

```

1  '''
2  -->Please first run the the 'audio_generator' file in the audio folder if the audio
   files are not created
3  -->Please make sure that you have installed the following modules before moving ahead
   with the program
4  1) Tkinter - 'pip install tk'
5  2) Mysql connector - 'pip install mysql-connector-python'
6  3) Pillow (referenced as PIL=Python Image Library) - 'pip install Pillow'
7  4) Pygame - 'pip install pygame'
8  5) Keyboard - 'pip install keyboard'
9  -->These commands are to be executed at command prompt on windows
10 '''
11
12 import mysql.connector as Connector # Python MySQL Connector
13 from tkinter import *               # Main tinker module
14 from tkinter.tix import *           # Module for tool tip
15 from PIL import ImageTk, Image      # Module for image manipulation
16 from tkinter.font import Font       # Module for font
17 from tkinter import messagebox      # Module for pop-up message box
18 from tkinter import ttk             # Module for Combo-box (Drop-down Box)
19 import pygame                       # Module for sound effects
20 import keyboard                      # Module to identify keystrokes
21
22
23 '''
24 Note: Please make sure before executing the program that
25       the database and table with following specifications
26       is already constructed on your PC.
27
28 Database name: username_pass
29 Table name: data
30 Table format: column1 = Username and column2 = Password
31 +-----+-----+
32 | Username | Password |
33 +-----+-----+
34 |   Rohit   |  abcdef  |
35 +-----+-----+
36 |  Sonia12  | son1234  |
37 +-----+-----+
38 '''
39
40 # Dictionary of elements in {"Atomic number" : "Name"} format
41 # Elements are referenced through out the program through this
42 # dictionary only.
43 elemDict = {
44     "1": "HYDROGEN",
45     "2": "HELIUM",
46     "3": "LITHIUM",
47     "4": "BERYLLIUM",
48     "5": "BORON",
49     "6": "CARBON",
50     "7": "NITROGEN",
51     "8": "OXYGEN",
52     "9": "FLUORINE",
53     "10": "NEON",
54     "11": "SODIUM",
55     "12": "MAGNESIUM",
56     "13": "ALUMINIUM",
57     "14": "SILICON",
58     "15": "PHOSPHORUS",
59     "16": "SULPHUR",
60     "17": "CHLORINE",
61     "18": "ARGON",
62     "19": "POTASSIUM",
63     "20": "CALCIUM",
64     "21": "SCANDIUM",
65     "22": "TITANIUM",
66     "23": "VANADIUM",
67     "24": "CHROMIUM",

```

68	"25": "MANGANESE",
69	"26": "IRON",
70	"27": "COBALT",
71	"28": "NICKEL",
72	"29": "COPPER",
73	"30": "ZINC",
74	"31": "GALLIUM",
75	"32": "GERMANIUM",
76	"33": "ARSENIC",
77	"34": "SELENIUM",
78	"35": "BROMINE",
79	"36": "KRYPTON",
80	"37": "RUBIDIUM",
81	"38": "STRONTIUM",
82	"39": "YTTRIUM",
83	"40": "ZIRCONIUM",
84	"41": "NIOBIUM",
85	"42": "MOLYBDENUM",
86	"43": "TECHNETIUM",
87	"44": "RUTHENIUM",
88	"45": "RHODIUM",
89	"46": "PALLADIUM",
90	"47": "SILVER",
91	"48": "CADMIUM",
92	"49": "INDIUM",
93	"50": "TIN",
94	"51": "ANTIMONY",
95	"52": "TELLURIUM",
96	"53": "IODINE",
97	"54": "XENON",
98	"55": "CAESIUM",
99	"56": "BARIUM",
100	"57": "LANTHANUM",
101	"58": "CERIUM",
102	"59": "PRASEODYMIUM",
103	"60": "NEODYMIUM",
104	"61": "PROMETHIUM",
105	"62": "SAMARIUM",
106	"63": "EUROPIUM",
107	"64": "GADOLINIUM",
108	"65": "TERBIUM",
109	"66": "DYSPROSIUM",
110	"67": "HOLMIUM",
111	"68": "ERBIUM",
112	"69": "THULLIUM",
113	"70": "YTTERBIUM",
114	"71": "LUTETIUM",
115	"72": "HAFNIUM",
116	"73": "TANTALUM",
117	"74": "TUNGSTEN",
118	"75": "RHENIUM",
119	"76": "OSMIUM",
120	"77": "IRIDIUM",
121	"78": "PLATINUM",
122	"79": "GOLD",
123	"80": "MERCURY",
124	"81": "THALIUM",
125	"82": "LEAD",
126	"83": "BISMUTH",
127	"84": "POLONIUM",
128	"85": "ASTATINE",
129	"86": "RADON",
130	"87": "FRANCIUM",
131	"88": "RADIUM",
132	"89": "ACTINIUM",
133	"90": "THORIUM",
134	"91": "PROTACTINIUM",
135	"92": "URANIUM",
136	"93": "NEPTUNIUM",

```

137     "94": "PLUTONIUM",
138     "95": "AMERICIUM",
139     "96": "CURIUM",
140     "97": "BERKELIUM",
141     "98": "CALIFORNIUM",
142     "99": "EINSTEINIUM",
143     "100": "FERMIUM",
144     "101": "MENDELEVIUM",
145     "102": "NOBELIUM",
146     "103": "LAWRENCIUM",
147     "104": "RUTHERFORDIUM",
148     "105": "DUBNIUM",
149     "106": "SEABORGIUM",
150     "107": "BOHRIUM",
151     "108": "HASSIUM",
152     "109": "MEITNERIUM",
153     "110": "DARMSTADTIUM",
154     "111": "ROENTGENIUM",
155     "112": "COPERNICIUM",
156     "113": "NIHONIUM",
157     "114": "FLEROVIUM",
158     "115": "MOSCOVIUM",
159     "116": "LIVERMORIUM",
160     "117": "TENNESSINE",
161     "118": "OGANESSON"
162 }
163
164 def main():
165
166     '''
167     This is the main function that contains complete program.
168     The whole program has been included in a function so as
169     to be able to use it anywhere in the program whenever we need
170     to launch the program from start e.g. in case when the user
171     wish to change the account then we need to launch the signin
172     window again, at that time we can simply call the main()
173     function which will automatically launch the program from start.
174     '''
175
176     # Signin-Window
177     global bg                                     # Global variable to hold
178     bg-img
179
180     window = Tk()                                # Creating window instance
181     window.resizable(False, False)               # Window is not resizable
182     window.iconbitmap('921815.ico')              # Setting icon of the window
183     window.title('Sign In Window')              # Setting title of the window
184     window.geometry('700x350+350+180')           # Setting window size
185     bg = ImageTk.PhotoImage(file='mainbg.jpg')   # Assigning bg-img to the
186     "bg" variable
187
188     NOTE: Here we are using canvas for holding background image
189     so that we can assemble all the different labels onto
190     the background image which is only possible in canvas.
191
192     '''
193     signin_canvas = Canvas(window, width=700, height=350) # Creating canvas to put in
194     the main window
195     signin_canvas.pack(fill='both', expand=True)          # Packing the canvas in the
196     window
197     signin_canvas.create_image(0,0, image=bg, anchor='nw') # "anchor" specifies the
198     position of top left corner of the image
199
200     # "nw" stands for
201     north-west i.e. top left of
202     the window
203
204     # Function to Create new Account (Sign Up)
205     def signup():
206         '''

```

```

199     This function defines the signup window and contains
200     different functions to do the following tasks:
201     1) Take new account's details like username and password
202         and store in the database named "username_pass"
203         # NOTE: The database "username_pass" is already created
204             and is not being created in the program
205     2) Check the strength of the password entered
206     '''
207
208     # Function to check password strength
209     def password_strength(password):
210
211         '''
212         This function checks the strength of the password entered
213         The password must:
214         1) Contain atleast 8 characters
215         2) Contain both letters and numbers
216
217         NOTE: This function returns a statement specifying the condition
218             that is not met by the password entered by the user. This
219             statement will be directly used to display the message
220             on the screen.
221         '''
222
223         # Min length of password = 8
224         if len(password)>=8:
225             # Password should contain both alphabets and digits and symbols
226             if not(password.isalpha()): # Password should not contain only alphabets
227                 if not(password.isdigit()): # Password should not contain only
228                     digits
229                     return 'strong' # Password satisfies all parameters
230             else:
231                 return 'Password must contain alphabets,\n digits and special
232                     symbols'
233         else:
234             # Password length less than 8 characters
235             return 'Password must contain \n atleast 8 characters'
236
237     def new_acc():
238
239         '''
240         This function when called takes the information entered in username
241         and password entry widget of the signup window checks their validity
242         and if valid stores it in the database, displaying message to the user
243         that the account has been created successfully otherwise due to some
244         invalidity it shows a message to the user specifying the problem.
245         '''
246
247         new_username = str(username_enter.get()).strip() # Get the username
248         entered and remove leading and trailing whitespaces
249         new_password = str(password_enter.get()).strip() # Get the password entered
250         and remove leading and trailing whitespaces
251         re_entered_pass = str(re_enter_textbox.get()).strip() # Get the re-entered
252         password
253         try:
254             # If length of the username entered is 0, i.e. Username not entered
255             if len(new_username) == 0:
256                 messagebox.showinfo('Information','Username can't be empty!')
257             # If length of the password entered is 0, i.e. Password not entered
258             elif len(new_password) == 0:
259                 messagebox.showinfo('Information','Password can't be empty!')
260             # If re-entered password doesn't match the previous one
261             elif new_password != re_entered_pass:
262                 messagebox.showinfo('Information','Re-entered password doesn't
263                     match')
264             # If all information entered correctly

```

```

261         else:
262             if password_strength(new_password) == 'strong': # Checking
                strength of the password entered using the "password_strength
                function defined above
263                 # Establishing connection with Mysql
264                 con =
                Connector.connect(host='localhost',user='root',passwd='root',data
                base='username_pass')
265                 # Creating cursor instance
266                 cursor = con.cursor()
267                 # Query to insert new accounts data
268                 query = "INSERT INTO data (Username, Password) VALUES (%s, %s)"
269                 cursor.execute(query,(new_username, new_password))
270                 messagebox.showinfo('Successful','Account created Successfully')
271                 # Saving data permanently
272                 cursor.execute('commit')
273                 # Closing connection
274                 con.close()
275                 # Destroying the previous Signup Window once the Sign Up is
                completed successfully
276                 signup_window.destroy()
277                 # Running the main program once the new account is successfully
                created
278                 # It will again open the signin window
279                 main()
280             else:
281                 messagebox.showinfo('Weak Password\n',
                password_strength(new_password))
282         except:
283             messagebox.showerror('Already Exists','Username already Exists')
284
285     # Window for sign up
286     global back # Global
                variable to hold bg-img
287
288     window.destroy() # Destroying
                the signin window once
289     signup_window = Tk() # Creating new
                window instance for signup
                window
290     signup_window.resizable(False, False) # Window not
                resizable
291     signup_window.iconbitmap('921815.ico') # Setting
                window icon
292     signup_window.title('Sign Up Window') # Setting
                window title
293     signup_window.geometry('700x350+350+180') # Setting
                window dimensions
294
295     back = ImageTk.PhotoImage(file='mainbg.jpg') # Assigning the
                bg-img to the "back" variable
296     '''
297     NOTE: Here we are using canvas for holding background image
298           so that we can assemble all the different labels onto
299           the background image which is only possible in canvas.
300     '''
301     signup_canvas = Canvas(signup_window, width=700, height=350) # Creating
                canvas to hold bg-img
302     signup_canvas.pack(fill='both',expand=True) # Packing the
                canvas into the window
303     signup_canvas.create_image(0,0, image=back, anchor='nw') # Inserting
                image into canvas
304
305     # Label displaying "Sign Up Here" in bold heading
306     signup_canvas.create_text(125, 70, text='Sign Up Here',
                font=('Impact',30,'bold'),fill='#285243')
307
308     # Label displaying "Username:"
309     signup_canvas.create_text(85, 110, text='Username:',

```

```

310     font=('Helvetica',15,'bold'),fill='black')
311     # Username entry widget for user to enter username
312     username_enter = Entry(signup_window, width = 40)
313     # Putting the entry widget up on canvas
314     username_enter_window = signup_canvas.create_window(155, 135, window =
username_enter)
315
316     # Label displaying "Password:"
317     signup_canvas.create_text(85, 175, text='Password:',
font=('Helvetica',15,'bold'),fill='black')
318
319     # Password entry widget for user to enter password
320     password_enter = Entry(signup_window, width = 40, show='*')
321     # Putting the entry widget up on canvas
322     password_enter_window = signup_canvas.create_window(155, 200, window =
password_enter)
323
324     # Label displaying "Re-enter Password:"
325     signup_canvas.create_text(125, 240, text='Re-enter Password:',
font=('Helvetica',15,'bold'),fill='black')
326
327     # Entry widget for user to enter password again
328     re_enter_textbox = Entry(signup_window, width = 40, show='*')
329     # Putting entry widget up on canvas
330     re_enter_textbox_window = signup_canvas.create_window(155, 265, window =
re_enter_textbox)
331
332     # Button to submit info
333     signup_button = Button(signup_window, text='SIGN
UP',command=new_acc,width=30,font=('Helvetica',10,'bold'),bg='#cccccc')
334     signup_button_window = signup_canvas.create_window(155, 315, window =
signup_button)
335
336     # Tooltip for signup button displaying "Create Account"
337     tip = Balloon(signup_window)
338     tip.bind_widget(signup_button, balloonmsg = 'Create Account')
339
340     signup_window.mainloop()
341
342
343     # Function for sign in window
344     def signin():
345
346         '''
347         This function is executed when user press the signin button
348         on the sign in window. It extracts out all the information
349         entered by the user in the entry box, extracts all the data
350         from the data and checks the validity of the information
351         entered; if correct then it launches the splash window
352         otherwise shows a messagebox to the user displaying the error
353         occurred.
354         '''
355
356         # Establishing connection with database
357         con =
Connector.connect(host='localhost',user='root',passwd='root',database='username_p
ass')
358         # Creating cursor instance to interact with the database
359         cursor = con.cursor()
360         # Getting the username and password entered by the user
361         username_entered = str(username_entry_widget.get())
362         password_entered = str(password_entry_widget.get())
363         # Extracting all data of usernames and passwords from DB
364         cursor.execute('select * from data')
365         # Fetching all data from cursor instance
366         all_data = cursor.fetchall()
367         # Dictionary to hold username: password values stored
368         usernames_passwords_dict = {}

```

```

369         for i in all_data: # Iterating through the extracted data which is a list of
tuples
370             usernames_passwords_dict[i[0]]=i[1] # Format is: {username : password}
because in the tuple the first element
371                                     # will be username and the second one
will be the password (see at the
372                                     # starting of the program for reference
373 # Checking if the entered username exists
374 if username_entered in usernames_passwords_dict: # If exists
375     if password_entered == usernames_passwords_dict.get(username_entered): #
Password check
376         splash_window() # Login successfully
377     else: # Wrong password entered
378         messagebox.showerror('Incorrect Password','INCORRECT PASSWORD')
379 else: # Username 'Does Not Exist'
380     messagebox.showerror('Username DNE','Username Does Not Exist \n Retry or
create a new account')
381 # Closing Connection
382 con.close()
383
384 def splash_window():
385     '''
386     This window will pop-up once the login is successful.
387     It will show welcome message and has nothing to do with
388     the actual loding of program, it is there just to make
389     program more realistic and interactive, it will automatically
390     execute login_successful function after 1.5 seconds which in
391     turn will close this splash window.
392     '''
393
394     global background # Global variable to hold
background image # Destroys the previous
396     window.destroy() # Creating new window
397     splash = Tk() # Window not resizable
instance # Setting title of the
398     splash.resizable(False, False) # Setting icon of the window
399     splash.title('Loading...') # Setting screen size
400     splash.iconbitmap('921815.ico')
401     splash.geometry('500x170+400+250')
402
403     background = ImageTk.PhotoImage(file='load.jpg') # Loading bg-img in the
variable
404     splash_canvas = Canvas(splash, width=500, height=170) # Creating canvas
405     splash_canvas.pack(fill='both',expand=True) # Packing canvas in the window
406     splash_canvas.create_image(0,0, image=background, anchor='nw') # Pushing image
into the canvas
407
408     # Creating text labels to be displayed
409     splash_canvas.create_text(250, 60, text='Welcome!',
font=('Helvetica',30,'bold'), fill='white')
410     splash_canvas.create_text(250, 100, text='Loading...',
font=('Helvetica',15,'bold'), fill='white')
411
412     # Creating sound effects
413     pygame.mixer.init()
414     pygame.mixer.music.load('audio/welcome.mp3')
415     pygame.mixer.music.play(loops=0)
416     # Automatically executing login_successful function after 1500 milliseconds i.e.
1.5 seconds
417     # Using lambda function because we need to pass an argument -splash window
itself- into the function
418     splash.after(3000, lambda:login_successful(splash))
419
420 def login_successful(splash):
421     '''
422

```

```

423 This is the function containing whole periodic table and its
424 functionalities, it takes the splash window as it's argument
425 and first destroys it. Then it creates a new window for
426 periodic table and put all the contents in it.
427 '''
428
429 splash.destroy() # Destroying the
previous splash window
430 root = Tk() # Creating main window
431 screen_width = root.winfo_screenwidth() # Getting screen width
of PC
432 screen_height = root.winfo_screenheight() # Getting screen height
of PC
433 root.iconbitmap('921815.ico') # Setting icon
434 root.title("Periodic Table") # Setting title
435 root.geometry(f'{screen_width}x{screen_height}+-10+0') # Setting screen size
and position on screen using
436 # the screen width and
height, using f string
literal
437 # to pass the arguments
of height and width of
screen
438
439 frame = LabelFrame(root, borderwidth=0, highlightthickness=0) # Creating
a frame to put contents
440 frame.grid(row=0,column=0) # Putting frame in the window
441
442
443 # Functions
444 # Move forward in elements
445 def forward(x, top): # Passing current element number and the current top
window as arguments
446
447     '''
448     This function defines the functionality of the forward button on the
elements window.
449     It destroys the current element window and creates a new window and
redefines all the functionality
450     for the next element and then displays the next element information by
reading it from the file of
451     the new element along with the image.
452     '''
453
454     global elemDict # Global dictionary containing names and atomic numbers of
all elements
455     global img # Global variable to hold the element image
456     # Defining the font to be used
457     myFont = Font(family="Helvetica",
458                   size=10,
459                   weight="bold")
460     file = open("Elements\\"+ elemDict[str(x+1)] + ".txt", 'r') # Opening file
of next element to work with
461     top.destroy() # Destroy the window of previous element
462     # Creating and defining new window
463     top = Toplevel() # Creating top level instance
464     top.resizable(False, False) # Window in not resizable
465     tip = Balloon(top) # Tooltip initiated
466     top.title(elemDict[str(x+1)]) # Putting the name of the next element as
title(here x is the atomic number
467                                # of currently displayed element which is
taken as an argument by the function call)
468     top.iconbitmap("921815.ico") # Setting icon of the window
469     img = ImageTk.PhotoImage(Image.open("imgs\\"+ elemDict[str(x+1)] + ".jpg"))
# Opening image of the new element
470     information = file.read() # Reading info from the file
471     # Status bar at bottom
472     # Status bar shows the atomic number of the current element which the user
is seeing

```



```

473         status = Label(top ,text = f"Element {int(x)+1} of 118", bd=1,
474         relief=SUNKEN) # Status bar is sunken a little bit

475     # from the rest of the text
476     status.grid(row=3, column=0, columnspan = 2, sticky = W+E) # sticky
477     attribute spans the status bar across the width
478     # of the entire
479     window(W+E
480     stands for west
481     to east i.e.
482     # entire width
483     of the window)

484     # Lable to carry information of element
485     label1 = Label(top, text=information, justify=LEFT, font=myFont, pady=10)
486     label1.grid(row=0, column=0)
487     # Lable to carry image of the element
488     label2 = Label(top, image=img)
489     label2.grid(row=0, column=1)

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

492     if x == 117: # Checking for last second element
493         # If the current element is last second than for the next element we
494         need to disable the forward button
495         bfor = Button(top, text=">>", command = lambda:forward(int(x)+1,top),
496         justify=RIGHT, state=DISABLED, width=10)
497         tip.bind_widget(bfor, balloonmsg = 'Next')
498     else: # If not the last second element
499         # We need not disable the forward button
500         bfor = Button(top, text=">>", command = lambda:forward(int(x)+1,top),
501         justify=RIGHT, width=10)
502         tip.bind_widget(bfor, balloonmsg = 'Next')

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

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

518     def on_closing():
519         try:
520             pygame.mixer.music.stop()
521         finally:
522             top.destroy()

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

525     def backward(x, top): # Passing current element atomic number and current top
526     window as arguments
527     '''
528     This function defines the functionality of the back button on the elements
529     window.
530     It destroys the current element window and creates a new window and

```

```

528         redefines all the functionality
529         for the previous element and then displays the previous element information
530         by reading it from the file of
531         the new element along with the image.
532         '''
533
534         global elemDict # Global dictionary containing all the elements with
535         atomic numbers
536         global img # Global variable for holding the element image
537         # Defining the font to be used
538         myFont = Font(family="Helvetica",
539                       size=10,
540                       weight="bold")
541         file = open("Elements\\"+ elemDict[str(x-1)] + ".txt", 'r') # Opening
542         previous element file to work with
543         top.destroy() # Destroying the current window which is open
544         # Creating and defining new window
545         top = Toplevel() # Creating new window instance
546         top.resizable(False, False) # Window not resizable
547         tip = Balloon(top) # Tooltip initiated
548         top.title(elemDict[str(x-1)]) # Taking the name of the element from the
549         dictionary using its atomic number
550         # and assigning it as the title of the window
551         top.iconbitmap("921815.ico") # Setting the icon of the window
552         img = ImageTk.PhotoImage(Image.open("imgs\\"+ elemDict[str(x-1)] + ".jpg"))
553         # Opening the image in the variable
554         information = file.read() # Reading information of the previous element
555         from its file
556         # Status bar at bottom
557         # Status bar shows the atomic number of the current element which the user
558         is seeing
559         status = Label(top, text = f"Element {int(x)-1} of 118", bd=1,
560                       relief=SUNKEN) # Status bar is sunken a little bit
561
562         # for it to look different from all
563
564         # other text
565         status.grid(row=3, column=0, columnspan = 2, sticky = W+E) # sticky is
566         used to extend the status bar so as to
567
568         # cover the
569         complete width
570         of the
571         window(W+E stands
572         # for west to
573         east i.e.
574         complete width
575         of the window)
576
577         # Lable to carry information
578         labell1 = Label(top, text=information, justify=LEFT, font=myFont, pady=10)
579         labell1.grid(row=0, column=0)
580         # Lable to carry image
581         label2 = Label(top, image=img)
582         label2.grid(row=0, column=1)
583
584         note_label = Label(top, text = 'Press "esc" to stop audio')
585         note_label.grid(column=0, row=1, columnspan=2)
586
587         # Forward and backward button
588         if x == 2: # If the element is the second element than we need to diable
589         the backward function for first element because there is no more element
590         before it
591         bback = Button(top, text="<<", command = lambda:backward(int(x)-1,top),
592                       justify=LEFT, state=DISABLED, width=10)
593         tip.bind_widget(bback, balloonmsg = 'Back')
594
595         else: # If the current element is not the second element
596         bback = Button(top, text="<<", command = lambda:backward(int(x)-1,top),
597                       justify=LEFT, width=10)
598         tip.bind_widget(bback, balloonmsg = 'Back')

```

```

575
576     # Not imposing any condition on forward button as there will always be
    atleast one element before
577     # the current element as we have pressed the back button atleast once
578     bfor = Button(top, text=">>", command = lambda:forward(int(x)-1,top),
    justify=RIGHT, width=10)
579     tip.bind_widget(bfor, balloonmsg = 'Next')
580     bfor.grid(row = 2, column = 1)
581     bback.grid(row = 2, column = 0)
582     # Flushing and closing file
583     file.flush() # A good practice to flush the file externally
584     file.close()
585
586     # Creating sound effects
587     pygame.mixer.init()
588     pygame.mixer.music.load('audio/'+elemDict[str(int(x)-1)]+'.mp3')
589     pygame.mixer.music.play(loops=0)
590
591     def on_closing():
592         try:
593             pygame.mixer.music.stop()
594             finally:
595                 top.destroy()
596
597     keyboard.add_hotkey('esc', lambda: pygame.mixer.music.stop())
598     top.protocol('WM_DELETE_WINDOW', on_closing)
599
600     def info(x): # Fuction to extract related information and images and put it up
    on different window
601         global elemDict # Dictionary containing all the elements and atomic numbers
602         global img # global variable to hold image
603         # Defining the font to be used
604         myFont = Font(family="Helvetica",
605                       size=10,
606                       weight="bold")
607         file = open("Elements\\"+ elemDict[x] + ".txt", 'r', encoding='utf-8') #
    Opening file to work with
608         # Creating and defining new window
609         top = Toplevel() # Creating new top window to open over the main window
610         top.resizable(False, False) # Window not resizable
611         tip = Balloon(top) # Initialising tooltip
612         top.title(elemDict[x]) # Setting title of window
613         top.iconbitmap("921815.ico") # Setting icon of the window
614         img = ImageTk.PhotoImage(Image.open("imgs\\"+ elemDict[x] + ".jpg")) #
    Opening image in variable
615         information = file.read() # Reading information from file
616
617         # Status bar at bottom
618         # Status bar shows the atomic number of the current element
619         status = Label(top ,text = f"Element {x} of 118", bd=1, relief=SUNKEN) #
    Status bar is sunken a bit inside the
620
621
622     screen so as to differentiate it from
623
624     other text
625         status.grid(row=3, column=0, columnspan = 2, sticky = W+E) # Sticky is
    used so as to extend the status bar over the
626
627
628         # complete
629         window screen
630
631         # Lable to carry information
632         label1 = Label(top, text=information, justify=LEFT, font=myFont, pady=10)
633         label1.grid(row=0, column=0)
634         # Lable to carry image
635         label2 = Label(top, image=img)
636         label2.grid(row=0, column=1)
637         # Flushing and closing file
638         file.flush() # A good practice to flush the file externally
639         # Closing file
640         file.close()

```

```

634
635     note_label = Label(top, text = 'Press "esc" to stop audio')
636     note_label.grid(column=0, row=1, columnspan=2)
637
638     # Forward and backward button
639     if int(x) != 1 and int(x) != 118: # Checking that the element is not the
        first or the last element
640         # If element is neither first or last then we need to enable both
        forward and backward button
641         bfor = Button(top, text=">>", command =lambda: forward(int(x),top),
            justify=RIGHT, width=10)
642         tip.bind_widget(bfor, balloonmsg = 'Next')
643
644         bback = Button(top, text="<<", command =lambda: backward(int(x), top),
            justify=LEFT, width=10)
645         tip.bind_widget(bback, balloonmsg = 'Back')
646
647         bfor.grid(row = 2, column = 1)
648         bback.grid(row = 2, column = 0)
649
650     elif int(x) == 1: # Checking if the element is the first element
651         # If the element is the first element than we need to diable the back
        button since there
652         # is no element before the very first element
653         bfor = Button(top, text=">>", command =lambda: forward(int(x),top),
            justify=RIGHT, width=10)
654         tip.bind_widget(bfor, balloonmsg = 'Next')
655
656         bback = Button(top, text="<<", command =lambda: backward(int(x), top),
            justify=LEFT, state=DISABLED, width=10)
657         tip.bind_widget(bback, balloonmsg = 'Back')
658
659         bfor.grid(row = 2, column = 1)
660         bback.grid(row = 2, column = 0)
661
662     elif int(x) == 118: # Checking if the element is the last element or not
663         # If the element is the last element than we need to disable the
        forward button since there
664         # is no element after the very last element
665         bfor = Button(top, text=">>", command =lambda: forward(int(x),top),
            justify=RIGHT, state=DISABLED, width=10)
666         tip.bind_widget(bfor, balloonmsg = 'Next')
667
668         bback = Button(top, text="<<", command =lambda: backward(int(x), top),
            justify=LEFT, width=10)
669         tip.bind_widget(bback, balloonmsg = 'Back')
670
671         bfor.grid(row = 2, column = 1)
672         bback.grid(row = 2, column = 0)
673
674     # Creating sound effects
675     pygame.mixer.init()
676     pygame.mixer.music.load('audio/'+elemDict[x]+'.mp3')
677     pygame.mixer.music.play(loops=0)
678
679     def on_closing():
680         try:
681             pygame.mixer.music.stop()
682         finally:
683             top.destroy()
684
685     keyboard.add_hotkey('esc', lambda: pygame.mixer.music.stop())
686     top.protocol('WM_DELETE_WINDOW', on_closing)
687
688     def selected(event):
689
690         '''
691         This function is executed when an option from the drop-down box(combo box)
        is selected

```

```

692         It has the options to logout from the current account or to change the
693         current account
694         '''
695     def splash_1():
696         '''
697         This function creates a splash window showing thankyou message and self
698         destroys
699         itself after 1000 milliseconds i.e. 1 second and the program gets
700         terminated
701         '''
702         global thanks_img # Global variable to hold the thanks image
703         splash_2 = Tk() # Creating splash window
704         splash_2.iconbitmap('921815.ico') # Setting window icon
705         splash_2.resizable(False, False) # Window not resizable
706         splash_2.geometry('562x270+450+200') # Setting window size
707         splash_2.title('Thank You!') # Setting window title
708         thanks_img = ImageTk.PhotoImage(file='ty.jpg') # Opening image in
709         variable
710         splash_2_canvas = Canvas(splash_2, width=562, height=280) # Creating
711         canvas
712         splash_2_canvas.pack(fill='both', expand=True) # Packing canvas in the
713         window
714         splash_2_canvas.create_image(0,0, image=thanks_img, anchor='nw') #
715         Putting image into canvas
716
717         splash_2.after(1000, lambda:splash_2.destroy()) # Function to
718         automatically destroy the splash window
719
720     selected = drop.get() # Extracting the selected option from drop down menu
721     if selected == 'Options':
722         pass
723     elif selected == 'Logout': # If logout option selected
724         # Asking the user to confirm that they really want to logout
725         reply = messagebox.askquestion('Logout','Are you sure, \nYou want to
726         Logout?')
727         if reply == 'yes': # If they reply 'yes'
728             root.destroy() # Destroy the root window
729             splash_1() # Executing splash_1 window which contains the thankyou
730             message
731     elif selected == 'Change Account': # If user selected the change account
732     option
733         root.destroy() # Destroy the root window
734         main() # Call the main function which executes the program again from
735         signin window
736
737
738     # NOTE label
739     noteLabel = Label(frame, text="NOTE : Click on the elements\nto know about
740     them", justify=CENTER, font=('Helvetica',10))
741     noteLabel.grid(row=0, column=6, columnspan=4)
742
743     # Copyright Label
744     copy_right_Label = Label(frame, text="\u00a9"+" 2020 Vinit Mehta",
745     justify=CENTER, font=('Helvetica',15))
746     copy_right_Label.grid(row=1, column=6, columnspan=4)
747
748     # Drop-Down Menu(Combo Box)
749     options = ['Options','Logout', 'Change Account'] # List containing the list of
750     option to be displayed in drop-down menu
751     drop = ttk.Combobox(frame,value=options,width=20) # Creating combo box
752     drop.grid(row=0, column=14, columnspan=2) # Putting combo box up on screen
753     drop.current(0) # Setting the default value to be displayed in combo box
754     drop.bind('<<ComboboxSelected>>', selected) # Action to be taken when some
755     option is selected

```

```

745 # Creating all 118 elements button groupwise
746 # Two step process
747 # First defining it using Button method
748 # Second putting it up on the screen using grid method
749
750 # Group 1
751
752 b1 = Button(frame, text="1\nH\nHydrogen\n1.0", padx=2, bg="white",
753             command=lambda: info("1"))
754 b1.grid(row=0, column=0)
755 b3 = Button(frame, text="3\nLi\nLithium\n6.9", padx=8, bg="yellow",
756             command=lambda: info("3"))
757 b3.grid(row=1, column=0)
758 b11 = Button(frame, text="11\nNa\nSodium\n23.0", padx=8, bg="yellow",
759             command=lambda: info("11"))
760 b11.grid(row=2, column=0)
761 b19 = Button(frame, text="19\nK\nPotassium\n39.1", bg="yellow", command=lambda:
762             info("19"))
763 b19.grid(row=3, column=0)
764 b37 = Button(frame, text="37\nRb\nRubidium\n85.5", padx=2, bg="yellow",
765             command=lambda: info("37"))
766 b37.grid(row=4, column=0)
767 b55 = Button(frame, text="55\nCs\nCesium\n132.9", padx=8, bg="yellow",
768             command=lambda: info("55"))
769 b55.grid(row=5, column=0)
770 b87 = Button(frame, text="87\nFr\nFrancium\n223.0", padx=4, bg="yellow",
771             command=lambda: info("87"))
772 b87.grid(row=6, column=0)
773
774 # Group 2
775
776 b4 = Button(frame, text="4\nBe\nBeryllium\n9.0", padx=8, bg="#c92a6e",
777             command=lambda: info("4"))
778 b4.grid(row=1, column=1)
779 b12 = Button(frame, text="12\nMg\nMagnesium\n24.3", bg="#c92a6e",
780             command=lambda: info("12"))
781 b12.grid(row=2, column=1)
782 b20 = Button(frame, text="20\nCa\nCalcium\n40.1", padx=10, bg="#c92a6e",
783             command=lambda: info("20"))
784 b20.grid(row=3, column=1)
785 b38 = Button(frame, text="38\nSr\nStrontium\n87.6", padx=6, bg="#c92a6e",
786             command=lambda: info("38"))
787 b38.grid(row=4, column=1)
788 b56 = Button(frame, text="56\nBa\nBarium\n137.3", padx=13, bg="#c92a6e",
789             command=lambda: info("56"))
790 b56.grid(row=5, column=1)
791 b88 = Button(frame, text="88\nRa\nRadium\n226.0", padx=12, bg="#c92a6e",
792             command=lambda: info("88"))
793 b88.grid(row=6, column=1)
794
795 # Group 3
796
797 b21 = Button(frame, text="21\nSc\nScandium\n45.0", padx=5, bg="#ffd900",
798             command=lambda: info("21"))
799 b21.grid(row=3, column=2)
800 b39 = Button(frame, text="39\nY\nYttrium\n88.9", padx=12, bg="#ffd900",
801             command=lambda: info("39"))
802 b39.grid(row=4, column=2)
803 b57 = Button(frame, text="57\nLa*\nLanthanum\n138.9", padx=2, bg="#ff5500",
804             command=lambda: info("57"))
805 b57.grid(row=5, column=2)
806 b89 = Button(frame, text="89\nAc**\nActinium\n227.0", padx=8, bg="#f7ff5e",
807             command=lambda: info("89"))
808 b89.grid(row=6, column=2)
809
810 # Group 4
811
812 b22 = Button(frame, text="22\nTi\nTitanium\n47.9", padx=16, bg="#ffd900",
813             command=lambda: info("22"))

```

```

796         b22.grid(row=3,column=3)
797         b40 = Button(frame, text="40\nZr\nZirconium\n91.2", padx=12, bg="#ffd900",
798         command=lambda: info("40"))
799         b40.grid(row=4,column=3)
800         b72 = Button(frame, text="72\nHf\nHafnium\n178.5", padx=16, bg="#ffd900",
801         command=lambda: info("72"))
802         b72.grid(row=5,column=3)
803         b104 = Button(frame, text="104\nRf\nRutherfordium\n261", padx=0, bg="#ffd900",
804         command=lambda: info("104"))
805         b104.grid(row=6,column=3)
806
807         # Group 5
808
809         b23 = Button(frame, text="23\nV\nVanadium\n50.9", padx=12, bg="#ffd900",
810         command=lambda: info("23"))
811         b23.grid(row=3,column=4)
812         b41 = Button(frame, text="41\nNb\nNiobium\n92.9", padx=16, bg="#ffd900",
813         command=lambda: info("41"))
814         b41.grid(row=4,column=4)
815         b73 = Button(frame, text="73\nTa\nTantalum\n180.9", padx=14, bg="#ffd900",
816         command=lambda: info("73"))
817         b73.grid(row=5,column=4)
818         b105 = Button(frame, text="105\nDb\nDubnium\n262", padx=15, bg="#ffd900",
819         command=lambda: info("105"))
820         b105.grid(row=6,column=4)
821
822         # Group 6
823
824         b24 = Button(frame, text="24\nCr\nChromium\n52.0", padx=10, bg="#ffd900",
825         command=lambda: info("24"))
826         b24.grid(row=3,column=5)
827         b42 = Button(frame, text="42\nMo\nMolybdenum\n95.9", padx=3, bg="#ffd900",
828         command=lambda: info("42"))
829         b42.grid(row=4,column=5)
830         b74 = Button(frame, text="74\nW\nTungsten\n183.9", padx=14, bg="#ffd900",
831         command=lambda: info("74"))
832         b74.grid(row=5,column=5)
833         b106 = Button(frame, text="106\nSg\nSeaborgium\n263", padx=7, bg="#ffd900",
834         command=lambda: info("106"))
835         b106.grid(row=6,column=5)
836
837         # Group 7
838
839         b25 = Button(frame, text="25\nMn\nManganese\n54.9", padx=1, bg="#ffd900",
840         command=lambda: info("25"))
841         b25.grid(row=3,column=6)
842         b43 = Button(frame, text="43\nTc\nTechnetium\n98", padx=0, bg="#ffd900",
843         command=lambda: info("43"))
844         b43.grid(row=4,column=6)
845         b75 = Button(frame, text="75\nRe\nRhenium\n186.2", padx=8, bg="#ffd900",
846         command=lambda: info("75"))
847         b75.grid(row=5,column=6)
848         b107 = Button(frame, text="107\nBh\nBohrium\n262", padx=9, bg="#ffd900",
849         command=lambda: info("107"))
850         b107.grid(row=6,column=6)
851
852         # Group 8
853
854         b26 = Button(frame, text="26\nFe\nIron\n55.9", padx=19, bg="#ffd900",
855         command=lambda: info("26"))
856         b26.grid(row=3,column=7)
857         b44 = Button(frame, text="44\nRu\nRuthenium\n101.0", padx=0, bg="#ffd900",
858         command=lambda: info("44"))
859         b44.grid(row=4,column=7)
860         b76 = Button(frame, text="76\nOs\nOsmium\n190.2", padx=7, bg="#ffd900",
861         command=lambda: info("76"))
862         b76.grid(row=5,column=7)
863         b108 = Button(frame, text="108\nHs\nHassium\n264", padx=7, bg="#ffd900",
864         command=lambda: info("108"))

```

```

846         b108.grid(row=6,column=7)
847
848     # Group 9
849
850     b27 = Button(frame, text="27\nCo\nCobalt\n58.9", padx=13, bg="#ffd900",
851                  command=lambda: info("27"))
852     b27.grid(row=3,column=8)
853     b45 = Button(frame, text="45\nRh\nRhodium\n102.9", padx=6, bg="#ffd900",
854                  command=lambda: info("45"))
855     b45.grid(row=4,column=8)
856     b77 = Button(frame, text="77\nIr\nIridium\n196.9", padx=12, bg="#ffd900",
857                  command=lambda: info("77"))
858     b77.grid(row=5,column=8)
859     b109 = Button(frame, text="109\nMt\nMeitnerium\n268", padx=0, bg="#ffd900",
860                   command=lambda: info("109"))
861     b109.grid(row=6,column=8)
862
863     # Group 10
864
865     b28 = Button(frame, text="28\nNi\nNickel\n58.7", padx=21, bg="#ffd900",
866                  command=lambda: info("28"))
867     b28.grid(row=3,column=9)
868     b46 = Button(frame, text="46\nPd\nPalladium\n106.4", padx=11, bg="#ffd900",
869                  command=lambda: info("46"))
870     b46.grid(row=4,column=9)
871     b78 = Button(frame, text="78\nPt\nPlatinum\n192.2", padx=14, bg="#ffd900",
872                  command=lambda: info("78"))
873     b78.grid(row=5,column=9)
874     b110 = Button(frame, text="110\nDs\nDarmstadtium\n261.9", padx=0, bg="#ffd900",
875                   command=lambda: info("110"))
876     b110.grid(row=6,column=9)
877
878     # Group 11
879
880     b29 = Button(frame, text="29\nCu\nCopper\n63.5", padx=16, bg="#ffd900",
881                  command=lambda: info("29"))
882     b29.grid(row=3,column=10)
883     b47 = Button(frame, text="47\nAg\nSilver\n107.9", padx=22, bg="#ffd900",
884                  command=lambda: info("47"))
885     b47.grid(row=4,column=10)
886     b79 = Button(frame, text="79\nAu\nGold\n195", padx=23, bg="#ffd900",
887                  command=lambda: info("79"))
888     b79.grid(row=5,column=10)
889     b111 = Button(frame, text="111\nRg\nRoentgenium\n271.8", padx=0, bg="#ffd900",
890                   command=lambda: info("111"))
891     b111.grid(row=6,column=10)
892
893     # Group 12
894
895     b30 = Button(frame, text="30\nZn\nZinc\n65.4", padx=23, bg="#ffd900",
896                  command=lambda: info("30"))
897     b30.grid(row=3,column=11)
898     b48 = Button(frame, text="48\nCd\nCadmium\n112.4", padx=8, bg="#ffd900",
899                  command=lambda: info("48"))
900     b48.grid(row=4,column=11)
901     b80 = Button(frame, text="80\nHg\nMercury\n200.6", padx=12, bg="#ffd900",
902                  command=lambda: info("80"))
903     b80.grid(row=5,column=11)
904     b112 = Button(frame, text="112\nCn\nCopernicium\n285", padx=0, bg="#ffd900",
905                   command=lambda: info("112"))
906     b112.grid(row=6,column=11)
907
908     # Group 13
909
910     b5 = Button(frame, text="5\nB\nBoron\n9.0", padx=15, bg="pink", command=lambda:
911                  info("5"))
912     b5.grid(row=1,column=12)
913     b13 = Button(frame, text="13\nAl\nAluminium\n24.3", padx=1, bg="#352f9e",
914                  command=lambda: info("13"))

```



```

897     b13.grid(row=2,column=12)
898     b31 = Button(frame, text="31\nGa\nGallium\n40.1", padx=11, bg="#352f9e",
899     command=lambda: info("31"))
900     b31.grid(row=3,column=12)
901     b49 = Button(frame, text="49\nIn\nIndium\n87.6", padx=13, bg="#352f9e",
902     command=lambda: info("49"))
903     b49.grid(row=4,column=12)
904     b81 = Button(frame, text="81\nTl\nThallium\n137.3", padx=10, bg="#352f9e",
905     command=lambda: info("81"))
906     b81.grid(row=5,column=12)
907     b113 = Button(frame, text="113\nNh\nNihonium\n286", padx=5, bg="#ffd900",
908     command=lambda: info("113"))
909     b113.grid(row=6,column=12)
910
911     # Group 14
912
913     b6 = Button(frame, text="6\nC\nCarbon\n12.0", padx=12, bg="green",
914     command=lambda: info("6"))
915     b6.grid(row=1,column=13)
916     b14 = Button(frame, text="14\nSi\nSilicon\n28.1",padx=14, bg="pink",
917     command=lambda: info("14"))
918     b14.grid(row=2,column=13)
919     b32 = Button(frame, text="32\nGe\nGermanium\n72.6", padx=0, bg="pink",
920     command=lambda: info("32"))
921     b32.grid(row=3,column=13)
922     b50 = Button(frame, text="50\nSn\nTin\n118.7", padx=18, bg="#352f9e",
923     command=lambda: info("50"))
924     b50.grid(row=4,column=13)
925     b82 = Button(frame, text="82\nPb\nLead\n207.2", padx=18, bg="#352f9e",
926     command=lambda: info("82"))
927     b82.grid(row=5,column=13)
928     b114 = Button(frame, text="114\nFl\nFlerovium\n289", padx=5, bg="#ffd900",
929     command=lambda: info("114"))
930     b114.grid(row=6,column=13)
931
932     # Group 15
933
934     b7 = Button(frame, text="7\nN\nNitrogen\n14.0", padx=13, bg="green",
935     command=lambda: info("7"))
936     b7.grid(row=1,column=14)
937     b15 = Button(frame, text="15\nP\nPhosphorus\n31.0",padx=5, bg="green",
938     command=lambda: info("15"))
939     b15.grid(row=2,column=14)
940     b33 = Button(frame, text="33\nAs\nArsenic\n74.9", padx=17, bg="pink",
941     command=lambda: info("33"))
942     b33.grid(row=3,column=14)
943     b51 = Button(frame, text="51\nSb\nAntimony\n121.8", padx=10, bg="pink",
944     command=lambda: info("51"))
945     b51.grid(row=4,column=14)
946     b83 = Button(frame, text="83\nBi\nBismuth\n209.0", padx=14, bg="#352f9e",
947     command=lambda: info("83"))
948     b83.grid(row=5,column=14)
949     b115 = Button(frame, text="115\nMc\nMoscovium\n288", padx=5, bg="#ffd900",
950     command=lambda: info("115"))
951     b115.grid(row=6,column=14)
952
953     # Group 16
954
955     b8 = Button(frame, text="8\nO\nOxygen\n16.0", padx=13, bg="green",
956     command=lambda: info("8"))
957     b8.grid(row=1,column=15)
958     b16 = Button(frame, text="16\nS\nSulphur\n32.1",padx=13, bg="green",
959     command=lambda: info("16"))
960     b16.grid(row=2,column=15)
961     b34 = Button(frame, text="34\nSe\nSelenium\n79.0", padx=10, bg="green",
962     command=lambda: info("34"))
963     b34.grid(row=3,column=15)
964     b52 = Button(frame, text="52\nTe\nTellurium\n127.6", padx=8, bg="pink",
965     command=lambda: info("52"))

```

```

946     b52.grid(row=4,column=15)
947     b84 = Button(frame, text="84\nPo\nPolonium\n209.0", padx=8, bg="pink",
948     command=lambda: info("84"))
949     b84.grid(row=5,column=15)
950     b116 = Button(frame, text="116\nLv\nLivermorium\n292", padx=0, bg="#ffd900",
951     command=lambda: info("116"))
952     b116.grid(row=6,column=15)
953
954     # Group 17
955     b9 = Button(frame, text="9\nF\nFluorine\n19.0", padx=11, bg="green",
956     command=lambda: info("9"))
957     b9.grid(row=1,column=16)
958     b17 = Button(frame, text="17\nCl\nChlorine\n35.5",padx=10, bg="green",
959     command=lambda: info("17"))
960     b17.grid(row=2,column=16)
961     b35 = Button(frame, text="35\nBr\nBromine\n79.9", padx=10, bg="green",
962     command=lambda: info("35"))
963     b35.grid(row=3,column=16)
964     b53 = Button(frame, text="53\nI\nIodine\n126.9", padx=16, bg="green",
965     command=lambda: info("53"))
966     b53.grid(row=4,column=16)
967     b85 = Button(frame, text="85\nAt\nAstatine\n210", padx=11, bg="green",
968     command=lambda: info("85"))
969     b85.grid(row=5,column=16)
970     b117 = Button(frame, text="117\nTs\nTennessine\n294", padx=3, bg="#ffd900",
971     command=lambda: info("117"))
972     b117.grid(row=6,column=16)
973
974     # Group 18
975     b2 = Button(frame, text="2\nHe\nHelium\n2.0", padx=10, bg="#9943ab",
976     command=lambda: info("2"))
977     b2.grid(row=0,column=17)
978     b10 = Button(frame, text="10\nNe\nNeon\n20.2", padx=15, bg="#9943ab",
979     command=lambda: info("10"))
980     b10.grid(row=1,column=17)
981     b18 = Button(frame, text="18\nAr\nArgon\n40.0", padx=13, bg="#9943ab",
982     command=lambda: info("18"))
983     b18.grid(row=2,column=17)
984     b36 = Button(frame, text="36\nKr\nKrypton\n83.8", padx=8, bg="#9943ab",
985     command=lambda: info("36"))
986     b36.grid(row=3,column=17)
987     b54 = Button(frame, text="54\nXe\nXenon\n131.3", padx=12, bg="#9943ab",
988     command=lambda: info("54"))
989     b54.grid(row=4,column=17)
990     b86 = Button(frame, text="86\nRn\nRadon\n222", padx=12, bg="#9943ab",
991     command=lambda: info("86"))
992     b86.grid(row=5,column=17)
993     b118 = Button(frame, text="118\nOg\nOganesson\n294", padx=0, bg="#ffd900",
994     command=lambda: info("118"))
995     b118.grid(row=6,column=17)
996
997     # Empty label
998     label1 = Label(frame, text="\n\n")
999     label1.grid(row=7, column=0, columnspan=18)
1000
1001     # Lanthanides
1002     bL = Label(frame, text="*Lanthanides", justify=RIGHT)
1003     bL.grid(row=8, column=2)
1004     b58 = Button(frame, text="58\nCe\nCerium\n140.1", padx=20, bg="#ff5500",
1005     command=lambda: info("58"))
1006     b58.grid(row=8, column=3)
1007     b59 = Button(frame, text="59\nPr\nPraseodymium\n140.9", padx=0, bg="#ff5500",
1008     command=lambda: info("59"))
1009     b59.grid(row=8, column=4)
1010     b60 = Button(frame, text="60\nNd\nNeodymium\n144.2", padx=4, bg="#ff5500",

```

```

1000 command=lambda: info("60"))
1001 b60.grid(row=8, column=5)
1002 999 b61 = Button(frame, text="61\nPm\nPromethium\n145", padx=0, bg="#ff5500",
1003 command=lambda: info("61"))
1004 b61.grid(row=8, column=6)
1005 1000 b62 = Button(frame, text="62\nSm\nSamarium\n150.4", padx=2, bg="#ff5500",
1006 command=lambda: info("62"))
1007 1001 b62.grid(row=8, column=7)
1008 1002 b63 = Button(frame, text="63\nEu\nEuropium\n152", padx=4, bg="#ff5500",
1009 command=lambda: info("63"))
1010 1003 b63.grid(row=8, column=8)
1011 1004 b64 = Button(frame, text="64\nGd\nGadolinium\n157.3", padx=7, bg="#ff5500",
1012 command=lambda: info("64"))
1013 1005 b64.grid(row=8, column=9)
1014 1006 b65 = Button(frame, text="65\nTb\nTerbium\n158.9", padx=13, bg="#ff5500",
1015 command=lambda: info("65"))
1016 1007 b65.grid(row=8, column=10)
1017 1008 b66 = Button(frame, text="66\nDy\nDysprosium\n162.5", padx=2, bg="#ff5500",
1018 command=lambda: info("66"))
1019 1009 b66.grid(row=8, column=11)
1020 1010 b67 = Button(frame, text="67\nHo\nHolmium\n164.9", padx=6, bg="#ff5500",
1021 command=lambda: info("67"))
1022 1011 b67.grid(row=8, column=12)
1023 1012 b68 = Button(frame, text="68\nEr\nErbium\n167.3", padx=12, bg="#ff5500",
1024 command=lambda: info("68"))
1025 1013 b68.grid(row=8, column=13)
1026 1014 b69 = Button(frame, text="69\nTm\nThullium\n168.9", padx=12, bg="#ff5500",
1027 command=lambda: info("69"))
1028 1015 b69.grid(row=8, column=14)
1029 1016 b70 = Button(frame, text="70\nYb\nYtterbium\n173", padx=7, bg="#ff5500",
1030 command=lambda: info("70"))
1031 1017 b70.grid(row=8, column=15)
1032 1018 b71 = Button(frame, text="71\nLu\nLutetium\n175", padx=8, bg="#ff5500",
1033 command=lambda: info("71"))
1034 1019 b71.grid(row=8, column=16)
1035 1020
1036 1021 # Actinides
1037 1022
1038 1023 bA = Label(frame, text="**Actinides", justify=RIGHT)
1039 1024 bA.grid(row=9, column=2)
1040 1025 b90 = Button(frame, text="90\nTh\nThorium\n232.0", padx=16, bg="#f7ff5e",
1041 command=lambda: info("90"))
1042 1026 b90.grid(row=9, column=3)
1043 1027 b91 = Button(frame, text="91\nPa\nProtactinium\n231", padx=6, bg="#f7ff5e",
1044 command=lambda: info("91"))
1045 1028 b91.grid(row=9, column=4)
1046 1029 b92 = Button(frame, text="92\nU\nUranium\n238.0", padx=14, bg="#f7ff5e",
1047 command=lambda: info("92"))
1048 1030 b92.grid(row=9, column=5)
1049 1031 b93 = Button(frame, text="93\nNp\nNeptunium\n237", padx=2, bg="#f7ff5e",
1050 command=lambda: info("93"))
1051 1032 b93.grid(row=9, column=6)
1052 1033 b94 = Button(frame, text="94\nPu\nPlutonium\n244", padx=1, bg="#f7ff5e",
1053 command=lambda: info("94"))
1054 1034 b94.grid(row=9, column=7)
1055 1035 b95 = Button(frame, text="95\nAm\nAmericium\n243", padx=1, bg="#f7ff5e",
1056 command=lambda: info("95"))
1057 1036 b95.grid(row=9, column=8)
1058 1037 b96 = Button(frame, text="96\nCm\nCurium\n247", padx=17, bg="#f7ff5e",
1059 command=lambda: info("96"))
1060 1038 b96.grid(row=9, column=9)
1061 1039 b97 = Button(frame, text="97\nBk\nBerkelium\n247", padx=10, bg="#f7ff5e",
1062 command=lambda: info("97"))
1063 1040 b97.grid(row=9, column=10)
1064 1041 b98 = Button(frame, text="98\nCf\nCalifornium\n251", padx=2, bg="#f7ff5e",
1065 command=lambda: info("98"))
1066 1042 b98.grid(row=9, column=11)
1067 1043 b99 = Button(frame, text="99\nEs\nEinsteinium\n252", padx=2, bg="#f7ff5e",
1068 command=lambda: info("99"))
1069 1044

```

```

1045     b99.grid(row=9, column=12)
1046     b100 = Button(frame, text="100\nFm\nFermium\n257", padx=7, bg="#f7ff5e",
1047                    command=lambda: info("100"))
1048     b100.grid(row=9, column=13)
1049     b101 = Button(frame, text="101\nMd\nMendelevium\n258", padx=0, bg="#f7ff5e",
1050                    command=lambda: info("101"))
1051     b101.grid(row=9, column=14)
1052     b102 = Button(frame, text="102\nNo\nNobelium\n259", padx=6, bg="#f7ff5e",
1053                    command=lambda: info("102"))
1054     b102.grid(row=9, column=15)
1055     b103 = Button(frame, text="103\nLr\nLawrencium\n262", padx=0, bg="#f7ff5e",
1056                    command=lambda: info("103"))
1057     b103.grid(row=9, column=16)
1058
1059     # Initiate tooltip
1060     tip = Balloon(root)
1061
1062     # Bind tooltip to buttons
1063     tip.bind_widget(b1, balloonmsg = elemDict["1"])
1064     tip.bind_widget(b2, balloonmsg = elemDict["2"])
1065     tip.bind_widget(b3, balloonmsg = elemDict["3"])
1066     tip.bind_widget(b4, balloonmsg = elemDict["4"])
1067     tip.bind_widget(b5, balloonmsg = elemDict["5"])
1068     tip.bind_widget(b6, balloonmsg = elemDict["6"])
1069     tip.bind_widget(b7, balloonmsg = elemDict["7"])
1070     tip.bind_widget(b8, balloonmsg = elemDict["8"])
1071     tip.bind_widget(b9, balloonmsg = elemDict["9"])
1072     tip.bind_widget(b10, balloonmsg = elemDict["10"])
1073     tip.bind_widget(b11, balloonmsg = elemDict["11"])
1074     tip.bind_widget(b12, balloonmsg = elemDict["12"])
1075     tip.bind_widget(b13, balloonmsg = elemDict["13"])
1076     tip.bind_widget(b14, balloonmsg = elemDict["14"])
1077     tip.bind_widget(b15, balloonmsg = elemDict["15"])
1078     tip.bind_widget(b16, balloonmsg = elemDict["16"])
1079     tip.bind_widget(b17, balloonmsg = elemDict["17"])
1080     tip.bind_widget(b18, balloonmsg = elemDict["18"])
1081     tip.bind_widget(b19, balloonmsg = elemDict["19"])
1082     tip.bind_widget(b20, balloonmsg = elemDict["20"])
1083     tip.bind_widget(b21, balloonmsg = elemDict["21"])
1084     tip.bind_widget(b22, balloonmsg = elemDict["22"])
1085     tip.bind_widget(b23, balloonmsg = elemDict["23"])
1086     tip.bind_widget(b24, balloonmsg = elemDict["24"])
1087     tip.bind_widget(b25, balloonmsg = elemDict["25"])
1088     tip.bind_widget(b26, balloonmsg = elemDict["26"])
1089     tip.bind_widget(b27, balloonmsg = elemDict["27"])
1090     tip.bind_widget(b28, balloonmsg = elemDict["28"])
1091     tip.bind_widget(b29, balloonmsg = elemDict["29"])
1092     tip.bind_widget(b30, balloonmsg = elemDict["30"])
1093     tip.bind_widget(b31, balloonmsg = elemDict["31"])
1094     tip.bind_widget(b32, balloonmsg = elemDict["32"])
1095     tip.bind_widget(b33, balloonmsg = elemDict["33"])
1096     tip.bind_widget(b34, balloonmsg = elemDict["34"])
1097     tip.bind_widget(b35, balloonmsg = elemDict["35"])
1098     tip.bind_widget(b36, balloonmsg = elemDict["36"])
1099     tip.bind_widget(b37, balloonmsg = elemDict["37"])
1100     tip.bind_widget(b38, balloonmsg = elemDict["38"])
1101     tip.bind_widget(b39, balloonmsg = elemDict["39"])
1102     tip.bind_widget(b40, balloonmsg = elemDict["40"])
1103     tip.bind_widget(b41, balloonmsg = elemDict["41"])
1104     tip.bind_widget(b42, balloonmsg = elemDict["42"])
1105     tip.bind_widget(b43, balloonmsg = elemDict["43"])
1106     tip.bind_widget(b44, balloonmsg = elemDict["44"])
1107     tip.bind_widget(b45, balloonmsg = elemDict["45"])
1108     tip.bind_widget(b46, balloonmsg = elemDict["46"])
1109     tip.bind_widget(b47, balloonmsg = elemDict["47"])
1110     tip.bind_widget(b48, balloonmsg = elemDict["48"])
1111     tip.bind_widget(b49, balloonmsg = elemDict["49"])
1112     tip.bind_widget(b50, balloonmsg = elemDict["50"])
1113     tip.bind_widget(b51, balloonmsg = elemDict["51"])

```

```
1110 tip.bind_widget(b52, balloonmsg = elemDict["52"])
1111 tip.bind_widget(b53, balloonmsg = elemDict["53"])
1112 tip.bind_widget(b54, balloonmsg = elemDict["54"])
1113 tip.bind_widget(b55, balloonmsg = elemDict["55"])
1114 tip.bind_widget(b56, balloonmsg = elemDict["56"])
1115 tip.bind_widget(b57, balloonmsg = elemDict["57"])
1116 tip.bind_widget(b58, balloonmsg = elemDict["58"])
1117 tip.bind_widget(b59, balloonmsg = elemDict["59"])
1118 tip.bind_widget(b60, balloonmsg = elemDict["60"])
1119 tip.bind_widget(b61, balloonmsg = elemDict["61"])
1120 tip.bind_widget(b62, balloonmsg = elemDict["62"])
1121 tip.bind_widget(b63, balloonmsg = elemDict["63"])
1122 tip.bind_widget(b64, balloonmsg = elemDict["64"])
1123 tip.bind_widget(b65, balloonmsg = elemDict["65"])
1124 tip.bind_widget(b66, balloonmsg = elemDict["66"])
1125 tip.bind_widget(b67, balloonmsg = elemDict["67"])
1126 tip.bind_widget(b68, balloonmsg = elemDict["68"])
1127 tip.bind_widget(b69, balloonmsg = elemDict["69"])
1128 tip.bind_widget(b70, balloonmsg = elemDict["70"])
1129 tip.bind_widget(b71, balloonmsg = elemDict["71"])
1130 tip.bind_widget(b72, balloonmsg = elemDict["72"])
1131 tip.bind_widget(b73, balloonmsg = elemDict["73"])
1132 tip.bind_widget(b74, balloonmsg = elemDict["74"])
1133 tip.bind_widget(b75, balloonmsg = elemDict["75"])
1134 tip.bind_widget(b76, balloonmsg = elemDict["76"])
1135 tip.bind_widget(b77, balloonmsg = elemDict["77"])
1136 tip.bind_widget(b78, balloonmsg = elemDict["78"])
1137 tip.bind_widget(b79, balloonmsg = elemDict["79"])
1138 tip.bind_widget(b80, balloonmsg = elemDict["80"])
1139 tip.bind_widget(b81, balloonmsg = elemDict["81"])
1140 tip.bind_widget(b82, balloonmsg = elemDict["82"])
1141 tip.bind_widget(b83, balloonmsg = elemDict["83"])
1142 tip.bind_widget(b84, balloonmsg = elemDict["84"])
1143 tip.bind_widget(b85, balloonmsg = elemDict["85"])
1144 tip.bind_widget(b86, balloonmsg = elemDict["86"])
1145 tip.bind_widget(b87, balloonmsg = elemDict["87"])
1146 tip.bind_widget(b88, balloonmsg = elemDict["88"])
1147 tip.bind_widget(b89, balloonmsg = elemDict["89"])
1148 tip.bind_widget(b90, balloonmsg = elemDict["90"])
1149 tip.bind_widget(b91, balloonmsg = elemDict["91"])
1150 tip.bind_widget(b92, balloonmsg = elemDict["92"])
1151 tip.bind_widget(b93, balloonmsg = elemDict["93"])
1152 tip.bind_widget(b94, balloonmsg = elemDict["94"])
1153 tip.bind_widget(b95, balloonmsg = elemDict["95"])
1154 tip.bind_widget(b96, balloonmsg = elemDict["96"])
1155 tip.bind_widget(b97, balloonmsg = elemDict["97"])
1156 tip.bind_widget(b98, balloonmsg = elemDict["98"])
1157 tip.bind_widget(b99, balloonmsg = elemDict["99"])
1158 tip.bind_widget(b100, balloonmsg = elemDict["100"])
1159 tip.bind_widget(b101, balloonmsg = elemDict["101"])
1160 tip.bind_widget(b102, balloonmsg = elemDict["102"])
1161 tip.bind_widget(b103, balloonmsg = elemDict["103"])
1162 tip.bind_widget(b104, balloonmsg = elemDict["104"])
1163 tip.bind_widget(b105, balloonmsg = elemDict["105"])
1164 tip.bind_widget(b106, balloonmsg = elemDict["106"])
1165 tip.bind_widget(b107, balloonmsg = elemDict["107"])
1166 tip.bind_widget(b108, balloonmsg = elemDict["108"])
1167 tip.bind_widget(b109, balloonmsg = elemDict["109"])
1168 tip.bind_widget(b110, balloonmsg = elemDict["110"])
1169 tip.bind_widget(b111, balloonmsg = elemDict["111"])
1170 tip.bind_widget(b112, balloonmsg = elemDict["112"])
1171 tip.bind_widget(b113, balloonmsg = elemDict["113"])
1172 tip.bind_widget(b114, balloonmsg = elemDict["114"])
1173 tip.bind_widget(b115, balloonmsg = elemDict["115"])
1174 tip.bind_widget(b116, balloonmsg = elemDict["116"])
1175 tip.bind_widget(b117, balloonmsg = elemDict["117"])
1176 tip.bind_widget(b118, balloonmsg = elemDict["118"])
1177
1178 # Defining hover over functions for all element buttons
```

```
1179         # Buttons change color when mouse tip move over them
1180
1181     def b_1hover(event):
1182         b1['bg'] = '#d9d3d2'
1183     def b_2hover(event):
1184         b2['bg'] = '#773585'
1185     def b_3hover(event):
1186         b3['bg'] = '#e0dd04'
1187     def b_4hover(event):
1188         b4['bg'] = '#a3295d'
1189     def b_5hover(event):
1190         b5['bg'] = '#f7a1e3'
1191     def b_6hover(event):
1192         b6['bg'] = '#15591f'
1193     def b_7hover(event):
1194         b7['bg'] = '#15591f'
1195     def b_8hover(event):
1196         b8['bg'] = '#15591f'
1197     def b_9hover(event):
1198         b9['bg'] = '#15591f'
1199     def b_10hover(event):
1200         b10['bg'] = '#773585'
1201     def b_11hover(event):
1202         b11['bg'] = '#e0dd04'
1203     def b_12hover(event):
1204         b12['bg'] = '#a3295d'
1205     def b_13hover(event):
1206         b13['bg'] = '#2c2680'
1207     def b_14hover(event):
1208         b14['bg'] = '#f7a1e3'
1209     def b_15hover(event):
1210         b15['bg'] = '#15591f'
1211     def b_16hover(event):
1212         b16['bg'] = '#15591f'
1213     def b_17hover(event):
1214         b17['bg'] = '#15591f'
1215     def b_18hover(event):
1216         b18['bg'] = '#773585'
1217     def b_19hover(event):
1218         b19['bg'] = '#e0dd04'
1219     def b_20hover(event):
1220         b20['bg'] = '#a3295d'
1221     def b_21hover(event):
1222         b21['bg'] = '#d1b304'
1223     def b_22hover(event):
1224         b22['bg'] = '#d1b304'
1225     def b_23hover(event):
1226         b23['bg'] = '#d1b304'
1227     def b_24hover(event):
1228         b24['bg'] = '#d1b304'
1229     def b_25hover(event):
1230         b25['bg'] = '#d1b304'
1231     def b_26hover(event):
1232         b26['bg'] = '#d1b304'
1233     def b_27hover(event):
1234         b27['bg'] = '#d1b304'
1235     def b_28hover(event):
1236         b28['bg'] = '#d1b304'
1237     def b_29hover(event):
1238         b29['bg'] = '#d1b304'
1239     def b_30hover(event):
1240         b30['bg'] = '#d1b304'
1241     def b_31hover(event):
1242         b31['bg'] = '#2c2680'
1243     def b_32hover(event):
1244         b32['bg'] = '#f7a1e3'
1245     def b_33hover(event):
1246         b33['bg'] = '#f7a1e3'
1247     def b_34hover(event):
```

```
1248         b34['bg'] = '#15591f'
1249     def b_35hover(event):
1250         b35['bg'] = '#15591f'
1251     def b_36hover(event):
1252         b36['bg'] = '#773585'
1253     def b_37hover(event):
1254         b37['bg'] = '#e0dd04'
1255     def b_38hover(event):
1256         b38['bg'] = '#a3295d'
1257     def b_39hover(event):
1258         b39['bg'] = '#d1b304'
1259     def b_40hover(event):
1260         b40['bg'] = '#d1b304'
1261     def b_41hover(event):
1262         b41['bg'] = '#d1b304'
1263     def b_42hover(event):
1264         b42['bg'] = '#d1b304'
1265     def b_43hover(event):
1266         b43['bg'] = '#d1b304'
1267     def b_44hover(event):
1268         b44['bg'] = '#d1b304'
1269     def b_45hover(event):
1270         b45['bg'] = '#d1b304'
1271     def b_46hover(event):
1272         b46['bg'] = '#d1b304'
1273     def b_47hover(event):
1274         b47['bg'] = '#d1b304'
1275     def b_48hover(event):
1276         b48['bg'] = '#d1b304'
1277     def b_49hover(event):
1278         b49['bg'] = '#2c2680'
1279     def b_50hover(event):
1280         b50['bg'] = '#2c2680'
1281     def b_51hover(event):
1282         b51['bg'] = '#f7a1e3'
1283     def b_52hover(event):
1284         b52['bg'] = '#f7a1e3'
1285     def b_53hover(event):
1286         b53['bg'] = '#15591f'
1287     def b_54hover(event):
1288         b54['bg'] = '#773585'
1289     def b_55hover(event):
1290         b55['bg'] = '#e0dd04'
1291     def b_56hover(event):
1292         b56['bg'] = '#a3295d'
1293     def b_57hover(event):
1294         b57['bg'] = '#db4f09'
1295     def b_58hover(event):
1296         b58['bg'] = '#db4f09'
1297     def b_59hover(event):
1298         b59['bg'] = '#db4f09'
1299     def b_60hover(event):
1300         b60['bg'] = '#db4f09'
1301     def b_61hover(event):
1302         b61['bg'] = '#db4f09'
1303     def b_62hover(event):
1304         b62['bg'] = '#db4f09'
1305     def b_63hover(event):
1306         b63['bg'] = '#db4f09'
1307     def b_64hover(event):
1308         b64['bg'] = '#db4f09'
1309     def b_65hover(event):
1310         b65['bg'] = '#db4f09'
1311     def b_66hover(event):
1312         b66['bg'] = '#db4f09'
1313     def b_67hover(event):
1314         b67['bg'] = '#db4f09'
1315     def b_68hover(event):
1316         b68['bg'] = '#db4f09'
```

```
1317 def b_69hover(event):
1318     b69['bg'] = '#db4f09'
1319 def b_70hover(event):
1320     b70['bg'] = '#db4f09'
1321 def b_71hover(event):
1322     b71['bg'] = '#db4f09'
1323 def b_72hover(event):
1324     b72['bg'] = '#d1b304'
1325 def b_73hover(event):
1326     b73['bg'] = '#d1b304'
1327 def b_74hover(event):
1328     b74['bg'] = '#d1b304'
1329 def b_75hover(event):
1330     b75['bg'] = '#d1b304'
1331 def b_76hover(event):
1332     b76['bg'] = '#d1b304'
1333 def b_77hover(event):
1334     b77['bg'] = '#d1b304'
1335 def b_78hover(event):
1336     b78['bg'] = '#d1b304'
1337 def b_79hover(event):
1338     b79['bg'] = '#d1b304'
1339 def b_80hover(event):
1340     b80['bg'] = '#d1b304'
1341 def b_81hover(event):
1342     b81['bg'] = '#2c2680'
1343 def b_82hover(event):
1344     b82['bg'] = '#2c2680'
1345 def b_83hover(event):
1346     b83['bg'] = '#2c2680'
1347 def b_84hover(event):
1348     b84['bg'] = '#f7a1e3'
1349 def b_85hover(event):
1350     b85['bg'] = '#15591f'
1351 def b_86hover(event):
1352     b86['bg'] = '#773585'
1353 def b_87hover(event):
1354     b87['bg'] = '#e0dd04'
1355 def b_88hover(event):
1356     b88['bg'] = '#a3295d'
1357 def b_89hover(event):
1358     b89['bg'] = '#c0c74e'
1359 def b_90hover(event):
1360     b90['bg'] = '#c0c74e'
1361 def b_91hover(event):
1362     b91['bg'] = '#c0c74e'
1363 def b_92hover(event):
1364     b92['bg'] = '#c0c74e'
1365 def b_93hover(event):
1366     b93['bg'] = '#c0c74e'
1367 def b_94hover(event):
1368     b94['bg'] = '#c0c74e'
1369 def b_95hover(event):
1370     b95['bg'] = '#c0c74e'
1371 def b_96hover(event):
1372     b96['bg'] = '#c0c74e'
1373 def b_97hover(event):
1374     b97['bg'] = '#c0c74e'
1375 def b_98hover(event):
1376     b98['bg'] = '#c0c74e'
1377 def b_99hover(event):
1378     b99['bg'] = '#c0c74e'
1379 def b_100hover(event):
1380     b100['bg'] = '#c0c74e'
1381 def b_101hover(event):
1382     b101['bg'] = '#c0c74e'
1383 def b_102hover(event):
1384     b102['bg'] = '#c0c74e'
1385 def b_103hover(event):
```



```

1386         b103['bg'] = '#c0c74e'
1387     def b_104hover(event):
1388         b104['bg'] = '#d1b304'
1389     def b_105hover(event):
1390         b105['bg'] = '#d1b304'
1391     def b_106hover(event):
1392         b106['bg'] = '#d1b304'
1393     def b_107hover(event):
1394         b107['bg'] = '#d1b304'
1395     def b_108hover(event):
1396         b108['bg'] = '#d1b304'
1397     def b_109hover(event):
1398         b109['bg'] = '#d1b304'
1399     def b_110hover(event):
1400         b110['bg'] = '#d1b304'
1401     def b_111hover(event):
1402         b111['bg'] = '#d1b304'
1403     def b_112hover(event):
1404         b112['bg'] = '#d1b304'
1405     def b_113hover(event):
1406         b113['bg'] = '#d1b304'
1407     def b_114hover(event):
1408         b114['bg'] = '#d1b304'
1409     def b_115hover(event):
1410         b115['bg'] = '#d1b304'
1411     def b_116hover(event):
1412         b116['bg'] = '#d1b304'
1413     def b_117hover(event):
1414         b117['bg'] = '#d1b304'
1415     def b_118hover(event):
1416         b118['bg'] = '#d1b304'
1417
1418     # Functions to restore previous color of the buttons when mouse tip leave the
1419     # element button
1420
1421     def b_1leave(event):
1422         b1['bg'] = 'white'
1423     def b_2leave(event):
1424         b2['bg'] = '#9943ab'
1425     def b_3leave(event):
1426         b3['bg'] = 'yellow'
1427     def b_4leave(event):
1428         b4['bg'] = '#c92a6e'
1429     def b_5leave(event):
1430         b5['bg'] = 'pink'
1431     def b_6leave(event):
1432         b6['bg'] = 'green'
1433     def b_7leave(event):
1434         b7['bg'] = 'green'
1435     def b_8leave(event):
1436         b8['bg'] = 'green'
1437     def b_9leave(event):
1438         b9['bg'] = 'green'
1439     def b_10leave(event):
1440         b10['bg'] = '#9943ab'
1441     def b_11leave(event):
1442         b11['bg'] = 'yellow'
1443     def b_12leave(event):
1444         b12['bg'] = '#c92a6e'
1445     def b_13leave(event):
1446         b13['bg'] = '#352f9e'
1447     def b_14leave(event):
1448         b14['bg'] = 'pink'
1449     def b_15leave(event):
1450         b15['bg'] = 'green'
1451     def b_16leave(event):
1452         b16['bg'] = 'green'
1453     def b_17leave(event):
1454         b17['bg'] = 'green'

```

```
1454     def b_18leave(event):
1455         b18['bg'] = '#9943ab'
1456     def b_19leave(event):
1457         b19['bg'] = 'yellow'
1458     def b_20leave(event):
1459         b20['bg'] = '#c92a6e'
1460     def b_21leave(event):
1461         b21['bg'] = '#ffd900'
1462     def b_22leave(event):
1463         b22['bg'] = '#ffd900'
1464     def b_23leave(event):
1465         b23['bg'] = '#ffd900'
1466     def b_24leave(event):
1467         b24['bg'] = '#ffd900'
1468     def b_25leave(event):
1469         b25['bg'] = '#ffd900'
1470     def b_26leave(event):
1471         b26['bg'] = '#ffd900'
1472     def b_27leave(event):
1473         b27['bg'] = '#ffd900'
1474     def b_28leave(event):
1475         b28['bg'] = '#ffd900'
1476     def b_29leave(event):
1477         b29['bg'] = '#ffd900'
1478     def b_30leave(event):
1479         b30['bg'] = '#ffd900'
1480     def b_31leave(event):
1481         b31['bg'] = '#352f9e'
1482     def b_32leave(event):
1483         b32['bg'] = 'pink'
1484     def b_33leave(event):
1485         b33['bg'] = 'pink'
1486     def b_34leave(event):
1487         b34['bg'] = 'green'
1488     def b_35leave(event):
1489         b35['bg'] = 'green'
1490     def b_36leave(event):
1491         b36['bg'] = '#9943ab'
1492     def b_37leave(event):
1493         b37['bg'] = 'yellow'
1494     def b_38leave(event):
1495         b38['bg'] = '#c92a6e'
1496     def b_39leave(event):
1497         b39['bg'] = '#ffd900'
1498     def b_40leave(event):
1499         b40['bg'] = '#ffd900'
1500     def b_41leave(event):
1501         b41['bg'] = '#ffd900'
1502     def b_42leave(event):
1503         b42['bg'] = '#ffd900'
1504     def b_43leave(event):
1505         b43['bg'] = '#ffd900'
1506     def b_44leave(event):
1507         b44['bg'] = '#ffd900'
1508     def b_45leave(event):
1509         b45['bg'] = '#ffd900'
1510     def b_46leave(event):
1511         b46['bg'] = '#ffd900'
1512     def b_47leave(event):
1513         b47['bg'] = '#ffd900'
1514     def b_48leave(event):
1515         b48['bg'] = '#ffd900'
1516     def b_49leave(event):
1517         b49['bg'] = '#352f9e'
1518     def b_50leave(event):
1519         b50['bg'] = '#352f9e'
1520     def b_51leave(event):
1521         b51['bg'] = 'pink'
1522     def b_52leave(event):
```

```
1523         b52['bg'] = 'pink'
1524     def b_53leave(event):
1525         b53['bg'] = 'green'
1526     def b_54leave(event):
1527         b54['bg'] = '#9943ab'
1528     def b_55leave(event):
1529         b55['bg'] = 'yellow'
1530     def b_56leave(event):
1531         b56['bg'] = '#c92a6e'
1532     def b_57leave(event):
1533         b57['bg'] = '#ff5500'
1534     def b_58leave(event):
1535         b58['bg'] = '#ff5500'
1536     def b_59leave(event):
1537         b59['bg'] = '#ff5500'
1538     def b_60leave(event):
1539         b60['bg'] = '#ff5500'
1540     def b_61leave(event):
1541         b61['bg'] = '#ff5500'
1542     def b_62leave(event):
1543         b62['bg'] = '#ff5500'
1544     def b_63leave(event):
1545         b63['bg'] = '#ff5500'
1546     def b_64leave(event):
1547         b64['bg'] = '#ff5500'
1548     def b_65leave(event):
1549         b65['bg'] = '#ff5500'
1550     def b_66leave(event):
1551         b66['bg'] = '#ff5500'
1552     def b_67leave(event):
1553         b67['bg'] = '#ff5500'
1554     def b_68leave(event):
1555         b68['bg'] = '#ff5500'
1556     def b_69leave(event):
1557         b69['bg'] = '#ff5500'
1558     def b_70leave(event):
1559         b70['bg'] = '#ff5500'
1560     def b_71leave(event):
1561         b71['bg'] = '#ff5500'
1562     def b_72leave(event):
1563         b72['bg'] = '#ffd900'
1564     def b_73leave(event):
1565         b73['bg'] = '#ffd900'
1566     def b_74leave(event):
1567         b74['bg'] = '#ffd900'
1568     def b_75leave(event):
1569         b75['bg'] = '#ffd900'
1570     def b_76leave(event):
1571         b76['bg'] = '#ffd900'
1572     def b_77leave(event):
1573         b77['bg'] = '#ffd900'
1574     def b_78leave(event):
1575         b78['bg'] = '#ffd900'
1576     def b_79leave(event):
1577         b79['bg'] = '#ffd900'
1578     def b_80leave(event):
1579         b80['bg'] = '#ffd900'
1580     def b_81leave(event):
1581         b81['bg'] = '#352f9e'
1582     def b_82leave(event):
1583         b82['bg'] = '#352f9e'
1584     def b_83leave(event):
1585         b83['bg'] = '#352f9e'
1586     def b_84leave(event):
1587         b84['bg'] = 'pink'
1588     def b_85leave(event):
1589         b85['bg'] = 'green'
1590     def b_86leave(event):
1591         b86['bg'] = '#9943ab'
```

```
1592     def b_87leave(event):
1593         b87['bg'] = 'yellow'
1594     def b_88leave(event):
1595         b88['bg'] = '#c92a6e'
1596     def b_89leave(event):
1597         b89['bg'] = '#f7ff5e'
1598     def b_90leave(event):
1599         b90['bg'] = '#f7ff5e'
1600     def b_91leave(event):
1601         b91['bg'] = '#f7ff5e'
1602     def b_92leave(event):
1603         b92['bg'] = '#f7ff5e'
1604     def b_93leave(event):
1605         b93['bg'] = '#f7ff5e'
1606     def b_94leave(event):
1607         b94['bg'] = '#f7ff5e'
1608     def b_95leave(event):
1609         b95['bg'] = '#f7ff5e'
1610     def b_96leave(event):
1611         b96['bg'] = '#f7ff5e'
1612     def b_97leave(event):
1613         b97['bg'] = '#f7ff5e'
1614     def b_98leave(event):
1615         b98['bg'] = '#f7ff5e'
1616     def b_99leave(event):
1617         b99['bg'] = '#f7ff5e'
1618     def b_100leave(event):
1619         b100['bg'] = '#f7ff5e'
1620     def b_101leave(event):
1621         b101['bg'] = '#f7ff5e'
1622     def b_102leave(event):
1623         b102['bg'] = '#f7ff5e'
1624     def b_103leave(event):
1625         b103['bg'] = '#f7ff5e'
1626     def b_104leave(event):
1627         b104['bg'] = '#ffd900'
1628     def b_105leave(event):
1629         b105['bg'] = '#ffd900'
1630     def b_106leave(event):
1631         b106['bg'] = '#ffd900'
1632     def b_107leave(event):
1633         b107['bg'] = '#ffd900'
1634     def b_108leave(event):
1635         b108['bg'] = '#ffd900'
1636     def b_109leave(event):
1637         b109['bg'] = '#ffd900'
1638     def b_110leave(event):
1639         b110['bg'] = '#ffd900'
1640     def b_111leave(event):
1641         b111['bg'] = '#ffd900'
1642     def b_112leave(event):
1643         b112['bg'] = '#ffd900'
1644     def b_113leave(event):
1645         b113['bg'] = '#ffd900'
1646     def b_114leave(event):
1647         b114['bg'] = '#ffd900'
1648     def b_115leave(event):
1649         b115['bg'] = '#ffd900'
1650     def b_116leave(event):
1651         b116['bg'] = '#ffd900'
1652     def b_117leave(event):
1653         b117['bg'] = '#ffd900'
1654     def b_118leave(event):
1655         b118['bg'] = '#ffd900'
1656
1657     # Binding hover functions to respective buttons
1658
1659     b1.bind('<Enter>', b_1hover)
1660     b2.bind('<Enter>', b_2hover)
```

```
1661      b3.bind('<Enter>', b_3hover)
1662      b4.bind('<Enter>', b_4hover)
1663      b5.bind('<Enter>', b_5hover)
1664      b6.bind('<Enter>', b_6hover)
1665      b7.bind('<Enter>', b_7hover)
1666      b8.bind('<Enter>', b_8hover)
1667      b9.bind('<Enter>', b_9hover)
1668      b10.bind('<Enter>', b_10hover)
1669      b11.bind('<Enter>', b_11hover)
1670      b12.bind('<Enter>', b_12hover)
1671      b13.bind('<Enter>', b_13hover)
1672      b14.bind('<Enter>', b_14hover)
1673      b15.bind('<Enter>', b_15hover)
1674      b16.bind('<Enter>', b_16hover)
1675      b17.bind('<Enter>', b_17hover)
1676      b18.bind('<Enter>', b_18hover)
1677      b19.bind('<Enter>', b_19hover)
1678      b20.bind('<Enter>', b_20hover)
1679      b21.bind('<Enter>', b_21hover)
1680      b22.bind('<Enter>', b_22hover)
1681      b23.bind('<Enter>', b_23hover)
1682      b24.bind('<Enter>', b_24hover)
1683      b25.bind('<Enter>', b_25hover)
1684      b26.bind('<Enter>', b_26hover)
1685      b27.bind('<Enter>', b_27hover)
1686      b28.bind('<Enter>', b_28hover)
1687      b29.bind('<Enter>', b_29hover)
1688      b30.bind('<Enter>', b_30hover)
1689      b31.bind('<Enter>', b_31hover)
1690      b32.bind('<Enter>', b_32hover)
1691      b33.bind('<Enter>', b_33hover)
1692      b34.bind('<Enter>', b_34hover)
1693      b35.bind('<Enter>', b_35hover)
1694      b36.bind('<Enter>', b_36hover)
1695      b37.bind('<Enter>', b_37hover)
1696      b38.bind('<Enter>', b_38hover)
1697      b39.bind('<Enter>', b_39hover)
1698      b40.bind('<Enter>', b_40hover)
1699      b41.bind('<Enter>', b_41hover)
1700      b42.bind('<Enter>', b_42hover)
1701      b43.bind('<Enter>', b_43hover)
1702      b44.bind('<Enter>', b_44hover)
1703      b45.bind('<Enter>', b_45hover)
1704      b46.bind('<Enter>', b_46hover)
1705      b47.bind('<Enter>', b_47hover)
1706      b48.bind('<Enter>', b_48hover)
1707      b49.bind('<Enter>', b_49hover)
1708      b50.bind('<Enter>', b_50hover)
1709      b51.bind('<Enter>', b_51hover)
1710      b52.bind('<Enter>', b_52hover)
1711      b53.bind('<Enter>', b_53hover)
1712      b54.bind('<Enter>', b_54hover)
1713      b55.bind('<Enter>', b_55hover)
1714      b56.bind('<Enter>', b_56hover)
1715      b57.bind('<Enter>', b_57hover)
1716      b58.bind('<Enter>', b_58hover)
1717      b59.bind('<Enter>', b_59hover)
1718      b60.bind('<Enter>', b_60hover)
1719      b61.bind('<Enter>', b_61hover)
1720      b62.bind('<Enter>', b_62hover)
1721      b63.bind('<Enter>', b_63hover)
1722      b64.bind('<Enter>', b_64hover)
1723      b65.bind('<Enter>', b_65hover)
1724      b66.bind('<Enter>', b_66hover)
1725      b67.bind('<Enter>', b_67hover)
1726      b68.bind('<Enter>', b_68hover)
1727      b69.bind('<Enter>', b_69hover)
1728      b70.bind('<Enter>', b_70hover)
1729      b71.bind('<Enter>', b_71hover)
```

```
1730      b72.bind('<Enter>', b_72hover)
1731      b73.bind('<Enter>', b_73hover)
1732      b74.bind('<Enter>', b_74hover)
1733      b75.bind('<Enter>', b_75hover)
1734      b76.bind('<Enter>', b_76hover)
1735      b77.bind('<Enter>', b_77hover)
1736      b78.bind('<Enter>', b_78hover)
1737      b79.bind('<Enter>', b_79hover)
1738      b80.bind('<Enter>', b_80hover)
1739      b81.bind('<Enter>', b_81hover)
1740      b82.bind('<Enter>', b_82hover)
1741      b83.bind('<Enter>', b_83hover)
1742      b84.bind('<Enter>', b_84hover)
1743      b85.bind('<Enter>', b_85hover)
1744      b86.bind('<Enter>', b_86hover)
1745      b87.bind('<Enter>', b_87hover)
1746      b88.bind('<Enter>', b_88hover)
1747      b89.bind('<Enter>', b_89hover)
1748      b90.bind('<Enter>', b_90hover)
1749      b91.bind('<Enter>', b_91hover)
1750      b92.bind('<Enter>', b_92hover)
1751      b93.bind('<Enter>', b_93hover)
1752      b94.bind('<Enter>', b_94hover)
1753      b95.bind('<Enter>', b_95hover)
1754      b96.bind('<Enter>', b_96hover)
1755      b97.bind('<Enter>', b_97hover)
1756      b98.bind('<Enter>', b_98hover)
1757      b99.bind('<Enter>', b_99hover)
1758      b100.bind('<Enter>', b_100hover)
1759      b101.bind('<Enter>', b_101hover)
1760      b102.bind('<Enter>', b_102hover)
1761      b103.bind('<Enter>', b_103hover)
1762      b104.bind('<Enter>', b_104hover)
1763      b105.bind('<Enter>', b_105hover)
1764      b106.bind('<Enter>', b_106hover)
1765      b107.bind('<Enter>', b_107hover)
1766      b108.bind('<Enter>', b_108hover)
1767      b109.bind('<Enter>', b_109hover)
1768      b110.bind('<Enter>', b_110hover)
1769      b111.bind('<Enter>', b_111hover)
1770      b112.bind('<Enter>', b_112hover)
1771      b113.bind('<Enter>', b_113hover)
1772      b114.bind('<Enter>', b_114hover)
1773      b115.bind('<Enter>', b_115hover)
1774      b116.bind('<Enter>', b_116hover)
1775      b117.bind('<Enter>', b_117hover)
1776      b118.bind('<Enter>', b_118hover)
1777
1778      # Binding leave function to respective buttons
1779
1780      b1.bind('<Leave>', b_1leave)
1781      b2.bind('<Leave>', b_2leave)
1782      b3.bind('<Leave>', b_3leave)
1783      b4.bind('<Leave>', b_4leave)
1784      b5.bind('<Leave>', b_5leave)
1785      b6.bind('<Leave>', b_6leave)
1786      b7.bind('<Leave>', b_7leave)
1787      b8.bind('<Leave>', b_8leave)
1788      b9.bind('<Leave>', b_9leave)
1789      b10.bind('<Leave>', b_10leave)
1790      b11.bind('<Leave>', b_11leave)
1791      b12.bind('<Leave>', b_12leave)
1792      b13.bind('<Leave>', b_13leave)
1793      b14.bind('<Leave>', b_14leave)
1794      b15.bind('<Leave>', b_15leave)
1795      b16.bind('<Leave>', b_16leave)
1796      b17.bind('<Leave>', b_17leave)
1797      b18.bind('<Leave>', b_18leave)
1798      b19.bind('<Leave>', b_19leave)
```

```
1799      b20.bind('<Leave>', b_20leave)
1800      b21.bind('<Leave>', b_21leave)
1801      b22.bind('<Leave>', b_22leave)
1802      b23.bind('<Leave>', b_23leave)
1803      b24.bind('<Leave>', b_24leave)
1804      b25.bind('<Leave>', b_25leave)
1805      b26.bind('<Leave>', b_26leave)
1806      b27.bind('<Leave>', b_27leave)
1807      b28.bind('<Leave>', b_28leave)
1808      b29.bind('<Leave>', b_29leave)
1809      b30.bind('<Leave>', b_30leave)
1810      b31.bind('<Leave>', b_31leave)
1811      b32.bind('<Leave>', b_32leave)
1812      b33.bind('<Leave>', b_33leave)
1813      b34.bind('<Leave>', b_34leave)
1814      b35.bind('<Leave>', b_35leave)
1815      b36.bind('<Leave>', b_36leave)
1816      b37.bind('<Leave>', b_37leave)
1817      b38.bind('<Leave>', b_38leave)
1818      b39.bind('<Leave>', b_39leave)
1819      b40.bind('<Leave>', b_40leave)
1820      b41.bind('<Leave>', b_41leave)
1821      b42.bind('<Leave>', b_42leave)
1822      b43.bind('<Leave>', b_43leave)
1823      b44.bind('<Leave>', b_44leave)
1824      b45.bind('<Leave>', b_45leave)
1825      b46.bind('<Leave>', b_46leave)
1826      b47.bind('<Leave>', b_47leave)
1827      b48.bind('<Leave>', b_48leave)
1828      b49.bind('<Leave>', b_49leave)
1829      b50.bind('<Leave>', b_50leave)
1830      b51.bind('<Leave>', b_51leave)
1831      b52.bind('<Leave>', b_52leave)
1832      b53.bind('<Leave>', b_53leave)
1833      b54.bind('<Leave>', b_54leave)
1834      b55.bind('<Leave>', b_55leave)
1835      b56.bind('<Leave>', b_56leave)
1836      b57.bind('<Leave>', b_57leave)
1837      b58.bind('<Leave>', b_58leave)
1838      b59.bind('<Leave>', b_59leave)
1839      b60.bind('<Leave>', b_60leave)
1840      b61.bind('<Leave>', b_61leave)
1841      b62.bind('<Leave>', b_62leave)
1842      b63.bind('<Leave>', b_63leave)
1843      b64.bind('<Leave>', b_64leave)
1844      b65.bind('<Leave>', b_65leave)
1845      b66.bind('<Leave>', b_66leave)
1846      b67.bind('<Leave>', b_67leave)
1847      b68.bind('<Leave>', b_68leave)
1848      b69.bind('<Leave>', b_69leave)
1849      b70.bind('<Leave>', b_70leave)
1850      b71.bind('<Leave>', b_71leave)
1851      b72.bind('<Leave>', b_72leave)
1852      b73.bind('<Leave>', b_73leave)
1853      b74.bind('<Leave>', b_74leave)
1854      b75.bind('<Leave>', b_75leave)
1855      b76.bind('<Leave>', b_76leave)
1856      b77.bind('<Leave>', b_77leave)
1857      b78.bind('<Leave>', b_78leave)
1858      b79.bind('<Leave>', b_79leave)
1859      b80.bind('<Leave>', b_80leave)
1860      b81.bind('<Leave>', b_81leave)
1861      b82.bind('<Leave>', b_82leave)
1862      b83.bind('<Leave>', b_83leave)
1863      b84.bind('<Leave>', b_84leave)
1864      b85.bind('<Leave>', b_85leave)
1865      b86.bind('<Leave>', b_86leave)
1866      b87.bind('<Leave>', b_87leave)
1867      b88.bind('<Leave>', b_88leave)
```

```

1868         b89.bind('<Leave>', b_89leave)
1869         b90.bind('<Leave>', b_90leave)
1870         b91.bind('<Leave>', b_91leave)
1871         b92.bind('<Leave>', b_92leave)
1872         b93.bind('<Leave>', b_93leave)
1873         b94.bind('<Leave>', b_94leave)
1874         b95.bind('<Leave>', b_95leave)
1875         b96.bind('<Leave>', b_96leave)
1876         b97.bind('<Leave>', b_97leave)
1877         b98.bind('<Leave>', b_98leave)
1878         b99.bind('<Leave>', b_99leave)
1879         b100.bind('<Leave>', b_100leave)
1880         b101.bind('<Leave>', b_101leave)
1881         b102.bind('<Leave>', b_102leave)
1882         b103.bind('<Leave>', b_103leave)
1883         b104.bind('<Leave>', b_104leave)
1884         b105.bind('<Leave>', b_105leave)
1885         b106.bind('<Leave>', b_106leave)
1886         b107.bind('<Leave>', b_107leave)
1887         b108.bind('<Leave>', b_108leave)
1888         b109.bind('<Leave>', b_109leave)
1889         b110.bind('<Leave>', b_110leave)
1890         b111.bind('<Leave>', b_111leave)
1891         b112.bind('<Leave>', b_112leave)
1892         b113.bind('<Leave>', b_113leave)
1893         b114.bind('<Leave>', b_114leave)
1894         b115.bind('<Leave>', b_115leave)
1895         b116.bind('<Leave>', b_116leave)
1896         b117.bind('<Leave>', b_117leave)
1897         b118.bind('<Leave>', b_118leave)
1898
1899         root.mainloop()
1900
1901     # Creating labels
1902     # Signin main label
1903     signin_canvas.create_text(125, 70, text='Sign In Here',
1904                             font=('Impact',30,'bold'),fill='#285243')
1905     # Username label
1906     signin_canvas.create_text(85, 110, text='Username:',
1907                             font=('Helvetica',15,'bold'),fill='black')
1908
1909     # Username entry widget
1910     username_entry_widget = Entry(window, width = 40,border=2)
1911     username_entry_widget_window = signin_canvas.create_window(155, 135, window =
1912     username_entry_widget)
1913
1914     # Password label
1915     signin_canvas.create_text(85, 175, text='Password:',
1916                             font=('Helvetica',15,'bold'),fill='black')
1917     password_entry_widget = Entry(window, width = 40, border=2, show='*')
1918
1919     # Password entry widget
1920     password_entry_widget_window = signin_canvas.create_window(155, 200, window =
1921     password_entry_widget)
1922
1923     # Initiating tooltip
1924     tip = Balloon(window)
1925
1926     # Signin Button
1927     signin_button = Button(window, text='SIGN
1928     IN',command=signin,width=30,font=('Helvetica',10,'bold'),bg='#cccccc')
1929     signin_button_window = signin_canvas.create_window(155, 250, window = signin_button)
1930     tip.bind_widget(signin_button, balloonmsg = 'Sign In')
1931
1932     # Signup Button
1933     signup_button = Button(window, text='SIGN UP',command=signup,width =
1934     30,font=('Helvetica',10,'bold'),bg='#cccccc')
1935     signup_button_window = signin_canvas.create_window(155, 300, window = signup_button)
1936     tip.bind_widget(signup_button, balloonmsg = 'Create Account')

```



```
1930
1931     window.mainloop()
1932
1933 # Launching the program
1934 main()
1935
```