM users WHERE email='"+emailvalue.get()+"'"
        myCursor.execute(emailCheckSql)
        myCursor.fetchall()
        if(myCursor.rowcount < 1):...
        else:
            myCursor.close()
            db.close()
            messagebox.showwarning("Warning","This Email is already registered!!")
    else:
        messagebox.showwarning("Warning","Database is not connected!")
```

Now the next first condition after validations will check if there are no errors the run this code otherwise it will not run this code.
Now, if there are no errors we will connect the database **pythonProject** with the help of **mysql.connector** package and by importing it with database name and my sql credentials on localhost.
Now, we will check the condition if database is connected or not, if it is not connected the we will go to else condition and generate the error that database is not connected.
Otherwise, we will first create a cursor of the database and will store it to **myCursor** variable
And the we will execute one command to create a table if it does not exists in our database.
If this table exists in our database this statement will not execute.
We all are executing these commands with the help of the cursor.
The we will check one more validation that is email validation as I mentioned previously in validations section now we will execute one statement and we will get the value of the email entry with the help of text variable and will execute one command to check that if that email is already present in the table or not.
Now after execution we will count the rows and if it is > 0 then we will close our database and cursor and will generate one error that this email is already present in the database as email will act as username here for login.

```
if(myCursor.rowcount < 1):
    # here email is not present then inserting the data and register the user
    insertionSql = "INSERT INTO users(email,phonenumber,name,pass) VALUES(%s,%s,%s,%s)"
    myCursor.execute(insertionSql,(emailvalue.get(), phonenumbervalue.get(),namevalue.get(),passwordvalue.get
    ()))
    db.commit()
    if(myCursor.rowcount == 1):
        myCursor.close()
        db.close()
        signupHead.grid_remove()
        signupFrame.grid_remove()
        head.grid()
        loginButton.grid()
        signupButton.grid()
        guestButton.grid()
        bottomFrame.grid()
        messagebox.showinfo("Success", "You are registered successfully!!")
    else:
        messagebox.showwarning("Warning","Something went wrong while inserting the data!!")
else:
    myCursor.close()
    db.close()
    messagebox.showwarning("Warning","This Email is already registered!!")
```

Now if we don't found the email there then we will take values from the text variables and execute insertion command with mysql prepare statement and then insert the data in the table and if the data is inserted then we will close the database and cursor and then will go send user back to the home screen with the success message of **you are registered successfully**.

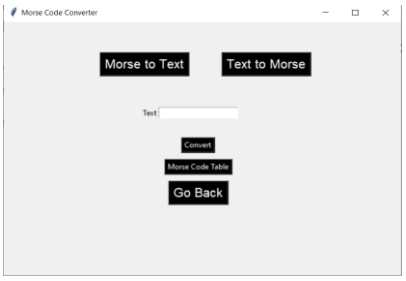
2. Text to morse module :-

2.1. Introduction:-

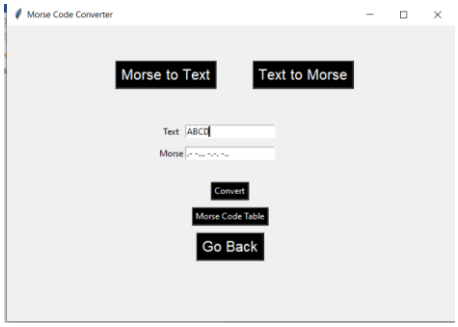
- In this module we are converting the text to the morse using dictionary.
- If the user is not logined then we will give info to the user if he/she want to save the history of the convertion then user have to login first.
- If the user is logined then we will save the history of the user with its **userid** that is already saved in the **loginedUserData** variable after login of the user in **texttomorsehistory** table in the database.

2.2. GUI [Graphic User Interface]:-

This is the GUI of the text to morse code convertion:-



After conversion:-

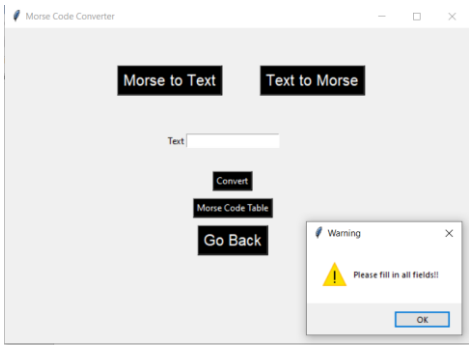


2.3. Validations :-

Here in this module only 2 types of validations are applied:-

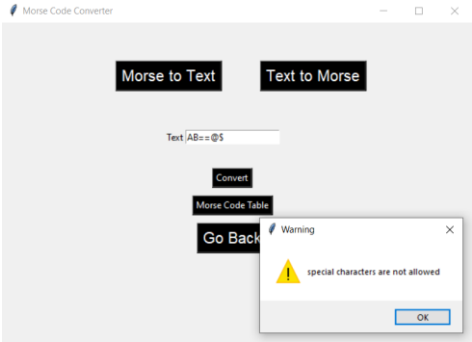
1. Empty field validation:-

   In this validation if the field is empty then it will give error please fill in all fields as :-



2. Special characters not allowed validation:-

   If the user by mistake will add special character then it will return error as:-



2.4. Code Documentation [without logic]:-

This coding documentation of this module is without logic explanation.
The logic explanation will be provided in **next module** with **morse to text** logic

```
MorseCoder.py
MorseCoder.py > English_to_Morse

def English_to_Morse():

    def convertion():...

    # def()
    global tapetom
    tapetom = True
    global countEtoM
    countEtoM += 1

    if(tapmtoe == True):
        global countMtoE
        mtoeFrame.grid_remove()
        #updating countMtoE and making it zero to start over when morse_to_english was call tap-morse-to-english(tapmtoe)
        becomes true and this condition execute
        countMtoE = 0
    #fix end

    global etomFrame

    if(countEtoM == 1):
        etomFrame = Frame(windowIndex)
        etomFrame.grid(row = 1, column = 0, columnspan = 2)

        label2 = Label(etomFrame, text = "Text")
        label2.grid(row = 1, column = 0, pady = (50,0))

        label3 = Label(etomFrame, text = "Morse")
```

BLOCK 1

BLOCK 2

Block 1 :-

The block 1 is explained in next module with morse to text conversion also.

Block 2 :-

Here in this block 2 three global variables are there these three variables are used to control visibility of the text to morse conversion frame.
This if condition is used to fix the problem that if user will click two times on the text to morse button then only one time this frame will be created.

```
MorseCoder.py
MorseCoder.py > English_to_Morse

    if(countEtoM == 1):
        etomFrame = Frame(windowIndex)
        etomFrame.grid(row = 1, column = 0, columnspan = 2)

        label2 = Label(etomFrame, text = "Text")
        label2.grid(row = 1, column = 0, pady = (50,0))

        label3 = Label(etomFrame, text = "Morse")
        label3.grid(row = 2, column = 0, pady = (10,0))
        label3.grid_remove()

        entry = StringVar()
        get = StringVar()

        value = Entry(etomFrame, textvariable = entry )
        value.grid(row = 1, column = 1, pady = (50,0))

        getvalue = Entry(etomFrame, textvariable=get)#####################################
        getvalue.grid(row = 2, column = 1, pady = (10,0))
        getvalue.grid_remove()

        Convert = Button(etomFrame, text = "Convert", bg = '#000000', fg = '#fff', command = convertion)
        Convert.grid( row  =  3, column = 1, pady = (30,0))
```

In this code of snippet in the if condition first there is checked if this frame is clicked or not.

Then if the button is clicked to go to this frame so its count is 1 and first we are creating the frames and gridding it in our application window.
There there label is created that is the heading named Text and gird it in that frame.
Then i created a label for Morse code after conversion but it is not visible yet.
When text will be converted into the morse code then this morse label and entry field will be visible, that all I will tell in the next module in the logics portion, that how it is working.
Then we created the text variables for entry to get the text from the user and to set the text in the text box after conversion so that user can copy that converted code.
Then we grid the text entry box but we removed the morse code entry box after griding it to avid the error after griding it again after conversion.
Then we created the convert button that will go to conversion function that we saw in block 1 and that is explained in next module.

3. Logics :-

3.1. Text To Morse:-

```
MorseCoder.py
MorseCoder.py > English_to_Morse

def convertion():
    if(entry.get() == ''):
        messagebox.showwarning("Warning","Please fill in all fields!!")
    else:
        data = {···
        }
        error = 0
        text = ''
        for i in entry.get():
            if i == ' ':
                text = text + i
            else:
                try:
                    text = text + data[i.capitalize()] + ' '
                except KeyError:
                    messagebox.showwarning("Warning","special characters are not allowed")
                    error = 1
                    break
        if error == 0:
            label3.grid()
            getvalue.grid()
            get.set(text)
            if registered == True:
                #connect database first
                db = mysql.connector.connect(host='localhost',user='root',password='Krantivir456@',database='pythonProject')
                #checking if database is connected or not
```

- Here in this code of snippet when the user will click on the convert button then it will go to conversion function.
- First, there is an if condition it will check if the text field is empty or not, if it is empty the in will show the warning message through the popup that please fill the field.
- Now, if there is no error then there is one variable named **data** that contains a dictionary with key as a letter and value as a morse code. The demo is given below:-

```
data = {
    "A" : ".-",
    "B" : "-...",
    "C" : "-.-.",
    "D" : "-..",
    "E" : ".",
    "F" : "..-.",
    "G" : "--.",
    "H" : "....",
    "I" : "..",
    "J" : ".---",
    "K" : "-.-",
    "L" : ".-..",
    "M" : "--",
    "N" : "-.",
    "O":"---",
    "P":".--.",
```

This value is upto Z and containing 0-9 numbers also

- After that there is one variable named **errors** which will store the error and then there are some validations as mentioned above.
- Now, the logic comes
- Now for loop is there which run till its text length and run character by character.
- Then there is an if condition to check the space, if space is there then concatinate it to the **text** variable else the condition will false.
- If condition is false then it will use **try except** first it will try to get that character with the help of key with the help of character and if the value founded then it will concatinate the morse code of that character into **text** variable.
- If the key is not found in that **data** dict. Then it will go to except and will raise the error and give the warning to the user as popup and **errors** will be incremented by 1 and will break the loop.
- If there is no error then, it will **show the morse textbox** and **will set its value that is in the text variable** .

```
MorseCoder.py ×
MorseCoder.py > English_to_Morse > convertion
            if registered == True:
                #connect databse first
                db = mysql.connector.connect(host='localhost',user='root',password='Krantivir456@',database='pythonProject')
                #checking if databse is connected or not
                if(db):
                    myCursor = db.cursor()
                    #create table if it not exists
                    tableSql = r"CREATE TABLE IF NOT EXISTS texttomorsehistory( id int(255) PRIMARY KEY AUTO_INCREMENT, text
                    varchar(255) NOT NULL, morseCode varchar(255) NOT NULL, userid int(255) NOT NULL)"
                    myCursor.execute(tableSql)

                    #inserting data to texttomorsehistory table to store the history
                    storeHistorySql = "INSERT INTO texttomorsehistory(text, morseCode, userid) VALUES(%s,%s,%s)"
                    myCursor.execute(storeHistorySql, (entry.get(),text,loginedUserData[0]))
                    db.commit()
                    resultExecution = myCursor.rowcount
                    if(resultExecution == 1):
                        myCursor.close()
                        db.close()
                        messagebox.showinfo("Success","Conversion completed!!")
                    else:
                        myCursor.close()
                        db.close()
                        messagebox.showwarning("Success","Something went wrong while adding history!!")
                else:
                    messagebox.showwarning("Warning","Database is not connected!!")
            else:
                messagebox.showinfo("Info","If you want to save this in your history please try convertion again after
                loging in to this application!!")
```

- Now, there is one if condition to check that if user is registered or not.
- If user is not registered we will simply generate the info to the user that **if you want to save the history then you have to login into the application**.
- If the user is registered then we will connect to the database with the database name and its credentials and will check if the database is connected or not.
- If the database is connected will be proceed further otherwise we will return warning that database is not connected.
- If database is connected will create cursor and will create table **texttomorse** in database if it is not present and then we will insert the value in it and will give success message to the user.

3.2. Morse To Text [Conversion Logic only]:-

```
        else:
            data = {···
            }
            error = 0
            text = ''
            for i in entry.get().split():
                try:
                    text = text + list(data.keys())[list(data.values()).index(i)]
                except ValueError:
                    messagebox.showwarning("Warning","Please enter the valid morser code")
                    error = 1
                    break
```

- Now after all the validations, now my logic comes.
- Here the **data** variable is same as text to morse
- Here **error** variable is there to check the error and **text** variable is there to store the text
- Now, since the more code entered by the user with the space after each character so I am splitting it and then running for loop.
- Now, under the for loop it will try **text=text+list(data.keys())[list(data.values()).index(i)]**
- This code is first creating the list of keys from that data variable and that list of keys index will be same as index of values so I am creating list of its values ad values contain morse code and variable i also contain the morse code so it will find the index of that morse code from the values of the dictionary.
- Then it have index and as the index of value and keys are same so it will decode it and will concatenate it to the text.