```
'''
-->Please first run the the 'audio_generator' file in the audio folder if the audio
files are not created
-->Please make sure that you have installed the following modules before moving ahead
with the program
1) Tkinter - 'pip install tk'
2) Mysql connector - 'pip install mysql-connector-python'
3) Pillow (referenced as PIL=Python Image Library) - 'pip install Pillow'
4) Pygame - 'pip install pygame'
5) Keyboard - 'pip install keyboard'
-->These commands are to be executed at command prompt on windows
'''

import mysql.connector as Connector # Python MySQL Connector
from tkinter import *                 # Main tinker module
from tkinter.tix import *             # Module for tool tip
from PIL import ImageTk, Image        # Module for image manipulation
from tkinter.font import Font         # Module for font
from tkinter import messagebox        # Module for pop-up message box
from tkinter import ttk               # Module for Combo-box (Drop-down Box)
import pygame                         # Module for sound effects
import keyboard                       # Module to identify keystrokes


'''
Note: Please make sure before executing the program that
      the database and table with following specifications
      is already constructed on your PC.

Database name: username_pass
Table name: data
Table format: column1 = Username and column2 = Password
+----------+----------+
| Username | Password |
+----------+----------+
|   Rohit  |  abcdef  |
+----------+----------+
|  Sonia12 | son1234  |
+----------+----------+
'''

# Dictionary of elements in {"Atomic number" : "Name"} format
# Elements are referenced through out the program through this
# dictionary only.
elemDict = {
    "1": "HYDROGEN",
    "2": "HELIUM",
    "3": "LITHIUM",
    "4": "BERYLLIUM",
    "5": "BORON",
    "6": "CARBON",
    "7": "NITROGEN",
    "8": "OXYGEN",
    "9": "FLUORINE",
    "10": "NEON",
    "11": "SODIUM",
    "12": "MAGNESIUM",
    "13": "ALUMINIUM",
    "14": "SILICON",
    "15": "PHOSPHORUS",
    "16": "SULPHUR",
    "17": "CHLORINE",
    "18": "ARGON",
    "19": "POTASSIUM",
    "20": "CALCIUM",
    "21": "SCANDIUM",
    "22": "TITANIUM",
    "23": "VANADIUM",
    "24": "CHROMIUM",
    "25": "MANGANESE",
    "26": "IRON",
    "27": "COBALT",
    "28": "NICKEL",
    "29": "COPPER",
    "30": "ZINC",
    "31": "GALLIUM",
    "32": "GERMANIUM",
    "33": "ARSENIC",
    "34": "SELENIUM",
    "35": "BROMINE",
    "36": "KRYPTON",
    "37": "RUBIDIUM",
    "38": "STRONTIUM",
    "39": "YTTRIUM",
    "40": "ZIRCONIUM",
    "41": "NIOBIUM",
    "42": "MOLYBDENUM",
    "43": "TECHNETIUM",
    "44": "RUTHENIUM",
    "45": "RHODIUM",
    "46": "PALLADIUM",
    "47": "SILVER",
    "48": "CADMIUM",
    "49": "INDIUM",
    "50": "TIN",
    "51": "ANTIMONY",
    "52": "TELLURIUM",
    "53": "IODINE",
    "54": "XENON",
    "55": "CAESIUM",
    "56": "BARIUM",
    "57": "LANTHANUM",
    "58": "CERIUM",
    "59": "PRASEODYMIUM",
    "60": "NEODYMIUM",
    "61": "PROMETHIUM",
    "62": "SAMARIUM",
    "63": "EUROPIUM",
    "64": "GADOLINIUM",
    "65": "TERBIUM",
    "66": "DYSPROSIUM",
    "67": "HOLMIUM",
    "68": "ERBIUM",
    "69": "THULLIUM",
    "70": "YTTERBIUM",
    "71": "LUTETIUM",
    "72": "HAFNIUM",
    "73": "TANTALUM",
    "74": "TUNGSTEN",
    "75": "RHENIUM",
    "76": "OSMIUM",
    "77": "IRIDIUM",
    "78": "PLATINUM",
    "79": "GOLD",
    "80": "MERCURY",
    "81": "THALIUM",
    "82": "LEAD",
    "83": "BISMUTH",
    "84": "POLONIUM",
    "85": "ASTATINE",
    "86": "RADON",
    "87": "FRANCIUM",
    "88": "RADIUM",
    "89": "ACTINIUM",
    "90": "THORIUM",
    "91": "PROTACTINIUM",
    "92": "URANIUM",
    "93": "NEPTUNIUM",
```

```python
    "94": "PLUTONIUM",
    "95": "AMERICIUM",
    "96": "CURIUM",
    "97": "BERKELIUM",
    "98": "CALIFORNIUM",
    "99": "EINSTEINIUM",
    "100": "FERMIUM",
    "101": "MENDELEVIUM",
    "102": "NOBELIUM",
    "103": "LAWRENCIUM",
    "104": "RUTHERFORDIUM",
    "105": "DUBNIUM",
    "106": "SEABORGIUM",
    "107": "BOHRIUM",
    "108": "HASSIUM",
    "109": "MEITNERIUM",
    "110": "DARMSTADTIUM",
    "111": "ROENTGENIUM",
    "112": "COPERNICIUM",
    "113": "NIHONIUM",
    "114": "FLEROVIUM",
    "115": "MOSCOVIUM",
    "116": "LIVERMORIUM",
    "117": "TENNESSINE",
    "118": "OGANESSON"
}

def main():

    '''
    This is the main function that contains complete program.
    The whole program has been included in a function so as
    to be able to use it anywhere in the program whenever we need
    to launch the program from start e.g. in case when the user
    wish to change the account then we need to launch the signin
    window again, at that time we can simply call the main()
    function which will automatically launch the program from start.
    '''

    # Signin-Window
    global bg                                              # Global variable to hold bg-img

    window = Tk()                                          # Creating window instance
    window.resizable(False, False)                         # Window is not resizable
    window.iconbitmap('921815.ico')                        # Setting icon of the window
    window.title('Sign In Window')                         # Setting title of the window
    window.geometry('700x350+350+180')                     # Setting window size
    bg = ImageTk.PhotoImage(file='mainbg.jpg')             # Assigning bg-img to the "bg" variable
    '''
    NOTE: Here we are using canvas for holding background image
          so that we can assemble all the different labels onto
          the background image which is only possible in canvas.
    '''
    signin_canvas = Canvas(window, width=700, height=350)   # Creating canvas to put in the main window
    signin_canvas.pack(fill='both',expand=True)             # Packing the canvas in the window
    signin_canvas.create_image(0,0, image=bg, anchor='nw')  # "anchor" specifies the position of top left corner of the image
                                                            # "nw" stands for north-west i.e. top left of the window


    # Function to Create new Account (Sign Up)
    def signup():
        '''
        This function defines the signup window and contains
        different functions to do the following tasks:
        1) Take new account's details like username and password
           and store in the database named "username_pass"
           # NOTE: The database "username_pass" is already created
                   and is not being created in the program
        2) Check the strength of the password entered
        '''

        # Function to check password strength
        def password_strength(password):

            '''
            This function checks the strength of the password entered
            The password must:
            1) Contain atleast 8 characters
            2) Contain both letters and numbers

            NOTE: This function returns a statement specifing the condition
                  that is not met by the password ented by the user. This
                  statement will be directly used to display the message
                  on the screen.
            '''

            # Min length of password = 8
            if len(password)>=8:
                # Password should contain both alphabets and digts and symbols
                if not(password.isalpha()):  # Password should not contain only alphabets
                    if not(password.isdigit()):  # Password should not contain only digits
                        return 'strong'  # Password satisfies all parameters
                    else:
                        return 'Password must contain alphabets,\n digits and special symbols'
                else:
                    return 'Password must contain alphabets,\n digits and special symbols'
            else:  # Password length less the 8 characters
                return 'Password must contain \n atleast 8 characters'


        def new_acc():

            '''
            This function when called takes the information entered in username
            and password entry widget of the signup window checks their validity
            and if valid stored it in the database, displaying message to the user
            that the account has been created successfully otherwise due to some
            invalidity it shows a message to the user specifing the problem.
            '''

            new_username = str(username_enter.get()).strip()  # Get the username entered and remove leading and trailing whitespaces
            new_password = str(password_enter.get()).strip() # Get the password entered and remove leading and trailing whitespaces
            re_entered_pass = str(re_enter_textbox.get()).strip()  # Get the re-entered password
            try:
                # If length of the username entered is 0, i.e. Username not entered
                if len(new_username) == 0:
                    messagebox.showinfo('Information',"Username can't be empty!")
                # If length of the password entered is 0, i.e. Password not entered
                elif len(new_password) == 0:
                    messagebox.showinfo('Information',"Password can't be empty!")
                # If re-entered password don't match the previous one
                elif new_password != re_entered_pass:
                    messagebox.showinfo('Information',"Re-entered password doesn't match")
                # If all information entered correctly
                else:
                    if password_strength(new_password) == 'strong':  # Checking strength of the password entered using the "password_strength function defined above
                        # Establishing connection with Mysql
                        con = Connector.connect(host='localhost',user='root',passwd='root',database='username_pass')
                        # Creating cursor instance
                        cursor = con.cursor()
                        # Query to insert new accounts data
                        query = "INSERT INTO data (Username, Password) VALUES (%s, %s)"
                        cursor.execute(query,(new_username, new_password))
                        messagebox.showinfo('Successful','Account created Successfully')
                        # Saving data permanently
                        cursor.execute('commit')
                        # Closing connection
                        con.close()
                        # Destroying the previous Signup Window once the Sign Up is completed successfully
                        signup_window.destroy()
                        # Running the main program once the new account is successfully created
                        # It will again open the signin window
                        main()
                    else:
                        messagebox.showinfo('Weak Password\n', password_strength(new_password))
            except:
                messagebox.showerror('Already Exists','Username already Exists')

        # Window for sign up
        global back                                                     # Global variable to hold bg-img

        window.destroy()                                                # Destroying the signin window once
        signup_window = Tk()                                            # Creating new window instance for signup window
        signup_window.resizable(False, False)                           # Window not resizable
        signup_window.iconbitmap('921815.ico')                          # Setting window icon
        signup_window.title('Sign Up Window')                           # Setting window title
        signup_window.geometry('700x350+350+180')                       # Setting window dimensions

        back = ImageTk.PhotoImage(file='mainbg.jpg')                    # Assigning the bg-img to the "back" variable
        '''
        NOTE: Here we are using canvas for holding background image
              so that we can assemble all the different labels onto
              the background image which is only possible in canvas.
        '''
        signup_canvas = Canvas(signup_window, width=700, height=350)    # Creating canvas to hold bg-img
        signup_canvas.pack(fill='both',expand=True)                     # Packing the canvas into the window
        signup_canvas.create_image(0,0, image=back, anchor='nw')        # Inserting image into canvas

        # Label displaying "Sign Up Here" in bold heading
        signup_canvas.create_text(125, 70, text='Sign Up Here', font=('Impact',30,'bold'),fill='#285243')

        # Label displaying "Username:"
        signup_canvas.create_text(85, 110, text='Username:',
```

```python
        font=('Helvetica',15,'bold'),fill='black')

        # Username entry widget for user to enter username
        username_enter = Entry(signup_window, width = 40)
        # Putting the entry widget up on canvas
        username_enter_window = signup_canvas.create_window(155, 135, window = username_enter)

        # Label displaying "Password:"
        signup_canvas.create_text(85, 175, text='Password:', font=('Helvetica',15,'bold'),fill='black')

        # Password entry widget for user to enter password
        password_enter = Entry(signup_window, width = 40, show='*')
        # Putting the entry widget up on canvas
        password_enter_window = signup_canvas.create_window(155, 200, window = password_enter)

        # Label displaying "Re-enter Password:"
        signup_canvas.create_text(125, 240, text='Re-enter Password:', font=('Helvetica',15,'bold'),fill='black')

        # Entry widget for user to enter password again
        re_enter_textbox = Entry(signup_window, width = 40, show='*')
        # Putting entry widget up on canvas
        re_enter_textbox_window = signup_canvas.create_window(155, 265, window = re_enter_textbox)

        # Button to submit info
        signup_button = Button(signup_window, text='SIGN UP',command=new_acc,width=30,font=('Helvetica',10,'bold'),bg='#cccccc')
        signup_button_window = signup_canvas.create_window(155, 315, window = signup_button)

        # Tooltip for signup button displaying "Create Account"
        tip = Balloon(signup_window)
        tip.bind_widget(signup_button, balloonmsg = 'Create Account')

        signup_window.mainloop()


    # Function for sign in window
    def signin():

        '''
        This function is executed when user press the signin button
        on the sign in window. It extracts out all the information
        entered by the user in the entry box, extracts all the data
        from the data and checks the validity of the information
        entered; if correct then it launches the splash window
        otherwise shows a messagebox to the user displaying the error
        occured.
        '''

        # Establishing connection with database
        con = Connector.connect(host='localhost',user='root',passwd='root',database='username_pass')
        # Creating cursor instance to interact with the database
        cursor = con.cursor()
        # Getting the username and password entered by the user
        username_entered = str(username_entry_widget.get())
        password_entered = str(password_entry_widget.get())
        # Extracting all data of usernames and passwords from DB
        cursor.execute('select * from data')
        # Fetching all data from cursor instance
        all_data = cursor.fetchall()
        # Dictionary to hold username: password values stored
        usernames_passwords_dict = {}
        for i in all_data:  # Iterating through the extracted data which is a list of tuples
            usernames_passwords_dict[i[0]]=i[1]  # Format is: {username : password} because in the tuple the first element
                                                 # will be username and the second one will be the password (see at the
                                                 # starting of the program for reference
        # Checking if the entered username exists
        if username_entered in usernames_passwords_dict:  # If exists
            if password_entered == usernames_passwords_dict.get(username_entered):  # Password check
                splash_window()  # Login successfully
            else:  # Wrong password entered
                messagebox.showerror('Incorrect Password','INCORRECT PASSWORD')
        else:  # Username 'Does Not Exist'
            messagebox.showerror('Username DNE','Username Does Not Exist \n Retry or create a new account')
        # Closing Connection
        con.close()

    def splash_window():

        '''
        This window will pop-up once the login is successful.
        It will show welcome message and has nothing to do with
        the actual loding of program, it is there just to make
        program more realistic and interactive, it will automatically
        execute login_successful function after 1.5 seconds which in
        turn will close this splash window.
        '''

        global background                                   # Global variable to hold background image
        window.destroy()                                    # Destroys the previous signin window
        splash = Tk()                                       # Creating new window instance
        splash.resizable(False, False)                      # Window not resizable
        splash.title('Loading...')                          # Setting title of the window to 'Loading...'
        splash.iconbitmap('921815.ico')                     # Setting icon of the window
        splash.geometry('500x170+400+250')                  # Setting screen size

        background = ImageTk.PhotoImage(file='load.jpg')  # Loading bg-img in the variable
        splash_canvas = Canvas(splash, width=500, height=170)  # Creating canvas
        splash_canvas.pack(fill='both',expand=True)  # Packing canvas in the window
        splash_canvas.create_image(0,0, image=background, anchor='nw')  # Pushing image into the canvas

        # Creating text labels to be displayed
        splash_canvas.create_text(250, 60, text='Welcome!', font=('Helvetica',30,'bold'), fill='white')
        splash_canvas.create_text(250, 100, text='Loading...', font=('Helvetica',15,'bold'), fill='white')

        # Creating sound effects
        pygame.mixer.init()
        pygame.mixer.music.load('audio/welcome.mp3')
        pygame.mixer.music.play(loops=0)
        # Automatically executing login_successful function after 1500 miliseconds i.e. 1.5 seconds
        # Using lambda function because we need to pass an argument -splash window itself- into the function
        splash.after(3000, lambda:login_successful(splash))

    def login_successful(splash):

        '''
```

```python
    This is the function containing whole periodic table and its
    functionalities, it takes the splash window as it's argument
    and first destroys it. Then it creates a new window for
    periodic table and put all the contents in it.
    '''

    splash.destroy()                                         # Destroying the previous splash window
    root = Tk()                                              # Creating main window
    screen_width = root.winfo_screenwidth()                  # Getting screen width of PC
    screen_height = root.winfo_screenheight()                # Getting screen height of PC
    root.iconbitmap('921815.ico')                            # Setting icon
    root.title("Periodic Table")                             # Setting title
    root.geometry(f'{screen_width}x{screen_height}+-10+0')  # Setting screen size and position on screen using
                                                             # the screen width and height, using f string literal
                                                             # to pass the arguments of height and width of screen

    frame = LabelFrame(root, borderwidth=0, highlightthickness=0)       # Creating a frame to put contents
    frame.grid(row=0,column=0)  # Putting frame in the window


    # Functions
    # Move forward in elements
    def forward(x, top):  # Passing current element number and the current top window as arguments

        '''
        This function defines the functionality of the forward button on the elements window.
        It destroys the current element window and creates a new window and redefines all the functionality
        for the next element and then displays the next element information by reading it from the file of
        the new element along with the image.
        '''

        global elemDict  # Global dictionary containing names and atomic numbers of all elements
        global img       # Global variable to hold the element image
        # Defining the font to be used
        myFont = Font(family="Helvetica",
                      size=10,
                      weight="bold")
        file = open("Elements\\"+ elemDict[str(x+1)] +".txt", 'r')   # Opening file of next element to work with
        top.destroy()  # Destroy the window of previous element
        # Creating and defining new window
        top = Toplevel()  # Creating top level instance
        top.resizable(False, False)  # Window in not resizable
        tip = Balloon(top)  # Tooltip initiated
        top.title(elemDict[str(x+1)])  # Putting the name of the next element as title(here x is the atomic number
                                       # of currently displayed element which is taken as an argument by the function call)
        top.iconbitmap("921815.ico")  # Setting icon of the window
        img = ImageTk.PhotoImage(Image.open("imgs\\"+ elemDict[str(x+1)] +".jpg"))  # Opening image of the new element
        information = file.read()  # Reading info from the file
        # Status bar at bottom
        # Status bar shows the atomic number of the current element which the user is seeing
```

```
            status = Label(top ,text = f"Element {int(x)+1} of 118", bd=1, relief=SUNKEN)  # Status bar is sunken a little bit
  # from the rest of the text
            status.grid(row=3, column=0, columnspan = 2, sticky = W+E)  # sticky attribute spans the status bar across the width
                                                                        # of the entire window(W+E stands for west to east i.e.
                                                                        # entire width of the window)
            # Lable to carry information of element
            label1 = Label(top, text=information, justify=LEFT, font=myFont, pady=10)
            label1.grid(row=0, column=0)
            # Lable to carry image of the element
            label2 = Label(top, image=img)
            label2.grid(row=0, column=1)

            note_label = Label(top, text = 'Press "esc" to stop audio')
            note_label.grid(column=0, row=1, columnspan=2)


            if x == 117:  # Checking for last second element
                # If the current element is last second than for the next element we need to disable the forward button
                bfor = Button(top, text=">>", command = lambda:forward(int(x)+1,top), justify=RIGHT, state=DISABLED, width=10)
                tip.bind_widget(bfor, balloonmsg = 'Next')

            else:  # If not the last second element
                # We need not disable the forward button
                bfor = Button(top, text=">>", command = lambda:forward(int(x)+1,top), justify=RIGHT, width=10)
                tip.bind_widget(bfor, balloonmsg = 'Next')

            # backbutton need not be checked as there will always be atleast one element before it as we have
            # pressed forward button atleast once so redefining it without any condition
            bback = Button(top, text="<<", command = lambda:backward(int(x)+1,top), justify=LEFT, width=10)
            tip.bind_widget(bback, balloonmsg = 'Back')
            bfor.grid(row = 2, column = 1)
            bback.grid(row = 2, column = 0)
            # Flushing and closing file
            file.flush()  # A good practice to flush the file externally
            file.close()

            # Creating sound effects
            pygame.mixer.init()
            pygame.mixer.music.load('audio//'+elemDict[str(int(x)+1)]+'.mp3')
            pygame.mixer.music.play(loops=0)

            def on_closing():
                try:
                    pygame.mixer.music.stop()
                finally:
                    top.destroy()

            keyboard.add_hotkey('esc', lambda: pygame.mixer.music.stop())
            top.protocol('WM_DELETE_WINDOW', on_closing)

        def backward(x, top):  # Passing current element atomic number and current top window as arguments

            '''
            This function defines the functionality of the back button on the elements window.
            It destroys the current element window and creates a new window and
            redefines all the functionality
            for the previous element and then displays the previous element information
            by reading it from the file of
            the new element along with the image.
            '''

            global elemDict  # Global dictionary containing all the elements with
            atomic numbers
            global img  # Global variable for holding the element image
            # Defining the font to be used
            myFont = Font(family="Helvetica",
                          size=10,
                          weight="bold")
            file = open("Elements\\"+ elemDict[str(x-1)] +".txt", 'r')   # Opening
            previous element file to work with
            top.destroy()  # Destroying the current window which is open
            # Creating and defining new window
            top = Toplevel()  # Creating new window instance
            top.resizable(False, False)  # Window not resizable
            tip = Balloon(top)  # Tooltip initiated
            top.title(elemDict[str(x-1)])  # Taking the name of the element from the
            dictionary using its atomic number
                                            # and assigning it as the title of the window
            top.iconbitmap("921815.ico")  # Setting the icon of the window
            img = ImageTk.PhotoImage(Image.open("imgs\\"+ elemDict[str(x-1)] +".jpg"))
            # Opening the image in the variable
            information = file.read()  # Reading information of the previous element
            from its file
            # Status bar at bottom
            # Status bar shows the atomic number of the current element which the user
            is seeing
            status = Label(top ,text = f"Element {int(x)-1} of 118", bd=1,
            relief=SUNKEN)  # Status bar is sunken a little bit

    # for it to look different from all

    # other text
            status.grid(row=3, column=0, columnspan = 2, sticky = W+E)  # sticky is
            used to extend the status bar so as to
                                                                          # cover the
                                                                          complete width
                                                                          of the
                                                                          window(W+E stands
                                                                          # for west to
                                                                          east i.e.
                                                                          complete width
                                                                          of the window)
            # Lable to carry information
            label1 = Label(top, text=information, justify=LEFT, font=myFont, pady=10)
            label1.grid(row=0, column=0)
            # Lable to carry image
            label2 = Label(top, image=img)
            label2.grid(row=0, column=1)

            note_label = Label(top, text = 'Press "esc" to stop audio')
            note_label.grid(column=0, row=1, columnspan=2)

            # Forward and backward button
            if x == 2:  # If the element is the second element than we need to diable
            the backward function for first element because there is no more element
            before it
                bback = Button(top, text="<<", command = lambda:backward(int(x)-1,top),
                justify=LEFT, state=DISABLED, width=10)
                tip.bind_widget(bback, balloonmsg = 'Back')

            else:  # If the current element is not the second element
                bback = Button(top, text="<<", command = lambda:backward(int(x)-1,top),
                justify=LEFT, width=10)
                tip.bind_widget(bback, balloonmsg = 'Back')
            # Not imposing any condition on forward button as there will always be atleast one element before
            # the current element as we have pressed the back button atleast once
            bfor = Button(top, text=">>", command = lambda:forward(int(x)-1,top), justify=RIGHT, width=10)
            tip.bind_widget(bfor, balloonmsg = 'Next')
            bfor.grid(row = 2, column = 1)
            bback.grid(row = 2, column = 0)
            # Flushing and closing file
            file.flush()  # A good practice to flush the file externally
            file.close()

            # Creating sound effects
            pygame.mixer.init()
            pygame.mixer.music.load('audio//'+elemDict[str(int(x)-1)]+'.mp3')
            pygame.mixer.music.play(loops=0)

            def on_closing():
                try:
                    pygame.mixer.music.stop()
                finally:
                    top.destroy()

            keyboard.add_hotkey('esc', lambda: pygame.mixer.music.stop())
            top.protocol('WM_DELETE_WINDOW', on_closing)

        def info(x):  # Fuction to extract related information and images and put it up on different window
            global elemDict  # Dictionary containing all the elements and atomic numbers
            global img  # global variable to hold image
            # Defining the font to be used
            myFont = Font(family="Helvetica",
                          size=10,
                          weight="bold")
            file = open("Elements\\"+ elemDict[x] +".txt", 'r', encoding='utf-8')  # Opening file to work with
            # Creating and defining new window
            top = Toplevel()  # Creating new top window to open over the main window
            top.resizable(False, False)  # Window not resizable
            tip = Balloon(top)  # Initialising tooltip
            top.title(elemDict[x])  # Setting title of window
            top.iconbitmap("921815.ico")  # Setting icon of the window
            img = ImageTk.PhotoImage(Image.open("imgs\\"+ elemDict[x] +".jpg"))  # Opening image in variable
            information = file.read()  #  Reading information from file

            # Status bar at bottom
            # Status bar shows the atomic number of the current element
            status = Label(top ,text = f"Element {x} of 118", bd=1, relief=SUNKEN)  # Status bar is sunken a bit inside the
                                                                                    # screen so as to differentiate it from
                                                                                    # other text
            status.grid(row=3, column=0, columnspan = 2, sticky = W+E)  # Sticky is used so as to extend the status bar over the
                                                                        # complete window screen
            # Lable to carry information
            label1 = Label(top, text=information, justify=LEFT, font=myFont, pady=10)
            label1.grid(row=0, column=0)
            # Lable to carry image
            label2 = Label(top, image=img)
            label2.grid(row=0, column=1)
            # Flushing and closing file
            file.flush()  # A good practice to flush the file externally
            # Closing file
            file.close()
            note_label = Label(top, text = 'Press "esc" to stop audio')
            note_label.grid(column=0, row=1, columnspan=2)

            # Forward and backward button
            if int(x)!= 1 and int(x) != 118:  # Checking that the element is not the first or the last element
                # If element is neither first or last then we need to enable both forward and backward button
                bfor = Button(top, text=">>", command =lambda: forward(int(x),top), justify=RIGHT, width=10)
                tip.bind_widget(bfor, balloonmsg = 'Next')

                bback = Button(top, text="<<", command =lambda: backward(int(x), top), justify=LEFT, width=10)
                tip.bind_widget(bback, balloonmsg = 'Back')

                bfor.grid(row = 2, column = 1)
                bback.grid(row = 2, column = 0)

            elif int(x) == 1:  # Checking if the element is the first element
                # If the element is the first element than we need to diable the back button since there
                # is no element before the very first element
                bfor = Button(top, text=">>", command =lambda: forward(int(x),top), justify=RIGHT, width=10)
                tip.bind_widget(bfor, balloonmsg = 'Next')

                bback = Button(top, text="<<", command =lambda: backward(int(x), top), justify=LEFT, state=DISABLED, width=10)
                tip.bind_widget(bback, balloonmsg = 'Back')

                bfor.grid(row = 2, column = 1)
                bback.grid(row = 2, column = 0)

            elif int(x) == 118:  # Checking if the element is the last element or not
                # If the element is the last element than we need to disable the forward button since there
                # is no element after the very last element
                bfor = Button(top, text=">>", command =lambda: forward(int(x),top), justify=RIGHT, state=DISABLED, width=10)
                tip.bind_widget(bfor, balloonmsg = 'Next')

                bback = Button(top, text="<<", command =lambda: backward(int(x), top), justify=LEFT, width=10)
                tip.bind_widget(bback, balloonmsg = 'Back')

                bfor.grid(row = 2, column = 1)
                bback.grid(row = 2, column = 0)

            # Creating sound effects
            pygame.mixer.init()
            pygame.mixer.music.load('audio//'+elemDict[x]+'.mp3')
            pygame.mixer.music.play(loops=0)

            def on_closing():
                try:
                    pygame.mixer.music.stop()
                finally:
                    top.destroy()

            keyboard.add_hotkey('esc', lambda: pygame.mixer.music.stop())
            top.protocol('WM_DELETE_WINDOW', on_closing)

        def selected(event):

            '''
            This function is executed when an option from the drop-down box(combo box) is selected
```

```python
            It has the options to logout from the current account or to change the current account
            '''

            def splash_1():

                '''
                This function creates a splash window showing thankyou message and self destroys
                itself after 1000 miliseconds i.e. 1 second and the program gets terminated
                '''

                global thanks_img  # Global variable to hold the thanks image
                splash_2 = Tk()  # Creating splash window
                splash_2.iconbitmap('921815.ico')  # Setting window icon
                splash_2.resizable(False, False)  # Window not resizable
                splash_2.geometry('562x270+450+200')  # Setting window size
                splash_2.title('Thank You!')  # Setting window title
                thanks_img = ImageTk.PhotoImage(file='ty.jpg')  # Opening image in variable
                splash_2_canvas = Canvas(splash_2, width=562, height=280)  # Creating canvas
                splash_2_canvas.pack(fill='both', expand=True)  # Packing canvas in the window
                splash_2_canvas.create_image(0,0, image=thanks_img, anchor='nw')  # Putting image into canvas

                splash_2.after(1000, lambda:splash_2.destroy())  # Function to automatically destroy the splash window

            selected = drop.get()  # Extracting the selected option from drop down menu
            if selected == 'Options':
                pass
            elif selected == 'Logout':  # If logout option selected
                # Asking the user to confirm that they really wnat to logout
                reply = messagebox.askquestion('Logout','Are you sure, \nYou want to Logout?')
                if reply == 'yes':  # If they reply 'yes'
                    root.destroy()  # Destroy the root window
                    splash_1()  # Executing splash_1 window which contains the thankyou message
            elif selected == 'Change Account':  # If user selected the change account option
                root.destroy()  # Destroy the root window
                main()  # Call the main function which executes the program again from signin window



        # NOTE label
        noteLabel = Label(frame, text="NOTE : CLick on the elements\nto know about them", justify=CENTER, font=('Helvetica',10))
        noteLabel.grid(row=0, column=6, columnspan=4)

        # Copyright Label
        copy_right_Label = Label(frame, text="\u00a9"+" 2020 Vinit Mehta", justify=CENTER, font=('Helvetica',15))
        copy_right_Label.grid(row=1, column=6, columnspan=4)

        # Drop-Down Menu(Combo Box)
        options = ['Options','Logout', 'Change Account']  # List containing the list of option to be displayed in drop-down menu
        drop = ttk.Combobox(frame,value=options,width=20)  # Creating combo box
        drop.grid(row=0, column=14, columnspan=2)  # Putting combo box up on screen
        drop.current(0)  # Setting the default value to be displayed in combo box
        drop.bind('<<ComboboxSelected>>', selected)  # Action to be taken when some option is selected

        # Creating all 118 elements button groupwise
        # Two step process
        # First defining it using Button method
        # Second putting it up on the screen using grid method

        # Group 1

        b1 = Button(frame, text="1\nH\nHydrogen\n1.0", padx=2, bg="white", command=lambda: info("1"))
        b1.grid(row=0,column=0)
        b3 = Button(frame, text="3\nLi\nLithium\n6.9", padx=8, bg="yellow", command=lambda: info("3"))
        b3.grid(row=1,column=0)
        b11 = Button(frame, text="11\nNa\nSodium\n23.0", padx=8, bg="yellow", command=lambda: info("11"))
        b11.grid(row=2,column=0)
        b19 = Button(frame, text="19\nK\nPotassium\n39.1", bg="yellow", command=lambda: info("19"))
        b19.grid(row=3,column=0)
        b37 = Button(frame, text="37\nRb\nRubidium\n85.5", padx=2, bg="yellow", command=lambda: info("37"))
        b37.grid(row=4,column=0)
        b55 = Button(frame, text="55\nCs\nCesium\n132.9", padx=8, bg="yellow", command=lambda: info("55"))
        b55.grid(row=5,column=0)
        b87 = Button(frame, text="87\nFr\nFrancium\n223.0", padx=4, bg="yellow", command=lambda: info("87"))
        b87.grid(row=6,column=0)

        # Group 2

        b4 = Button(frame, text="4\nBe\nBeryllium\n9.0", padx=8 , bg="#c92a6e", command=lambda: info("4"))
        b4.grid(row=1,column=1)
        b12 = Button(frame, text="12\nMg\nMagnesium\n24.3", bg="#c92a6e", command=lambda: info("12"))
        b12.grid(row=2,column=1)
        b20 = Button(frame, text="20\nCa\nCalcium\n40.1", padx=10, bg="#c92a6e", command=lambda: info("20"))
        b20.grid(row=3,column=1)
        b38 = Button(frame, text="38\nSr\nStrontium\n87.6", padx=6, bg="#c92a6e", command=lambda: info("38"))
        b38.grid(row=4,column=1)
        b56 = Button(frame, text="56\nBa\nBarium\n137.3", padx=13, bg="#c92a6e", command=lambda: info("56"))
        b56.grid(row=5,column=1)
        b88 = Button(frame, text="88\nRa\nRadium\n226.0", padx=12, bg="#c92a6e", command=lambda: info("88"))
        b88.grid(row=6,column=1)

        # Group 3

        b21 = Button(frame, text="21\nSc\nScandium\n45.0", padx=5, bg="#ffd900", command=lambda: info("21"))
        b21.grid(row=3,column=2)
        b39 = Button(frame, text="39\nY\nYttrium\n88.9", padx=12, bg="#ffd900", command=lambda: info("39"))
        b39.grid(row=4,column=2)
        b57 = Button(frame, text="57\nLa*\nLanthanum\n138.9", padx=2, bg="#ff5500", command=lambda: info("57"))
        b57.grid(row=5,column=2)
        b89 = Button(frame, text="89\nAc**\nActinium\n227.0", padx=8, bg="#f7ff5e", command=lambda: info("89"))
        b89.grid(row=6,column=2)

        # Group 4

        b22 = Button(frame, text="22\nTi\nTitanium\n47.9", padx=16, bg="#ffd900", command=lambda: info("22"))
        b22.grid(row=3,column=3)
        b40 = Button(frame, text="40\nZr\nZirconium\n91.2", padx=12, bg="#ffd900",
        command=lambda: info("40"))
        b40.grid(row=4,column=3)
        b72 = Button(frame, text="72\nHf\nHafnium\n178.5", padx=16, bg="#ffd900",
        command=lambda: info("72"))
        b72.grid(row=5,column=3)
        b104 = Button(frame, text="104\nRf\nRutherfordium\n261", padx=0, bg="#ffd900",
        command=lambda: info("104"))
        b104.grid(row=6,column=3)

        # Group 5

        b23 = Button(frame, text="23\nV\nVanadium\n50.9", padx=12, bg="#ffd900",
        command=lambda: info("23"))
        b23.grid(row=3,column=4)
        b41 = Button(frame, text="41\nNb\nNiobium\n92.9", padx=16, bg="#ffd900",
        command=lambda: info("41"))
        b41.grid(row=4,column=4)
        b73 = Button(frame, text="73\nTa\nTantalum\n180.9", padx=14, bg="#ffd900",
        command=lambda: info("73"))
        b73.grid(row=5,column=4)
        b105 = Button(frame, text="105\nDb\nDubnium\n262", padx=15, bg="#ffd900",
        command=lambda: info("105"))
        b105.grid(row=6,column=4)

        # Group 6

        b24 = Button(frame, text="24\nCr\nChromium\n52.0", padx=10, bg="#ffd900",
        command=lambda: info("24"))
        b24.grid(row=3,column=5)
        b42 = Button(frame, text="42\nMo\nMolybdenum\n95.9", padx=3, bg="#ffd900",
        command=lambda: info("42"))
        b42.grid(row=4,column=5)
        b74 = Button(frame, text="74\nW\nTungsten\n183.9", padx=14, bg="#ffd900",
        command=lambda: info("74"))
        b74.grid(row=5,column=5)
        b106 = Button(frame, text="106\nSg\nSeaborgium\n263", padx=7, bg="#ffd900",
        command=lambda: info("106"))
        b106.grid(row=6,column=5)

        # Group 7

        b25 = Button(frame, text="25\nMn\nManganese\n54.9", padx=1, bg="#ffd900",
        command=lambda: info("25"))
        b25.grid(row=3,column=6)
        b43 = Button(frame, text="43\nTc\nTechnetium\n98", padx=0, bg="#ffd900",
        command=lambda: info("43"))
        b43.grid(row=4,column=6)
        b75 = Button(frame, text="75\nRe\nRhenium\n186.2", padx=8, bg="#ffd900",
        command=lambda: info("75"))
        b75.grid(row=5,column=6)
        b107 = Button(frame, text="107\nBh\nBohrium\n262", padx=9, bg="#ffd900",
        command=lambda: info("107"))
        b107.grid(row=6,column=6)

        # Group 8

        b26 = Button(frame, text="26\nFe\nIron\n55.9", padx=19, bg="#ffd900",
        command=lambda: info("26"))
        b26.grid(row=3,column=7)
        b44 = Button(frame, text="44\nRu\nRuthenium\n101.0", padx=0, bg="#ffd900",
        command=lambda: info("44"))
        b44.grid(row=4,column=7)
        b76 = Button(frame, text="76\nOs\nOsmium\n190.2", padx=7, bg="#ffd900",
        command=lambda: info("76"))
        b76.grid(row=5,column=7)
        b108 = Button(frame, text="108\nHs\nHassium\n264", padx=7, bg="#ffd900",
        command=lambda: info("108"))
        b108.grid(row=6,column=7)

        # Group 9

        b27 = Button(frame, text="27\nCo\nCobalt\n58.9", padx=13, bg="#ffd900", command=lambda: info("27"))
        b27.grid(row=3,column=8)
        b45 = Button(frame, text="45\nRh\nRhodium\n102.9", padx=6, bg="#ffd900", command=lambda: info("45"))
        b45.grid(row=4,column=8)
        b77 = Button(frame, text="77\nIr\nIridium\n196.9", padx=12, bg="#ffd900", command=lambda: info("77"))
        b77.grid(row=5,column=8)
        b109 = Button(frame, text="109\nMt\nMeitnerium\n268", padx=0, bg="#ffd900", command=lambda: info("109"))
        b109.grid(row=6,column=8)

        # Group 10

        b28 = Button(frame, text="28\nNi\nNickel\n58.7", padx=21, bg="#ffd900", command=lambda: info("28"))
        b28.grid(row=3,column=9)
        b46 = Button(frame, text="46\nPd\nPalladium\n106.4", padx=11, bg="#ffd900", command=lambda: info("46"))
        b46.grid(row=4,column=9)
        b78 = Button(frame, text="78\nPt\nPlatinum\n192.2", padx=14, bg="#ffd900", command=lambda: info("78"))
        b78.grid(row=5,column=9)
        b110 = Button(frame, text="110\nDs\nDarmstadtium\n261.9", padx=0, bg="#ffd900", command=lambda: info("110"))
        b110.grid(row=6,column=9)

        # Group 11

        b29 = Button(frame, text="29\nCu\nCopper\n63.5", padx=16, bg="#ffd900", command=lambda: info("29"))
        b29.grid(row=3,column=10)
        b47 = Button(frame, text="47\nAg\nSilver\n107.9", padx=22, bg="#ffd900", command=lambda: info("47"))
        b47.grid(row=4,column=10)
        b79 = Button(frame, text="79\nAu\nGold\n195", padx=23, bg="#ffd900", command=lambda: info("79"))
        b79.grid(row=5,column=10)
        b111 = Button(frame, text="111\nRg\nRoentgenium\n271.8", padx=0, bg="#ffd900", command=lambda: info("111"))
        b111.grid(row=6,column=10)

        # Group 12

        b30 = Button(frame, text="30\nZn\nZinc\n65.4", padx=23, bg="#ffd900", command=lambda: info("30"))
        b30.grid(row=3,column=11)
        b48 = Button(frame, text="48\nCd\nCadmium\n112.4", padx=8, bg="#ffd900", command=lambda: info("48"))
        b48.grid(row=4,column=11)
        b80 = Button(frame, text="80\nHg\nMercury\n200.6", padx=12, bg="#ffd900", command=lambda: info("80"))
        b80.grid(row=5,column=11)
        b112 = Button(frame, text="112\nCn\nCopernicium\n285", padx=0, bg="#ffd900", command=lambda: info("112"))
        b112.grid(row=6,column=11)

        # Group 13

        b5 = Button(frame, text="5\nB\nBoron\n9.0", padx=15, bg="pink", command=lambda: info("5"))
        b5.grid(row=1,column=12)
        b13 = Button(frame, text="13\nAl\nAluminium\n24.3",padx=1, bg="#352f9e", command=lambda: info("13"))
        b13.grid(row=2,column=12)
        b31 = Button(frame, text="31\nGa\nGallium\n40.1", padx=11, bg="#352f9e",
        command=lambda: info("31"))
        b31.grid(row=3,column=12)
        b49 = Button(frame, text="49\nIn\nIndium\n87.6", padx=13, bg="#352f9e",
        command=lambda: info("49"))
        b49.grid(row=4,column=12)
        b81 = Button(frame, text="81\nTl\nThalium\n137.3", padx=10, bg="#352f9e",
        command=lambda: info("81"))
        b81.grid(row=5,column=12)
        b113 = Button(frame, text="113\nNh\nNihonium\n286", padx=5, bg="#ffd900",
        command=lambda: info("113"))
        b113.grid(row=6,column=12)

        # Group 14

        b6 = Button(frame, text="6\nC\nCarbon\n12.0", padx=12, bg="green",
        command=lambda: info("6"))
        b6.grid(row=1,column=13)
        b14 = Button(frame, text="14\nSi\nSilicon\n28.1",padx=14, bg="pink",
        command=lambda: info("14"))
        b14.grid(row=2,column=13)
        b32 = Button(frame, text="32\nGe\nGermanium\n72.6", padx=0, bg="pink",
        command=lambda: info("32"))
        b32.grid(row=3,column=13)
        b50 = Button(frame, text="50\nSn\nTin\n118.7", padx=18, bg="#352f9e",
        command=lambda: info("50"))
        b50.grid(row=4,column=13)
        b82 = Button(frame, text="82\nPb\nLead\n207.2", padx=18, bg="#352f9e",
        command=lambda: info("82"))
        b82.grid(row=5,column=13)
        b114 = Button(frame, text="114\nFl\nFlerovium\n289", padx=5, bg="#ffd900",
        command=lambda: info("114"))
        b114.grid(row=6,column=13)

        # Group 15

        b7 = Button(frame, text="7\nN\nNitrogen\n14.0", padx=13, bg="green",
        command=lambda: info("7"))
        b7.grid(row=1,column=14)
        b15 = Button(frame, text="15\nP\nPhosphorus\n31.0",padx=5, bg="green",
        command=lambda: info("15"))
        b15.grid(row=2,column=14)
        b33 = Button(frame, text="33\nAs\nArsenic\n74.9", padx=17, bg="pink",
        command=lambda: info("33"))
        b33.grid(row=3,column=14)
        b51 = Button(frame, text="51\nSb\nAntimony\n121.8", padx=10, bg="pink",
        command=lambda: info("51"))
        b51.grid(row=4,column=14)
        b83 = Button(frame, text="83\nBi\nBismuth\n209.0", padx=14, bg="#352f9e",
        command=lambda: info("83"))
        b83.grid(row=5,column=14)
        b115 = Button(frame, text="115\nMc\nMoscovium\n288", padx=5, bg="#ffd900",
        command=lambda: info("115"))
        b115.grid(row=6,column=14)

        # Group 16

        b8 = Button(frame, text="8\nO\nOxygen\n16.0", padx=13, bg="green",
        command=lambda: info("8"))
        b8.grid(row=1,column=15)
        b16 = Button(frame, text="16\nS\nSulphur\n32.1",padx=13, bg="green",
        command=lambda: info("16"))
        b16.grid(row=2,column=15)
        b34 = Button(frame, text="34\nSe\nSelenium\n79.0", padx=10, bg="green",
        command=lambda: info("34"))
        b34.grid(row=3,column=15)
        b52 = Button(frame, text="52\nTe\nTellurium\n127.6", padx=8, bg="pink",
        command=lambda: info("52"))
        b52.grid(row=4,column=15)
        b84 = Button(frame, text="84\nPo\nPolonium\n209.0", padx=8, bg="pink", command=lambda: info("84"))
        b84.grid(row=5,column=15)
        b116 = Button(frame, text="116\nLv\nLivermorium\n292", padx=0, bg="#ffd900", command=lambda: info("116"))
        b116.grid(row=6,column=15)

        # Group 17

        b9 = Button(frame, text="9\nF\nFluorine\n19.0", padx=11, bg="green", command=lambda: info("9"))
        b9.grid(row=1,column=16)
        b17 = Button(frame, text="17\nCl\nChlorine\n35.5",padx=10, bg="green", command=lambda: info("17"))
        b17.grid(row=2,column=16)
        b35 = Button(frame, text="35\nBr\nBromine\n79.9", padx=10, bg="green", command=lambda: info("35"))
        b35.grid(row=3,column=16)
        b53 = Button(frame, text="53\nI\nIodine\n126.9", padx=16, bg="green", command=lambda: info("53"))
        b53.grid(row=4,column=16)
        b85 = Button(frame, text="85\nAt\nAstatine\n210", padx=11, bg="green", command=lambda: info("85"))
        b85.grid(row=5,column=16)
        b117 = Button(frame, text="117\nTs\nTennessine\n294", padx=3, bg="#ffd900", command=lambda: info("117"))
        b117.grid(row=6,column=16)

        # Group 18

        b2 = Button(frame, text="2\nHe\nHelium\n2.0", padx=10, bg="#9943ab", command=lambda: info("2"))
        b2.grid(row=0,column=17)
        b10 = Button(frame, text="10\nNe\nNeon\n20.2", padx=15, bg="#9943ab", command=lambda: info("10"))
        b10.grid(row=1,column=17)
        b18 = Button(frame, text="18\nAr\nArgon\n40.0", padx=13, bg="#9943ab", command=lambda: info("18"))
        b18.grid(row=2,column=17)
        b36 = Button(frame, text="36\nKr\nKrypton\n83.8", padx=8, bg="#9943ab", command=lambda: info("36"))
        b36.grid(row=3,column=17)
        b54 = Button(frame, text="54\nXe\nXenon\n131.3", padx=12, bg="#9943ab", command=lambda: info("54"))
        b54.grid(row=4,column=17)
        b86 = Button(frame, text="86\nRn\nRadon\n222", padx=12, bg="#9943ab", command=lambda: info("86"))
        b86.grid(row=5,column=17)
        b118 = Button(frame, text="118\nOg\nOganesson\n294", padx=0, bg="#ffd900", command=lambda: info("118"))
        b118.grid(row=6,column=17)

        # Empty label

        label1 = Label(frame, text="\n\n")
        label1.grid(row=7, column=0, columnspan=18)

        # Lanthanides

        bL = Label(frame, text="*Lanthanides", justify=RIGHT)
        bL.grid(row=8, column=2)
        b58 = Button(frame, text="58\nCe\nCerium\n140.1", padx=20, bg="#ff5500", command=lambda: info("58"))
        b58.grid(row=8, column=3)
        b59 = Button(frame, text="59\nPr\nPraseodymium\n140.9", padx=0, bg="#ff5500", command=lambda: info("59"))
        b59.grid(row=8, column=4)
        b60 = Button(frame, text="60\nNd\nNeodymium\n144.2", padx=4, bg="#ff5500",
        command=lambda: info("60"))
        b60.grid(row=8, column=5)
        b61 = Button(frame, text="61\nPm\nPromethium\n145", padx=0, bg="#ff5500",
        command=lambda: info("61"))
        b61.grid(row=8, column=6)
        b62 = Button(frame, text="62\nSm\nSamarium\n150.4", padx=2, bg="#ff5500",
        command=lambda: info("62"))
        b62.grid(row=8, column=7)
        b63 = Button(frame, text="63\nEu\nEuropium\n152", padx=4, bg="#ff5500",
        command=lambda: info("63"))
        b63.grid(row=8, column=8)
        b64 = Button(frame, text="64\nGd\nGadolinium\n157.3", padx=7, bg="#ff5500",
        command=lambda: info("64"))
        b64.grid(row=8, column=9)
        b65 = Button(frame, text="65\nTb\nTerbium\n158.9", padx=13, bg="#ff5500",
        command=lambda: info("65"))
        b65.grid(row=8, column=10)
        b66 = Button(frame, text="66\nDy\nDysprosium\n162.5", padx=2, bg="#ff5500",
        command=lambda: info("66"))
        b66.grid(row=8, column=11)
        b67 = Button(frame, text="67\nHo\nHolmium\n164.9", padx=6, bg="#ff5500",
        command=lambda: info("67"))
        b67.grid(row=8, column=12)
        b68 = Button(frame, text="68\nEr\nErbium\n167.3", padx=12, bg="#ff5500",
        command=lambda: info("68"))
        b68.grid(row=8, column=13)
        b69 = Button(frame, text="69\nTm\nThullium\n168.9", padx=12, bg="#ff5500",
        command=lambda: info("69"))
        b69.grid(row=8, column=14)
        b70 = Button(frame, text="70\nYb\nYtterbium\n173", padx=7, bg="#ff5500",
        command=lambda: info("70"))
        b70.grid(row=8, column=15)
        b71 = Button(frame, text="71\nLu\nLutetium\n175", padx=8, bg="#ff5500",
        command=lambda: info("71"))
        b71.grid(row=8, column=16)

        # Actinides

        bA = Label(frame, text="**Actinides", justify=RIGHT)
        bA.grid(row=9, column=2)
        b90 = Button(frame, text="90\nTh\nThorium\n232.0", padx=16, bg="#f7ff5e",
        command=lambda: info("90"))
        b90.grid(row=9, column=3)
        b91 = Button(frame, text="91\nPa\nProtactinium\n231", padx=6, bg="#f7ff5e",
        command=lambda: info("91"))
        b91.grid(row=9, column=4)
        b92 = Button(frame, text="92\nU\nUranium\n238.0", padx=14, bg="#f7ff5e",
        command=lambda: info("92"))
        b92.grid(row=9, column=5)
        b93 = Button(frame, text="93\nNp\nNeptunium\n237", padx=2, bg="#f7ff5e",
        command=lambda: info("93"))
        b93.grid(row=9, column=6)
        b94 = Button(frame, text="94\nPu\nPlutonium\n244", padx=1, bg="#f7ff5e",
        command=lambda: info("94"))
        b94.grid(row=9, column=7)
        b95 = Button(frame, text="95\nAm\nAmericium\n243", padx=1, bg="#f7ff5e",
        command=lambda: info("95"))
        b95.grid(row=9, column=8)
        b96 = Button(frame, text="96\nCm\nCurium\n247", padx=17, bg="#f7ff5e",
        command=lambda: info("96"))
        b96.grid(row=9, column=9)
        b97 = Button(frame, text="97\nBk\nBerkelium\n247", padx=10, bg="#f7ff5e",
        command=lambda: info("97"))
        b97.grid(row=9, column=10)
        b98 = Button(frame, text="98\nCf\nCalifornium\n251", padx=2, bg="#f7ff5e",
        command=lambda: info("98"))
        b98.grid(row=9, column=11)
        b99 = Button(frame, text="99\nEs\nEinstenium\n252", padx=2, bg="#f7ff5e",
        command=lambda: info("99"))
        b99.grid(row=9, column=12)
        b100 = Button(frame, text="100\nFm\nFermium\n257", padx=7, bg="#f7ff5e",
        command=lambda: info("100"))
        b100.grid(row=9, column=13)
        b101 = Button(frame, text="101\nMd\nMendelevium\n258", padx=0, bg="#f7ff5e",
        command=lambda: info("101"))
        b101.grid(row=9, column=14)
        b102 = Button(frame, text="102\nNo\nNobelium\n259", padx=6, bg="#f7ff5e",
        command=lambda: info("102"))
        b102.grid(row=9, column=15)
        b103 = Button(frame, text="103\nLr\nLawrencium\n262", padx=0, bg="#f7ff5e",
        command=lambda: info("103"))
        b103.grid(row=9, column=16)

        # Initiate tooltip
        tip = Balloon(root)

        # Bind tooltip to buttons
        tip.bind_widget(b1, balloonmsg = elemDict["1"])
        tip.bind_widget(b2, balloonmsg = elemDict["2"])
        tip.bind_widget(b3, balloonmsg = elemDict["3"])
        tip.bind_widget(b4, balloonmsg = elemDict["4"])
        tip.bind_widget(b5, balloonmsg = elemDict["5"])
        tip.bind_widget(b6, balloonmsg = elemDict["6"])
        tip.bind_widget(b7, balloonmsg = elemDict["7"])
        tip.bind_widget(b8, balloonmsg = elemDict["8"])
        tip.bind_widget(b9, balloonmsg = elemDict["9"])
        tip.bind_widget(b10, balloonmsg = elemDict["10"])
        tip.bind_widget(b11, balloonmsg = elemDict["11"])
        tip.bind_widget(b12, balloonmsg = elemDict["12"])
        tip.bind_widget(b13, balloonmsg = elemDict["13"])
        tip.bind_widget(b14, balloonmsg = elemDict["14"])
        tip.bind_widget(b15, balloonmsg = elemDict["15"])
        tip.bind_widget(b16, balloonmsg = elemDict["16"])
        tip.bind_widget(b17, balloonmsg = elemDict["17"])
        tip.bind_widget(b18, balloonmsg = elemDict["18"])
        tip.bind_widget(b19, balloonmsg = elemDict["19"])
        tip.bind_widget(b20, balloonmsg = elemDict["20"])
        tip.bind_widget(b21, balloonmsg = elemDict["21"])
        tip.bind_widget(b22, balloonmsg = elemDict["22"])
        tip.bind_widget(b23, balloonmsg = elemDict["23"])
        tip.bind_widget(b24, balloonmsg = elemDict["24"])
        tip.bind_widget(b25, balloonmsg = elemDict["25"])
        tip.bind_widget(b26, balloonmsg = elemDict["26"])
        tip.bind_widget(b27, balloonmsg = elemDict["27"])
        tip.bind_widget(b28, balloonmsg = elemDict["28"])
        tip.bind_widget(b29, balloonmsg = elemDict["29"])
        tip.bind_widget(b30, balloonmsg = elemDict["30"])
        tip.bind_widget(b31, balloonmsg = elemDict["31"])
        tip.bind_widget(b32, balloonmsg = elemDict["32"])
        tip.bind_widget(b33, balloonmsg = elemDict["33"])
        tip.bind_widget(b34, balloonmsg = elemDict["34"])
        tip.bind_widget(b35, balloonmsg = elemDict["35"])
        tip.bind_widget(b36, balloonmsg = elemDict["36"])
        tip.bind_widget(b37, balloonmsg = elemDict["37"])
        tip.bind_widget(b38, balloonmsg = elemDict["38"])
        tip.bind_widget(b39, balloonmsg = elemDict["39"])
        tip.bind_widget(b40, balloonmsg = elemDict["40"])
        tip.bind_widget(b41, balloonmsg = elemDict["41"])
        tip.bind_widget(b42, balloonmsg = elemDict["42"])
        tip.bind_widget(b43, balloonmsg = elemDict["43"])
        tip.bind_widget(b44, balloonmsg = elemDict["44"])
        tip.bind_widget(b45, balloonmsg = elemDict["45"])
        tip.bind_widget(b46, balloonmsg = elemDict["46"])
        tip.bind_widget(b47, balloonmsg = elemDict["47"])
        tip.bind_widget(b48, balloonmsg = elemDict["48"])
        tip.bind_widget(b49, balloonmsg = elemDict["49"])
        tip.bind_widget(b50, balloonmsg = elemDict["50"])
        tip.bind_widget(b51, balloonmsg = elemDict["51"])
        tip.bind_widget(b52, balloonmsg = elemDict["52"])
        tip.bind_widget(b53, balloonmsg = elemDict["53"])
        tip.bind_widget(b54, balloonmsg = elemDict["54"])
        tip.bind_widget(b55, balloonmsg = elemDict["55"])
        tip.bind_widget(b56, balloonmsg = elemDict["56"])
        tip.bind_widget(b57, balloonmsg = elemDict["57"])
        tip.bind_widget(b58, balloonmsg = elemDict["58"])
        tip.bind_widget(b59, balloonmsg = elemDict["59"])
        tip.bind_widget(b60, balloonmsg = elemDict["60"])
        tip.bind_widget(b61, balloonmsg = elemDict["61"])
        tip.bind_widget(b62, balloonmsg = elemDict["62"])
        tip.bind_widget(b63, balloonmsg = elemDict["63"])
        tip.bind_widget(b64, balloonmsg = elemDict["64"])
        tip.bind_widget(b65, balloonmsg = elemDict["65"])
        tip.bind_widget(b66, balloonmsg = elemDict["66"])
        tip.bind_widget(b67, balloonmsg = elemDict["67"])
        tip.bind_widget(b68, balloonmsg = elemDict["68"])
        tip.bind_widget(b69, balloonmsg = elemDict["69"])
        tip.bind_widget(b70, balloonmsg = elemDict["70"])
        tip.bind_widget(b71, balloonmsg = elemDict["71"])
        tip.bind_widget(b72, balloonmsg = elemDict["72"])
        tip.bind_widget(b73, balloonmsg = elemDict["73"])
        tip.bind_widget(b74, balloonmsg = elemDict["74"])
        tip.bind_widget(b75, balloonmsg = elemDict["75"])
        tip.bind_widget(b76, balloonmsg = elemDict["76"])
        tip.bind_widget(b77, balloonmsg = elemDict["77"])
        tip.bind_widget(b78, balloonmsg = elemDict["78"])
        tip.bind_widget(b79, balloonmsg = elemDict["79"])
        tip.bind_widget(b80, balloonmsg = elemDict["80"])
        tip.bind_widget(b81, balloonmsg = elemDict["81"])
        tip.bind_widget(b82, balloonmsg = elemDict["82"])
        tip.bind_widget(b83, balloonmsg = elemDict["83"])
        tip.bind_widget(b84, balloonmsg = elemDict["84"])
        tip.bind_widget(b85, balloonmsg = elemDict["85"])
        tip.bind_widget(b86, balloonmsg = elemDict["86"])
        tip.bind_widget(b87, balloonmsg = elemDict["87"])
        tip.bind_widget(b88, balloonmsg = elemDict["88"])
        tip.bind_widget(b89, balloonmsg = elemDict["89"])
        tip.bind_widget(b90, balloonmsg = elemDict["90"])
        tip.bind_widget(b91, balloonmsg = elemDict["91"])
        tip.bind_widget(b92, balloonmsg = elemDict["92"])
        tip.bind_widget(b93, balloonmsg = elemDict["93"])
        tip.bind_widget(b94, balloonmsg = elemDict["94"])
        tip.bind_widget(b95, balloonmsg = elemDict["95"])
        tip.bind_widget(b96, balloonmsg = elemDict["96"])
        tip.bind_widget(b97, balloonmsg = elemDict["97"])
        tip.bind_widget(b98, balloonmsg = elemDict["98"])
        tip.bind_widget(b99, balloonmsg = elemDict["99"])
        tip.bind_widget(b100, balloonmsg = elemDict["100"])
        tip.bind_widget(b101, balloonmsg = elemDict["101"])
        tip.bind_widget(b102, balloonmsg = elemDict["102"])
        tip.bind_widget(b103, balloonmsg = elemDict["103"])
        tip.bind_widget(b104, balloonmsg = elemDict["104"])
        tip.bind_widget(b105, balloonmsg = elemDict["105"])
        tip.bind_widget(b106, balloonmsg = elemDict["106"])
        tip.bind_widget(b107, balloonmsg = elemDict["107"])
        tip.bind_widget(b108, balloonmsg = elemDict["108"])
        tip.bind_widget(b109, balloonmsg = elemDict["109"])
        tip.bind_widget(b110, balloonmsg = elemDict["110"])
        tip.bind_widget(b111, balloonmsg = elemDict["111"])
        tip.bind_widget(b112, balloonmsg = elemDict["112"])
        tip.bind_widget(b113, balloonmsg = elemDict["113"])
        tip.bind_widget(b114, balloonmsg = elemDict["114"])
        tip.bind_widget(b115, balloonmsg = elemDict["115"])
        tip.bind_widget(b116, balloonmsg = elemDict["116"])
        tip.bind_widget(b117, balloonmsg = elemDict["117"])
        tip.bind_widget(b118, balloonmsg = elemDict["118"])

        # Defining hover over functions for all element buttons
        # Buttons change color when mouse tip move over them

        def b_1hover(event):
            b1['bg'] = '#d9d3d2'
        def b_2hover(event):
            b2['bg'] = '#773585'
        def b_3hover(event):
            b3['bg'] = '#e0dd04'
        def b_4hover(event):
            b4['bg'] = '#a3295d'
        def b_5hover(event):
            b5['bg'] = '#f7a1e3'
        def b_6hover(event):
            b6['bg'] = '#15591f'
        def b_7hover(event):
            b7['bg'] = '#15591f'
        def b_8hover(event):
            b8['bg'] = '#15591f'
        def b_9hover(event):
            b9['bg'] = '#15591f'
        def b_10hover(event):
            b10['bg'] = '#773585'
        def b_11hover(event):
            b11['bg'] = '#e0dd04'
        def b_12hover(event):
            b12['bg'] = '#a3295d'
        def b_13hover(event):
            b13['bg'] = '#2c2680'
        def b_14hover(event):
            b14['bg'] = '#f7a1e3'
        def b_15hover(event):
            b15['bg'] = '#15591f'
        def b_16hover(event):
            b16['bg'] = '#15591f'
        def b_17hover(event):
            b17['bg'] = '#15591f'
        def b_18hover(event):
            b18['bg'] = '#773585'
        def b_19hover(event):
            b19['bg'] = '#e0dd04'
        def b_20hover(event):
            b20['bg'] = '#a3295d'
        def b_21hover(event):
            b21['bg'] = '#d1b304'
        def b_22hover(event):
            b22['bg'] = '#d1b304'
        def b_23hover(event):
            b23['bg'] = '#d1b304'
        def b_24hover(event):
            b24['bg'] = '#d1b304'
        def b_25hover(event):
            b25['bg'] = '#d1b304'
        def b_26hover(event):
            b26['bg'] = '#d1b304'
        def b_27hover(event):
            b27['bg'] = '#d1b304'
        def b_28hover(event):
            b28['bg'] = '#d1b304'
        def b_29hover(event):
            b29['bg'] = '#d1b304'
        def b_30hover(event):
            b30['bg'] = '#d1b304'
        def b_31hover(event):
            b31['bg'] = '#2c2680'
        def b_32hover(event):
            b32['bg'] = '#f7a1e3'
        def b_33hover(event):
            b33['bg'] = '#f7a1e3'
        def b_34hover(event):
            b34['bg'] = '#15591f'
        def b_35hover(event):
            b35['bg'] = '#15591f'
        def b_36hover(event):
            b36['bg'] = '#773585'
        def b_37hover(event):
            b37['bg'] = '#e0dd04'
        def b_38hover(event):
            b38['bg'] = '#a3295d'
        def b_39hover(event):
            b39['bg'] = '#d1b304'
        def b_40hover(event):
            b40['bg'] = '#d1b304'
        def b_41hover(event):
            b41['bg'] = '#d1b304'
        def b_42hover(event):
            b42['bg'] = '#d1b304'
        def b_43hover(event):
            b43['bg'] = '#d1b304'
        def b_44hover(event):
            b44['bg'] = '#d1b304'
        def b_45hover(event):
            b45['bg'] = '#d1b304'
        def b_46hover(event):
            b46['bg'] = '#d1b304'
        def b_47hover(event):
            b47['bg'] = '#d1b304'
        def b_48hover(event):
            b48['bg'] = '#d1b304'
        def b_49hover(event):
            b49['bg'] = '#2c2680'
        def b_50hover(event):
            b50['bg'] = '#2c2680'
        def b_51hover(event):
            b51['bg'] = '#f7a1e3'
        def b_52hover(event):
            b52['bg'] = '#f7a1e3'
        def b_53hover(event):
            b53['bg'] = '#15591f'
        def b_54hover(event):
            b54['bg'] = '#773585'
        def b_55hover(event):
            b55['bg'] = '#e0dd04'
        def b_56hover(event):
            b56['bg'] = '#a3295d'
        def b_57hover(event):
            b57['bg'] = '#db4f09'
        def b_58hover(event):
            b58['bg'] = '#db4f09'
        def b_59hover(event):
            b59['bg'] = '#db4f09'
        def b_60hover(event):
            b60['bg'] = '#db4f09'
        def b_61hover(event):
            b61['bg'] = '#db4f09'
        def b_62hover(event):
            b62['bg'] = '#db4f09'
        def b_63hover(event):
            b63['bg'] = '#db4f09'
        def b_64hover(event):
            b64['bg'] = '#db4f09'
        def b_65hover(event):
            b65['bg'] = '#db4f09'
        def b_66hover(event):
            b66['bg'] = '#db4f09'
        def b_67hover(event):
            b67['bg'] = '#db4f09'
        def b_68hover(event):
            b68['bg'] = '#db4f09'
        def b_69hover(event):
            b69['bg'] = '#db4f09'
        def b_70hover(event):
            b70['bg'] = '#db4f09'
        def b_71hover(event):
            b71['bg'] = '#db4f09'
        def b_72hover(event):
            b72['bg'] = '#d1b304'
        def b_73hover(event):
            b73['bg'] = '#d1b304'
        def b_74hover(event):
            b74['bg'] = '#d1b304'
        def b_75hover(event):
            b75['bg'] = '#d1b304'
        def b_76hover(event):
            b76['bg'] = '#d1b304'
        def b_77hover(event):
            b77['bg'] = '#d1b304'
        def b_78hover(event):
            b78['bg'] = '#d1b304'
        def b_79hover(event):
            b79['bg'] = '#d1b304'
        def b_80hover(event):
            b80['bg'] = '#d1b304'
        def b_81hover(event):
            b81['bg'] = '#2c2680'
        def b_82hover(event):
            b82['bg'] = '#2c2680'
        def b_83hover(event):
            b83['bg'] = '#2c2680'
        def b_84hover(event):
            b84['bg'] = '#f7a1e3'
        def b_85hover(event):
            b85['bg'] = '#15591f'
        def b_86hover(event):
            b86['bg'] = '#773585'
        def b_87hover(event):
            b87['bg'] = '#e0dd04'
        def b_88hover(event):
            b88['bg'] = '#a3295d'
        def b_89hover(event):
            b89['bg'] = '#c0c74e'
        def b_90hover(event):
            b90['bg'] = '#c0c74e'
        def b_91hover(event):
            b91['bg'] = '#c0c74e'
        def b_92hover(event):
            b92['bg'] = '#c0c74e'
        def b_93hover(event):
            b93['bg'] = '#c0c74e'
        def b_94hover(event):
            b94['bg'] = '#c0c74e'
        def b_95hover(event):
            b95['bg'] = '#c0c74e'
        def b_96hover(event):
            b96['bg'] = '#c0c74e'
        def b_97hover(event):
            b97['bg'] = '#c0c74e'
        def b_98hover(event):
            b98['bg'] = '#c0c74e'
        def b_99hover(event):
            b99['bg'] = '#c0c74e'
        def b_100hover(event):
            b100['bg'] = '#c0c74e'
        def b_101hover(event):
            b101['bg'] = '#c0c74e'
        def b_102hover(event):
            b102['bg'] = '#c0c74e'
        def b_103hover(event):
```

```
        b103['bg'] = '#c0c74e'
    def b_104hover(event):
        b104['bg'] = '#d1b304'
    def b_105hover(event):
        b105['bg'] = '#d1b304'
    def b_106hover(event):
        b106['bg'] = '#d1b304'
    def b_107hover(event):
        b107['bg'] = '#d1b304'
    def b_108hover(event):
        b108['bg'] = '#d1b304'
    def b_109hover(event):
        b109['bg'] = '#d1b304'
    def b_110hover(event):
        b110['bg'] = '#d1b304'
    def b_111hover(event):
        b111['bg'] = '#d1b304'
    def b_112hover(event):
        b112['bg'] = '#d1b304'
    def b_113hover(event):
        b113['bg'] = '#d1b304'
    def b_114hover(event):
        b114['bg'] = '#d1b304'
    def b_115hover(event):
        b115['bg'] = '#d1b304'
    def b_116hover(event):
        b116['bg'] = '#d1b304'
    def b_117hover(event):
        b117['bg'] = '#d1b304'
    def b_118hover(event):
        b118['bg'] = '#d1b304'

    # Functions to restore previous color of the buttons when mouse tip leave the
    element button

    def b_1leave(event):
        b1['bg'] = 'white'
    def b_2leave(event):
        b2['bg'] = '#9943ab'
    def b_3leave(event):
        b3['bg'] = 'yellow'
    def b_4leave(event):
        b4['bg'] = '#c92a6e'
    def b_5leave(event):
        b5['bg'] = 'pink'
    def b_6leave(event):
        b6['bg'] = 'green'
    def b_7leave(event):
        b7['bg'] = 'green'
    def b_8leave(event):
        b8['bg'] = 'green'
    def b_9leave(event):
        b9['bg'] = 'green'
    def b_10leave(event):
        b10['bg'] = '#9943ab'
    def b_11leave(event):
        b11['bg'] = 'yellow'
    def b_12leave(event):
        b12['bg'] = '#c92a6e'
    def b_13leave(event):
        b13['bg'] = '#352f9e'
    def b_14leave(event):
        b14['bg'] = 'pink'
    def b_15leave(event):
        b15['bg'] = 'green'
    def b_16leave(event):
        b16['bg'] = 'green'
    def b_17leave(event):
        b17['bg'] = 'green'
    def b_18leave(event):
        b18['bg'] = '#9943ab'
    def b_19leave(event):
        b19['bg'] = 'yellow'
    def b_20leave(event):
        b20['bg'] = '#c92a6e'
    def b_21leave(event):
        b21['bg'] = '#ffd900'
    def b_22leave(event):
        b22['bg'] = '#ffd900'
    def b_23leave(event):
        b23['bg'] = '#ffd900'
    def b_24leave(event):
        b24['bg'] = '#ffd900'
    def b_25leave(event):
        b25['bg'] = '#ffd900'
    def b_26leave(event):
        b26['bg'] = '#ffd900'
    def b_27leave(event):
        b27['bg'] = '#ffd900'
    def b_28leave(event):
        b28['bg'] = '#ffd900'
    def b_29leave(event):
        b29['bg'] = '#ffd900'
    def b_30leave(event):
        b30['bg'] = '#ffd900'
    def b_31leave(event):
        b31['bg'] = '#352f9e'
    def b_32leave(event):
        b32['bg'] = 'pink'
    def b_33leave(event):
        b33['bg'] = 'pink'
    def b_34leave(event):
        b34['bg'] = 'green'
    def b_35leave(event):
        b35['bg'] = 'green'
    def b_36leave(event):
        b36['bg'] = '#9943ab'
    def b_37leave(event):
        b37['bg'] = 'yellow'
    def b_38leave(event):
        b38['bg'] = '#c92a6e'
    def b_39leave(event):
        b39['bg'] = '#ffd900'
    def b_40leave(event):
        b40['bg'] = '#ffd900'
    def b_41leave(event):
        b41['bg'] = '#ffd900'
    def b_42leave(event):
        b42['bg'] = '#ffd900'
    def b_43leave(event):
        b43['bg'] = '#ffd900'
    def b_44leave(event):
        b44['bg'] = '#ffd900'
    def b_45leave(event):
        b45['bg'] = '#ffd900'
    def b_46leave(event):
        b46['bg'] = '#ffd900'
    def b_47leave(event):
        b47['bg'] = '#ffd900'
    def b_48leave(event):
        b48['bg'] = '#ffd900'
    def b_49leave(event):
        b49['bg'] = '#352f9e'
    def b_50leave(event):
        b50['bg'] = '#352f9e'
    def b_51leave(event):
        b51['bg'] = 'pink'
    def b_52leave(event):
        b52['bg'] = 'pink'
    def b_53leave(event):
        b53['bg'] = 'green'
    def b_54leave(event):
        b54['bg'] = '#9943ab'
    def b_55leave(event):
        b55['bg'] = 'yellow'
    def b_56leave(event):
        b56['bg'] = '#c92a6e'
    def b_57leave(event):
        b57['bg'] = '#ff5500'
    def b_58leave(event):
        b58['bg'] = '#ff5500'
    def b_59leave(event):
        b59['bg'] = '#ff5500'
    def b_60leave(event):
        b60['bg'] = '#ff5500'
    def b_61leave(event):
        b61['bg'] = '#ff5500'
    def b_62leave(event):
        b62['bg'] = '#ff5500'
    def b_63leave(event):
        b63['bg'] = '#ff5500'
    def b_64leave(event):
        b64['bg'] = '#ff5500'
    def b_65leave(event):
        b65['bg'] = '#ff5500'
    def b_66leave(event):
        b66['bg'] = '#ff5500'
    def b_67leave(event):
        b67['bg'] = '#ff5500'
    def b_68leave(event):
        b68['bg'] = '#ff5500'
    def b_69leave(event):
        b69['bg'] = '#ff5500'
    def b_70leave(event):
        b70['bg'] = '#ff5500'
    def b_71leave(event):
        b71['bg'] = '#ff5500'
    def b_72leave(event):
        b72['bg'] = '#ffd900'
    def b_73leave(event):
        b73['bg'] = '#ffd900'
    def b_74leave(event):
        b74['bg'] = '#ffd900'
    def b_75leave(event):
        b75['bg'] = '#ffd900'
    def b_76leave(event):
        b76['bg'] = '#ffd900'
    def b_77leave(event):
        b77['bg'] = '#ffd900'
    def b_78leave(event):
        b78['bg'] = '#ffd900'
    def b_79leave(event):
        b79['bg'] = '#ffd900'
    def b_80leave(event):
        b80['bg'] = '#ffd900'
    def b_81leave(event):
        b81['bg'] = '#352f9e'
    def b_82leave(event):
        b82['bg'] = '#352f9e'
    def b_83leave(event):
        b83['bg'] = '#352f9e'
    def b_84leave(event):
        b84['bg'] = 'pink'
    def b_85leave(event):
        b85['bg'] = 'green'
    def b_86leave(event):
        b86['bg'] = '#9943ab'
    def b_87leave(event):
        b87['bg'] = 'yellow'
    def b_88leave(event):
        b88['bg'] = '#c92a6e'
    def b_89leave(event):
        b89['bg'] = '#f7ff5e'
    def b_90leave(event):
        b90['bg'] = '#f7ff5e'
    def b_91leave(event):
        b91['bg'] = '#f7ff5e'
    def b_92leave(event):
        b92['bg'] = '#f7ff5e'
    def b_93leave(event):
        b93['bg'] = '#f7ff5e'
    def b_94leave(event):
        b94['bg'] = '#f7ff5e'
    def b_95leave(event):
        b95['bg'] = '#f7ff5e'
    def b_96leave(event):
        b96['bg'] = '#f7ff5e'
    def b_97leave(event):
        b97['bg'] = '#f7ff5e'
    def b_98leave(event):
        b98['bg'] = '#f7ff5e'
    def b_99leave(event):
        b99['bg'] = '#f7ff5e'
    def b_100leave(event):
        b100['bg'] = '#f7ff5e'
    def b_101leave(event):
        b101['bg'] = '#f7ff5e'
    def b_102leave(event):
        b102['bg'] = '#f7ff5e'
    def b_103leave(event):
        b103['bg'] = '#f7ff5e'
    def b_104leave(event):
        b104['bg'] = '#ffd900'
    def b_105leave(event):
        b105['bg'] = '#ffd900'
    def b_106leave(event):
        b106['bg'] = '#ffd900'
    def b_107leave(event):
        b107['bg'] = '#ffd900'
    def b_108leave(event):
        b108['bg'] = '#ffd900'
    def b_109leave(event):
        b109['bg'] = '#ffd900'
    def b_110leave(event):
        b110['bg'] = '#ffd900'
    def b_111leave(event):
        b111['bg'] = '#ffd900'
    def b_112leave(event):
        b112['bg'] = '#ffd900'
    def b_113leave(event):
        b113['bg'] = '#ffd900'
    def b_114leave(event):
        b114['bg'] = '#ffd900'
    def b_115leave(event):
        b115['bg'] = '#ffd900'
    def b_116leave(event):
        b116['bg'] = '#ffd900'
    def b_117leave(event):
        b117['bg'] = '#ffd900'
    def b_118leave(event):
        b118['bg'] = '#ffd900'

    # Binding hover functions to respective buttons

    b1.bind('<Enter>', b_1hover)
    b2.bind('<Enter>', b_2hover)
    b3.bind('<Enter>', b_3hover)
    b4.bind('<Enter>', b_4hover)
    b5.bind('<Enter>', b_5hover)
    b6.bind('<Enter>', b_6hover)
    b7.bind('<Enter>', b_7hover)
    b8.bind('<Enter>', b_8hover)
    b9.bind('<Enter>', b_9hover)
    b10.bind('<Enter>', b_10hover)
    b11.bind('<Enter>', b_11hover)
    b12.bind('<Enter>', b_12hover)
    b13.bind('<Enter>', b_13hover)
    b14.bind('<Enter>', b_14hover)
    b15.bind('<Enter>', b_15hover)
    b16.bind('<Enter>', b_16hover)
    b17.bind('<Enter>', b_17hover)
    b18.bind('<Enter>', b_18hover)
    b19.bind('<Enter>', b_19hover)
    b20.bind('<Enter>', b_20hover)
    b21.bind('<Enter>', b_21hover)
    b22.bind('<Enter>', b_22hover)
    b23.bind('<Enter>', b_23hover)
    b24.bind('<Enter>', b_24hover)
    b25.bind('<Enter>', b_25hover)
    b26.bind('<Enter>', b_26hover)
    b27.bind('<Enter>', b_27hover)
    b28.bind('<Enter>', b_28hover)
    b29.bind('<Enter>', b_29hover)
    b30.bind('<Enter>', b_30hover)
    b31.bind('<Enter>', b_31hover)
    b32.bind('<Enter>', b_32hover)
    b33.bind('<Enter>', b_33hover)
    b34.bind('<Enter>', b_34hover)
    b35.bind('<Enter>', b_35hover)
    b36.bind('<Enter>', b_36hover)
    b37.bind('<Enter>', b_37hover)
    b38.bind('<Enter>', b_38hover)
    b39.bind('<Enter>', b_39hover)
    b40.bind('<Enter>', b_40hover)
    b41.bind('<Enter>', b_41hover)
    b42.bind('<Enter>', b_42hover)
    b43.bind('<Enter>', b_43hover)
    b44.bind('<Enter>', b_44hover)
    b45.bind('<Enter>', b_45hover)
    b46.bind('<Enter>', b_46hover)
    b47.bind('<Enter>', b_47hover)
    b48.bind('<Enter>', b_48hover)
    b49.bind('<Enter>', b_49hover)
    b50.bind('<Enter>', b_50hover)
    b51.bind('<Enter>', b_51hover)
    b52.bind('<Enter>', b_52hover)
    b53.bind('<Enter>', b_53hover)
    b54.bind('<Enter>', b_54hover)
    b55.bind('<Enter>', b_55hover)
    b56.bind('<Enter>', b_56hover)
    b57.bind('<Enter>', b_57hover)
    b58.bind('<Enter>', b_58hover)
    b59.bind('<Enter>', b_59hover)
    b60.bind('<Enter>', b_60hover)
    b61.bind('<Enter>', b_61hover)
    b62.bind('<Enter>', b_62hover)
    b63.bind('<Enter>', b_63hover)
    b64.bind('<Enter>', b_64hover)
    b65.bind('<Enter>', b_65hover)
    b66.bind('<Enter>', b_66hover)
    b67.bind('<Enter>', b_67hover)
    b68.bind('<Enter>', b_68hover)
    b69.bind('<Enter>', b_69hover)
    b70.bind('<Enter>', b_70hover)
    b71.bind('<Enter>', b_71hover)
    b72.bind('<Enter>', b_72hover)
    b73.bind('<Enter>', b_73hover)
    b74.bind('<Enter>', b_74hover)
    b75.bind('<Enter>', b_75hover)
    b76.bind('<Enter>', b_76hover)
    b77.bind('<Enter>', b_77hover)
    b78.bind('<Enter>', b_78hover)
    b79.bind('<Enter>', b_79hover)
    b80.bind('<Enter>', b_80hover)
    b81.bind('<Enter>', b_81hover)
    b82.bind('<Enter>', b_82hover)
    b83.bind('<Enter>', b_83hover)
    b84.bind('<Enter>', b_84hover)
    b85.bind('<Enter>', b_85hover)
    b86.bind('<Enter>', b_86hover)
    b87.bind('<Enter>', b_87hover)
    b88.bind('<Enter>', b_88hover)
    b89.bind('<Enter>', b_89hover)
    b90.bind('<Enter>', b_90hover)
    b91.bind('<Enter>', b_91hover)
    b92.bind('<Enter>', b_92hover)
    b93.bind('<Enter>', b_93hover)
    b94.bind('<Enter>', b_94hover)
    b95.bind('<Enter>', b_95hover)
    b96.bind('<Enter>', b_96hover)
    b97.bind('<Enter>', b_97hover)
    b98.bind('<Enter>', b_98hover)
    b99.bind('<Enter>', b_99hover)
    b100.bind('<Enter>', b_100hover)
    b101.bind('<Enter>', b_101hover)
    b102.bind('<Enter>', b_102hover)
    b103.bind('<Enter>', b_103hover)
    b104.bind('<Enter>', b_104hover)
    b105.bind('<Enter>', b_105hover)
    b106.bind('<Enter>', b_106hover)
    b107.bind('<Enter>', b_107hover)
    b108.bind('<Enter>', b_108hover)
    b109.bind('<Enter>', b_109hover)
    b110.bind('<Enter>', b_110hover)
    b111.bind('<Enter>', b_111hover)
    b112.bind('<Enter>', b_112hover)
    b113.bind('<Enter>', b_113hover)
    b114.bind('<Enter>', b_114hover)
    b115.bind('<Enter>', b_115hover)
    b116.bind('<Enter>', b_116hover)
    b117.bind('<Enter>', b_117hover)
    b118.bind('<Enter>', b_118hover)

    # Binding leave function to respective buttons

    b1.bind('<Leave>', b_1leave)
    b2.bind('<Leave>', b_2leave)
    b3.bind('<Leave>', b_3leave)
    b4.bind('<Leave>', b_4leave)
    b5.bind('<Leave>', b_5leave)
    b6.bind('<Leave>', b_6leave)
    b7.bind('<Leave>', b_7leave)
    b8.bind('<Leave>', b_8leave)
    b9.bind('<Leave>', b_9leave)
    b10.bind('<Leave>', b_10leave)
    b11.bind('<Leave>', b_11leave)
    b12.bind('<Leave>', b_12leave)
    b13.bind('<Leave>', b_13leave)
    b14.bind('<Leave>', b_14leave)
    b15.bind('<Leave>', b_15leave)
    b16.bind('<Leave>', b_16leave)
    b17.bind('<Leave>', b_17leave)
    b18.bind('<Leave>', b_18leave)
    b19.bind('<Leave>', b_19leave)
    b20.bind('<Leave>', b_20leave)
    b21.bind('<Leave>', b_21leave)
    b22.bind('<Leave>', b_22leave)
    b23.bind('<Leave>', b_23leave)
    b24.bind('<Leave>', b_24leave)
    b25.bind('<Leave>', b_25leave)
    b26.bind('<Leave>', b_26leave)
    b27.bind('<Leave>', b_27leave)
    b28.bind('<Leave>', b_28leave)
    b29.bind('<Leave>', b_29leave)
    b30.bind('<Leave>', b_30leave)
    b31.bind('<Leave>', b_31leave)
    b32.bind('<Leave>', b_32leave)
    b33.bind('<Leave>', b_33leave)
    b34.bind('<Leave>', b_34leave)
    b35.bind('<Leave>', b_35leave)
    b36.bind('<Leave>', b_36leave)
    b37.bind('<Leave>', b_37leave)
    b38.bind('<Leave>', b_38leave)
    b39.bind('<Leave>', b_39leave)
    b40.bind('<Leave>', b_40leave)
    b41.bind('<Leave>', b_41leave)
    b42.bind('<Leave>', b_42leave)
    b43.bind('<Leave>', b_43leave)
    b44.bind('<Leave>', b_44leave)
    b45.bind('<Leave>', b_45leave)
    b46.bind('<Leave>', b_46leave)
    b47.bind('<Leave>', b_47leave)
    b48.bind('<Leave>', b_48leave)
    b49.bind('<Leave>', b_49leave)
    b50.bind('<Leave>', b_50leave)
    b51.bind('<Leave>', b_51leave)
    b52.bind('<Leave>', b_52leave)
    b53.bind('<Leave>', b_53leave)
    b54.bind('<Leave>', b_54leave)
    b55.bind('<Leave>', b_55leave)
    b56.bind('<Leave>', b_56leave)
    b57.bind('<Leave>', b_57leave)
    b58.bind('<Leave>', b_58leave)
    b59.bind('<Leave>', b_59leave)
    b60.bind('<Leave>', b_60leave)
    b61.bind('<Leave>', b_61leave)
    b62.bind('<Leave>', b_62leave)
    b63.bind('<Leave>', b_63leave)
    b64.bind('<Leave>', b_64leave)
    b65.bind('<Leave>', b_65leave)
    b66.bind('<Leave>', b_66leave)
    b67.bind('<Leave>', b_67leave)
    b68.bind('<Leave>', b_68leave)
    b69.bind('<Leave>', b_69leave)
    b70.bind('<Leave>', b_70leave)
    b71.bind('<Leave>', b_71leave)
    b72.bind('<Leave>', b_72leave)
    b73.bind('<Leave>', b_73leave)
    b74.bind('<Leave>', b_74leave)
    b75.bind('<Leave>', b_75leave)
    b76.bind('<Leave>', b_76leave)
    b77.bind('<Leave>', b_77leave)
    b78.bind('<Leave>', b_78leave)
    b79.bind('<Leave>', b_79leave)
    b80.bind('<Leave>', b_80leave)
    b81.bind('<Leave>', b_81leave)
    b82.bind('<Leave>', b_82leave)
    b83.bind('<Leave>', b_83leave)
    b84.bind('<Leave>', b_84leave)
    b85.bind('<Leave>', b_85leave)
    b86.bind('<Leave>', b_86leave)
    b87.bind('<Leave>', b_87leave)
    b88.bind('<Leave>', b_88leave)
    b89.bind('<Leave>', b_89leave)
    b90.bind('<Leave>', b_90leave)
    b91.bind('<Leave>', b_91leave)
    b92.bind('<Leave>', b_92leave)
    b93.bind('<Leave>', b_93leave)
    b94.bind('<Leave>', b_94leave)
    b95.bind('<Leave>', b_95leave)
    b96.bind('<Leave>', b_96leave)
    b97.bind('<Leave>', b_97leave)
    b98.bind('<Leave>', b_98leave)
    b99.bind('<Leave>', b_99leave)
    b100.bind('<Leave>', b_100leave)
    b101.bind('<Leave>', b_101leave)
    b102.bind('<Leave>', b_102leave)
    b103.bind('<Leave>', b_103leave)
    b104.bind('<Leave>', b_104leave)
    b105.bind('<Leave>', b_105leave)
    b106.bind('<Leave>', b_106leave)
    b107.bind('<Leave>', b_107leave)
    b108.bind('<Leave>', b_108leave)
    b109.bind('<Leave>', b_109leave)
    b110.bind('<Leave>', b_110leave)
    b111.bind('<Leave>', b_111leave)
    b112.bind('<Leave>', b_112leave)
    b113.bind('<Leave>', b_113leave)
    b114.bind('<Leave>', b_114leave)
    b115.bind('<Leave>', b_115leave)
    b116.bind('<Leave>', b_116leave)
    b117.bind('<Leave>', b_117leave)
    b118.bind('<Leave>', b_118leave)

    root.mainloop()

# Creating labels
# Signin main label
signin_canvas.create_text(125, 70, text='Sign In Here', font=('Impact',30,'bold'),fill='#285243')
# Username label
signin_canvas.create_text(85, 110, text='Username:', font=('Helvetica',15,'bold'),fill='black')

# Usename entry widget
username_entry_widget = Entry(window, width = 40,border=2)
username_entry_widget_window = signin_canvas.create_window(155, 135, window = username_entry_widget)

# Password label
signin_canvas.create_text(85, 175, text='Password:', font=('Helvetica',15,'bold'),fill='black')
password_entry_widget = Entry(window, width = 40, border=2, show='*')

# Password entry widget
password_entry_widget_window = signin_canvas.create_window(155, 200, window = password_entry_widget)

# Initiating tooltip
tip = Balloon(window)

# Signin Button
signin_button = Button(window, text='SIGN IN',command=signin,width=30,font=('Helvetica',10,'bold'),bg='#cccccc')
signin_button_window = signin_canvas.create_window(155, 250, window = signin_button)
tip.bind_widget(signin_button, balloonmsg = 'Sign In')

# Signup Button
signup_button = Button(window, text='SIGN UP',command=signup,width = 30,font=('Helvetica',10,'bold'),bg='#cccccc')
signup_button_window = signin_canvas.create_window(155, 300, window = signup_button)
tip.bind_widget(signup_button, balloonmsg = 'Create Account')

window.mainloop()

# Launching the program
main()

```