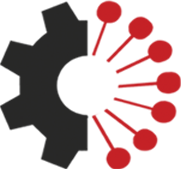
Project Report

**Computer Programming – II (MCT-243)**

**CODE ATOM**

******

Submitted By : 2018-MC-40

AYESHA ASLAM

Submitted To: Mr. M. Ahsan Naeem

DEPARTMENT OF MECHATRONICS & CONTROL ENGINEERING

University of Engineering and Technology, Lahore.

September 06, 2020

# Project Abstract:

**CONTENTS:**

* General Overview
* Introduction
* Methodology
* Code

**General Overview:**

This is a python coded version of periodic Table. Here, we can search properties of any atom or a list of atoms. We can also search for a range of properties i.e. all the metals is the periodic table. Furthermore, a single atom can be searched by using its name, symbol or any other properties. This version of periodic table is far more better than the conservative hard-form of periodic Table, where the user has to search manually for a single property. Gathering all the properties, the code-atom also provide a very good user interface which help in questing the properties easily and apace.

**Introduction:**

This program will help the user to quest for different properties of different elements. The user just have to enter a single property like atomic number or mass number or the name or symbol of element and all the properties of that element will be printed. The user can also quest for a range of properties like all the elements in a period or all the metals in the periodic table.

Some Features:

**Good User Interface:** The program provides the user a great interface for searching of the required element or a list of element. It guides the user throughout the program and aid him in his quest.

**Methodology:**

The whole periodic Table is stored in a file, from where it is induced in our code. The code than uses simple logic techniques to reach the specified element. The whole process and program is explained here:

When user run the code the logo is open which is created in main program by using image (using pillow module attributes).

Logo has two buttons start and exit. exit button quit all the root of tkinter and start button call the class periodic table and window is initialized.

Initial function created the window using Toplevel() method of tkinter and its background color geometry size is set and creating frames for rest of features and having two buttons and one label. One button is used to quer single element and other button is used to query properties of range of elements.

* If single button pressed it called single properties functions which displayed further five buttons like name, symbol, atomic number, group number, period number which are calling further five functions:

1. Name button call Name of element()
2. symbol button call Symbol()
3. Atomic number button call Atomic number()
4. Period button call Period()
5. Group button call Group()

* If range button is pressed it called range properties function which displayed further 7 buttons and all have their respective functions to call:

1. Atomic ranges button call >atm\_ranges()
2. Mass ranges button call > mass()
3. Electronegativity ranges button call> EN\_ranges()
4. Ionization energy ranges button call>IE\_ranges()
5. Electron affinity ranges button call> EA\_ranges()
6. Type button call >Type()
7. State button call > state()

and these functions called clickable function which get values from previous functions and displayed onto the window using labels with aid of display properties, remaining pop up and remaining root, content display, image displayed functions.

Single elements has it own clickable function name as pro\_clicked and own image display function.

Range elements has its own range clicked function and own image range function which is taking aid through image play and forward and previous function to display ranges of images in new window.

# Class Hierarchy:

Show the class hierarchy via UML diagram showing class name and attributes, method signatures and links between different classes.

**PERIODIC\_TABLE**

**ATTRIBUTES**

* Detail
* Image folder
* Periodic table
* Range details

**Methods()**

* \_\_init\_\_
* Single\_properties
* Range\_properties
* Name\_of\_element
* Symbol
* Atomic number
* Period
* Group
* EN-ranges
* IE\_ranges
* EA\_ranges
* Mass
* Atm\_ranges
* Type
* State
* Pro\_clicked
* Range\_clicked
* Display\_properties
* Content\_display
* Display\_prop1
* Remaining\_pop\_up
* Remaing root
* Next
* Property root
* Image display
* Img\_ranges
* Image\_play
* Forward
* Previous

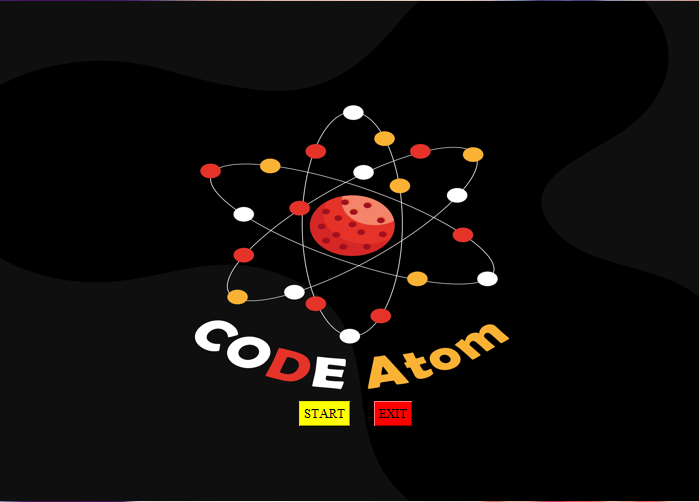
Detail of classes:

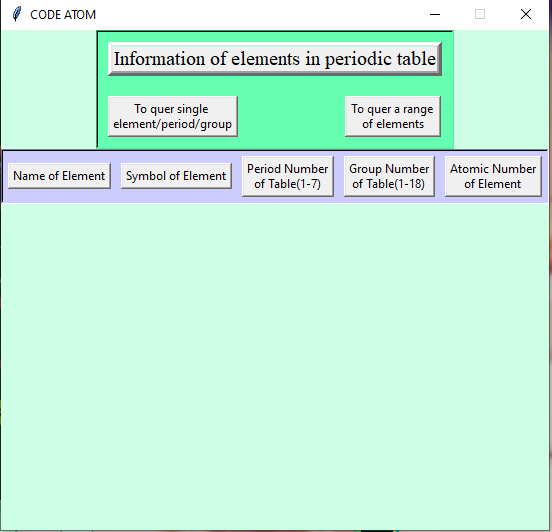
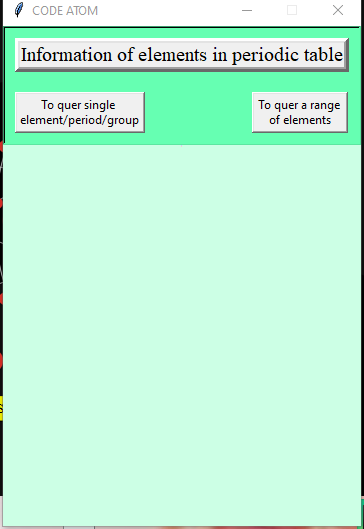
|  |  |  |  |
| --- | --- | --- | --- |
| Class Periodic\_table | Description of Attribute/Method | Purpose of Attribute/Method | Any other detail needed |
| Att1 detail | List of names of all properties | It is used in various functions for properties name as text of labels. | ['Name of Element', 'Symbol of Element', 'Period Number ', 'G....] |
| Att2 Img\_folders | List of names of all images folders | It is used in Image functions for folder name as text of labels and also for image source name. | ['presentation', 'pure elements', 'uses', 'properties'] |
| Att3 Periodic table | List having lists of all elements and its information | This attribute used to get elements and their information which is displaying in GUI. | [['Hydrogen', 'H', '1', '1', '1', '1.008', 'Non Metal......]……….] |
| Att4 Range details | Dictionary having number keys and values are indexes of element properties. | This attribute used in range functions to get indices of properties using the values of keys. | {1: 4, 2: 5, 3: 7, 4: 8, 5: 9, 6: 6, 7: 10} |
| Method1 \_\_init\_\_ | It initializes the class in which all the required widgets have been created. | This \_\_init\_\_ function created a main window and all the required widgets which has to be displayed in GUI. | e.g. the main root named as window, buttons like for query of single and ranges and main label, certain frames which is used in rest of functions. |
| Method2 Single properties | It is called by button named as to quer single properties and having buttons for connecting rest of functions. | It is displaying buttons having text as name of properties assigned and calling the functions w.r.t to pressed button. | e.g. name, symbol, atomic number, group number, period number |
| Method3 Range properties | It is called by button named as to quer range properties and having buttons for connecting rest of functions. | It is displaying buttons having text as name of properties assigned and calling the functions w.r.t to pressed button. | e.g. ranges of electronegativity, ionization energy, electron affinity, atomic number mass numbers, type, state. |
| Method4 Name of element | It called by button name created in single properties function and option menu have list of names of all elements. | In this function, list of all names of elements are collected from class attribute “periodic table” and Set as values for option menu and getting string variable when any value is selected and calling clickable function. | E.g.  self.values = [i[0] for i in Periodic\_table.periodic\_table]  self.name = OptionMenu(self.centerframe, self.variable1, \*self.values ,command=self.pro\_clicked) ## button having menu and call the function |
| Method5 Symbol of element | It called by button symbol created in single properties function and option menu have list of symbol of all elements. | In this function, list of all symbols of elements are collected from class attribute “periodic table” and Set as values for option menu and getting string variable when any value is selected and calling clickable function. | E.g.  self.values = [i[1] for i in Periodic\_table.periodic\_table]  self.name = OptionMenu(self.centerframe, self.variable1, \*self.values ,command=self.pro\_clicked) ## button having menu and call the function |
| Method6 Period | It called by button period created in single properties function and option menu have list of period numbers of elements ranges from 1-7. | In this function, list of all period number is created and Set as values for option menu and getting int variable when any value is selected and calling clickable function. | E.g.  self.values = [1,2,3,4,5,6,7]  self.name = OptionMenu(self.centerframe, self.variable1, \*self.values ,command=self.pro\_clicked) ## button having menu and call the function |
| Method7 Group | It called by button group created in single properties function and option menu have list of group numbers of elements ranges from 1-18. | In this function, list of all group number is created and Set as values for option menu and getting int variable when any value is selected and calling clickable function. | E.g.  self.values = [i for i in range(1,19)]  self.name = OptionMenu(self.centerframe, self.variable1, \*self.values ,command=self.pro\_clicked) ## button having menu and call the function |
| Method8 Atomic number | It called by button atomic number created in single properties function and option menu have list of atomic numbers of all elements. | In this function, list of all atomic numbers of elements are collected from class attribute “periodic table” and Set as values for option menu and getting string variable when any value is selected and calling clickable function. | E.g.  self.values = [i[4] for i in Periodic\_table.periodic\_table]  self.name = OptionMenu(self.centerframe, self.variable1, \*self.values ,command=self.pro\_clicked) ## button having menu and call the function |
| Method9 EN ranges | It called by button electronegativity created in range properties function and radio buttons gives 4 options of ranges of electronegativities. | In this function, list of tuples is created in 1st tuple value is treated as text of radio buttons and 2nd tuple value is treated values to be selected in radio buttons as variable and called the range clicked function. | e.g.  self.options = [("0-1", 0), ("1-2", 1), ("2-3", 2), ("N/A", 3)]  Radiobutton(self.root, text=text, variable=self.range\_EN, value=r\_value,command=lambda: self.range\_clicked(self.range\_EN.get())) |
| Method10 EA ranges | It called by button electron affinity created in range properties function and radio buttons gives 4 options of ranges of electron affinity. | In this function, list of tuples is created in 1st tuple value is treated as text of radio buttons and 2nd tuple value is treated values to be selected in radio buttons as variable and called the range clicked function. | e.g.  self.options = [("0-100", 0), ("100-250", 1), ("250-400", 2), ("N/A", 3)]  Radiobutton(self.root, text=text, variable=self.range\_EA, value=r\_value,command=lambda: self.range\_clicked(self.range\_EA.get())) |
| Method11 atm ranges | It called by button atomic number ranges created in range properties function and radio buttons gives 4 options of ranges of atomic numbers. | In this function, list of tuples is created in 1st tuple value is treated as text of radio buttons and 2nd tuple value is treated values to be selected in radio buttons as variable and called the range clicked function. | e.g.  elf.options = [("1-30", 0), ("30-60", 1), ("60-90", 2), ("90-118", 3)]  Radiobutton(self.root, text=text, variable=self.range\_Atom, value=r\_value,command=lambda: self.range\_clicked(self.range\_Atom.get())) |
| Method12 IE ranges | It called by button ionization energy created in range properties function and radio buttons gives 4 options of ranges of ionization energies. | In this function, list of tuples is created in 1st tuple value is treated as text of radio buttons and 2nd tuple value is treated values to be selected in radio buttons as variable and called the range clicked function. | e.g.  self.options = [(**"100-500"**, 0), (**"500-1000"**, 1), (**"1000-1500"**, 2), (**"N/A"**, 3)]  Radiobutton(self.root, text=text, variable=self.range\_IE, value=r\_value,command=**lambda**: self.range\_clicked(self.range\_IE.get()) |
| Method13 mass ranges | It called by button mass number range created in range properties function and radio buttons gives 4 options of ranges of mass numbers. | In this function, list of tuples is created in 1st tuple value is treated as text of radio buttons and 2nd tuple value is treated e.g.values to be selected in radio buttons as variable and called the range clicked function. | e.g.  self.options = [(**"1-75"**, 0), (**"75-150"**, 1), (**"150-225"**, 2), (**"225-300"**, 3)]  Radiobutton(self.root, text=text, variable=self.range\_mass, value=r\_value,command=**lambda**: self.range\_clicked(self.range\_mass.get())) |
| Method14 type | It called by button type created in range properties function and radio buttons gives 3 options of types. | In this function, list of tuples is created in 1st tuple value is treated as text of radio buttons and 2nd tuple value is treated values to be selected in radio buttons as variable and called the range clicked function. | e.g.  self.options = [(**"Metal"**, 0), (**"Non-Metal"**, 1), (**"Metalloid"**, 2)]  Radiobutton(self.root, text=text, variable=self.range\_Type, value=r\_value,command=**lambda**: self.range\_clicked(self.range\_Type.get())) |
| Method15 state | It called by button state created in range properties function and radio buttons gives 3 options of states. | In this function, list of tuples is created in 1st tuple value is treated as text of radio buttons and 2nd tuple value is treated values to be selected in radio buttons as variable and called the range clicked function. | e.g.  self.options = [(**"Solid"**, 0), (**"Liquid"**, 1), (**"Gas"**, 2)]  Radiobutton(self.root, text=text, variable=self.range\_State, value=r\_value,command=**lambda**: self.range\_clicked  (self.range\_State.get())) |
| Method16  Property root | Having window for clicked functions of single and range properties. | Creating window and frames for clicked functions. | *"""Creating root for property diplay"""* self.prop\_root = Toplevel() *## creating window for displaying properties* self.prop\_root.configure(bg=**'sky blue'**) self.propframe = Frame(self.prop\_root, bd=2, relief=SUNKEN, bg=**'sky blue'**) *## creating frame for property root* self.propframe.pack() |
| Method17 Pro-clicked | It is called by bottons of single properties and called diplay functions,property root function to create new window.  One input argument as value. | It checked which button is pressed and display the properties of respective clicked button using display properties and display\_prop1 functions. | e.g.given is few part of code:  self.property\_root()   **if** self.button == **"single"**:  self.click = **"No"** self.item\_index = self.values.index(value)  self.selected = [Periodic\_table.periodic\_table[self.item\_index]]  self.imageDisplay(self.item\_index)  etc. |
| Method18 range-clicked | It is called by bottons of range properties and called diplay functions,property root function to create new window.  1 input argument as value1 | It checked which button is pressed and display the properties of respective clicked button using display properties and display\_prop1 functions. | e.g. given is few part for sae of example:  **def** range\_clicked(self,value1):  *"""function if any of button of range properties click and displaying its output"""* self.property\_root()   **if** self.ranges == **"Atomic\_no"**:  **etc.** |
| Method19 display-properties | Having labels which are placed by pack layout.  Row, dis\_root (parent for labels) as input arguments. | Labels displaying text using attribute detail to display name of properties. | Eg. self.prop1 = Label(dis\_root, text=**f"{**Periodic\_table.detail[0]**}"**) |
| Method20 display-prop1 | Having labels and 2 buttons to if length of list of selected item is greater than 25.  Row\_number and select\_item as input argument. | Calling the content display functions and one image button to call image functions and one button is appear if length of selected item list is greater than 25. | item = [atm[4] **for** atm **in** select\_items] *## button display if buttons for range of elements call the display\_prop1 function* **if** self.click == **"yes"**:  self.click\_button = Button(self.propframe, text=**"Click to view images"**, command=**lambda**: self.img\_range(item)) |
| Method21 content-display | Having labels for displaying element information  Root,row\_no and select items as input aregument. | It is collecting text for labels using items of selected list of items. | e.g. self.elem\_name = Label(root, text=**f"{**selected[list\_num][0]**}"**) |
| Method22 remaining-pop-up | It is displayed when length of selected item list is greater than 25.  Limit and select item as input arguments. | Displaying property in new window for remaining elements of selected item list  And calling Next function if length is greater than 50. | self.remaining\_root() self.propframe2.pack() self.display\_properties(1, self.propframe2) row\_num = 3 **for** i **in** range(limit, len(select\_items)):  **if** i < 50:  self.content\_diplay(self.propframe2, row\_num, select\_items, i)  row\_num += 1 |
| Method23 remaining-root | Having window and frame for remaining pop-up function and next function | Creating frames in a new window. | self.new\_window = Toplevel() *##### creating window for remaing elements* self.new\_window.configure(bg=**'sky blue'**) self.propframe2 = Frame(self.new\_window, bd=2, relief=SUNKEN, bg=**'sky blue'**) *#pack in reamining pop up function.* self.propframe3 = Frame(self.new\_window, bd=2, relief=SUNKEN, bg=**'sky blue'**) *#pack in NExt function* self.propframe4 = Frame(self.new\_window, bd=2, relief=SUNKEN, bg=**'sky blue'**) |
| Method24 Next-ele | It is called if list length is grater then 50 and having limit and selected list as input argument | Forgetting the previous frames and packing new frames to display rest of elements | self.propframe2.pack\_forget() self.propframe3.pack() self.display\_properties(1, self.propframe3) row\_num = 3 **for** i **in** range(50, len(select\_items)):  **if** i < 79:  self.content\_diplay(self.propframe3, row\_num, select\_items, i) |
| Method25 image-Display | Displaying single element Images of element symbol and uses properties and pure form. | Using Pillow module attributes like Image and Photoimage and labels playing images from image folders. | e.g.  **for** i **in** range(len(Periodic\_table.img\_folders)):  **try**:  self.ele\_img = Image.open(**f'{**Periodic\_table.img\_folders[i]**}/{**img\_name + 1**}.png'**)  **except**:  self.ele\_img = Image.open(**f'{**Periodic\_table.img\_folders[i]**}/{**img\_name + 1**}.jpg'**)   self.ele\_img = self.ele\_img.resize((300, 300)) *### resizing of open images* self.ele\_img = ImageTk.PhotoImage(self.ele\_img)  self.img.append(self.ele\_img)*##showing image to python* |
| Method26 Img-range | Displaying range of Images of elements symbol and uses properties and pure form. | Using Pillow module attributes like Image and Photoimage and labels playing images from image folders. And calling forward and previous functions | self.all\_img=[ele,pure,use,phy] **for** img\_list **in** range(len(self.all\_img)):  **for** img\_name **in** item:  **try**:  self.ele\_img = Image.open(**f'{**Periodic\_table.img\_folders[img\_list]**}/{**img\_name**}.png'**)   **except**:  self.ele\_img = Image.open(**f'{**Periodic\_table.img\_folders[img\_list]**}/{**img\_name**}.jpg'**) |
| Method27 image-play | Having labels and forward and previous button to displayed in every click. | Displaying next images images on clicking forward and previos button as aid of forward and previous function | **for** j **in** range(len(self.all\_img)):  self.ele\_img\_label = Label(self.img\_frame2, image=self.all\_img[j][img\_no])  self.ele\_img\_label.grid(row=**f"{**row + 1**}"**, column=**f"{**col**}"**, columnspan=**"4"**)  col += 4 self.back\_button = Button(self.img\_frame2, text=**"Previous"**, command=**lambda**: self.Previous(img\_no - 1), bg=**"orange"**).grid(row=**"14"**, column=**"0"**) *## button to get previous images* self.frwd\_button = Button(self.img\_frame2, text=**"Next"**, command=**lambda**: self.forward(img\_no + 1), bg=**"orange"**) |
| Method28 forward | Having forward and reverse button’s text over write. | Calling image play function and disable the forward button if last images is displayed | self.Image\_play(img\_no) **if** img\_no == (len(self.all\_img[0]) - 1):  self.frwd\_button = Button(self.img\_frame2, text=**"Next"**, state=DISABLED) |
| Method29 Previous | Having forward and reverse button’s text over write. | Calling image play function and disable the previous button if last images is displayed | self.Image\_play(img\_no) **if** img\_no == 0:  self.back\_button = Button(self.img\_frame2, text=**"Previous"**, state=DISABLED) |

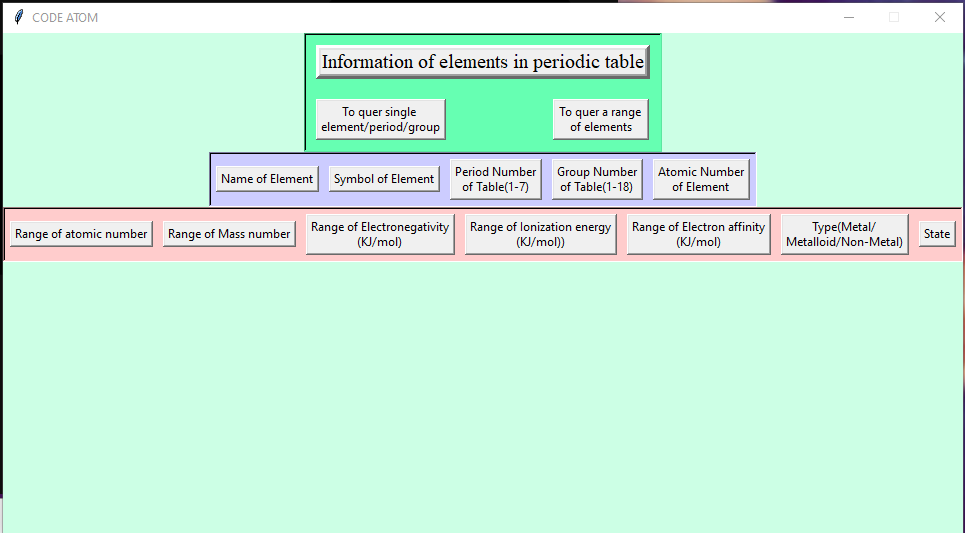
GUI:

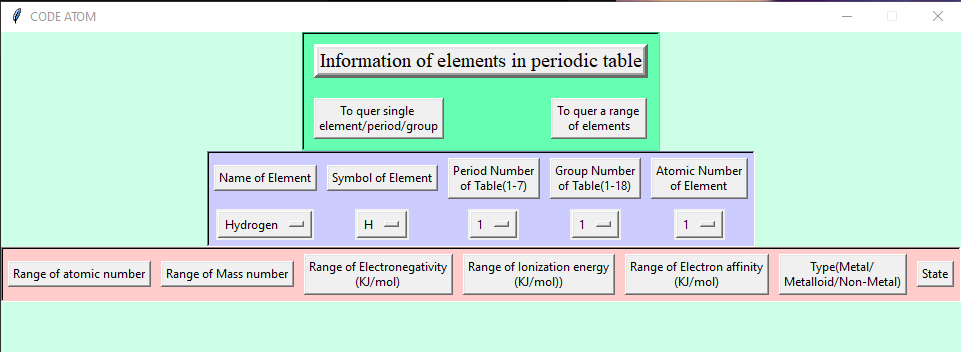
|  |  |  |  |
| --- | --- | --- | --- |
| GUI Widget | Purpose | Connection with Program | Any other detail needed |
| Widget1 Type Label | Labels are used to display text and images tkinter window. | Labels are displaying information of elements when any button is pressed and it is creating in various functions e.g. displaying properties, content display, display\_prop1 etc. | Labels having text e.g.  'presentation', 'pure elements', 'uses', 'properties', 'Name of Element', 'Symbol of Element', 'Period Number’, etc. and also images. |
| Widget2 Type Frame | Frames are used in window to organize different widgets and displaying in presentable way. | In tkinter windows, frames have been used with pack layout, in which different widgets are displaying with grid layout. | e.g. prop frame, center frame, top frame, root frame etc. |
| Widget3 Type Button | Buttons having text and act as connection between two functions.  On pressing button, it calls the associated function. | Different buttons like to quer single properties, range properties are made through which different functions are calling associated using command attribute of button widget and displaying name of button using text attribute of button widget. | e.g. buttons like name, symbol, atomic number group number, period number and for range properties like ranges of electronegativity, ionization energy, electron affinity, etc. also have button to show up the image roots and remaining element roots. |
| Widget4 Type Radio Button | Creating sub-buttons for the main buttons of range properties which is pressed. | Radio buttons are used in range functions where it displaying the options for the user to choose and it display the information of selected option. | e.g. on pressing state:  solid, liquid, gas options are created and if any of this selected it will display information all solid elements of periodic table and first option is set as default option. |
| Widget5 Type option menu | It is creating menu of all single properties’ information. | Option menu opens a menu of all options which are existing for pressed button. | e.g. on pressing name menu displayed which as 1st value as default value on clicking all the options of that button displayed which gives all information of selected name in anew window |

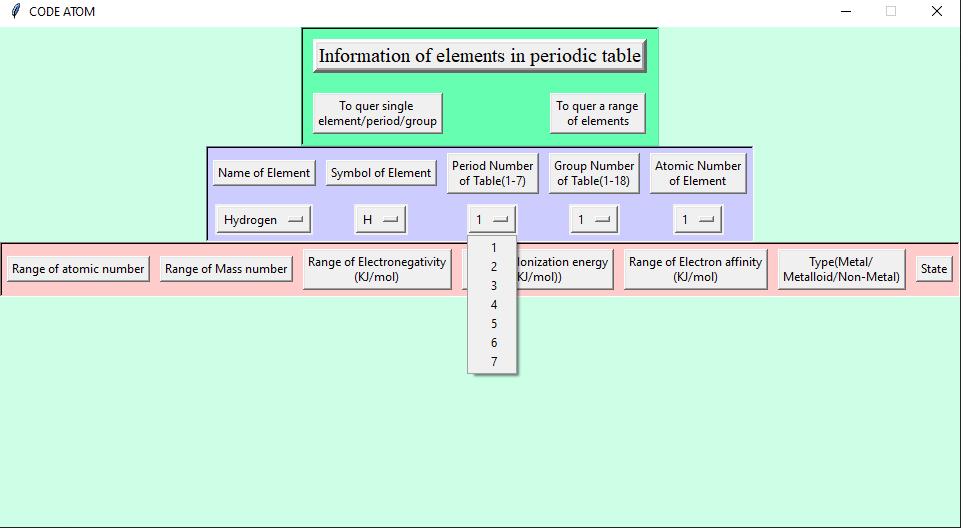
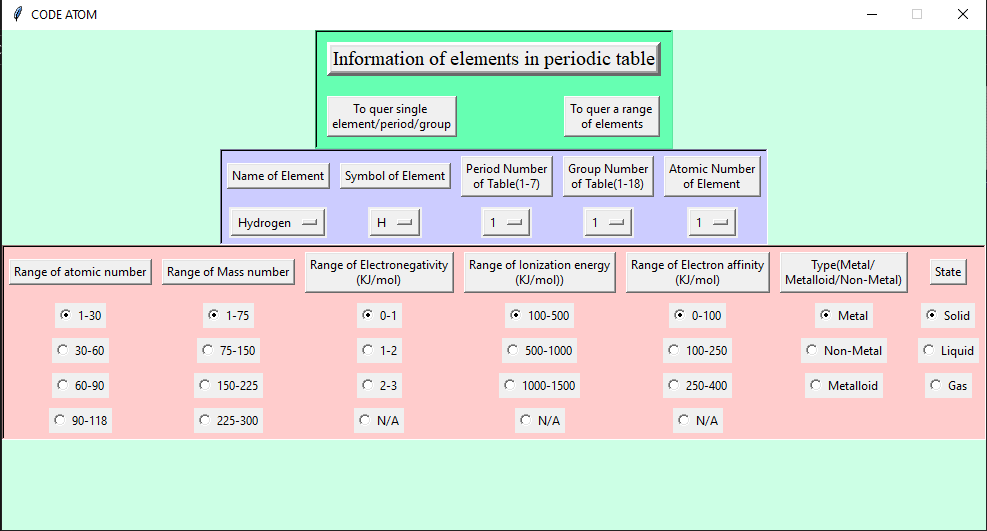
Screen Shot of GUI

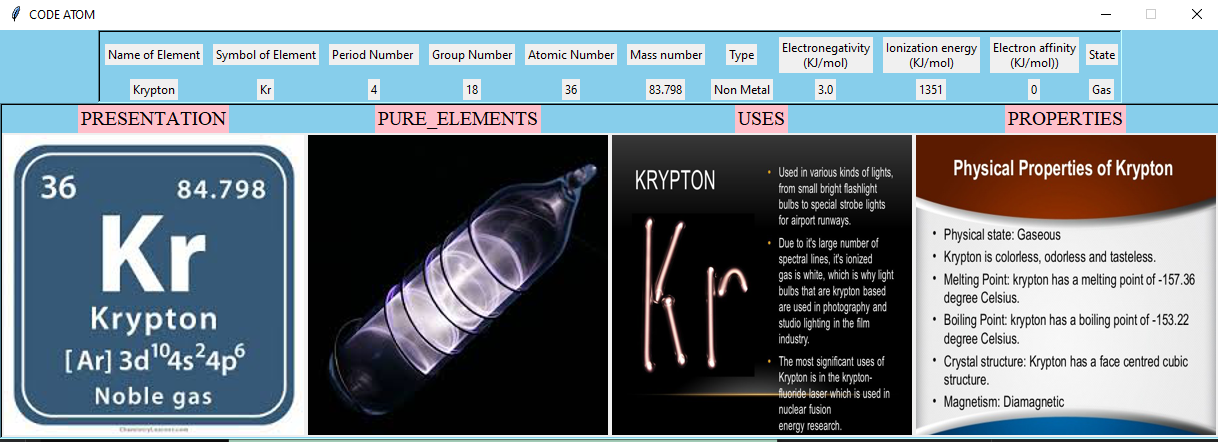


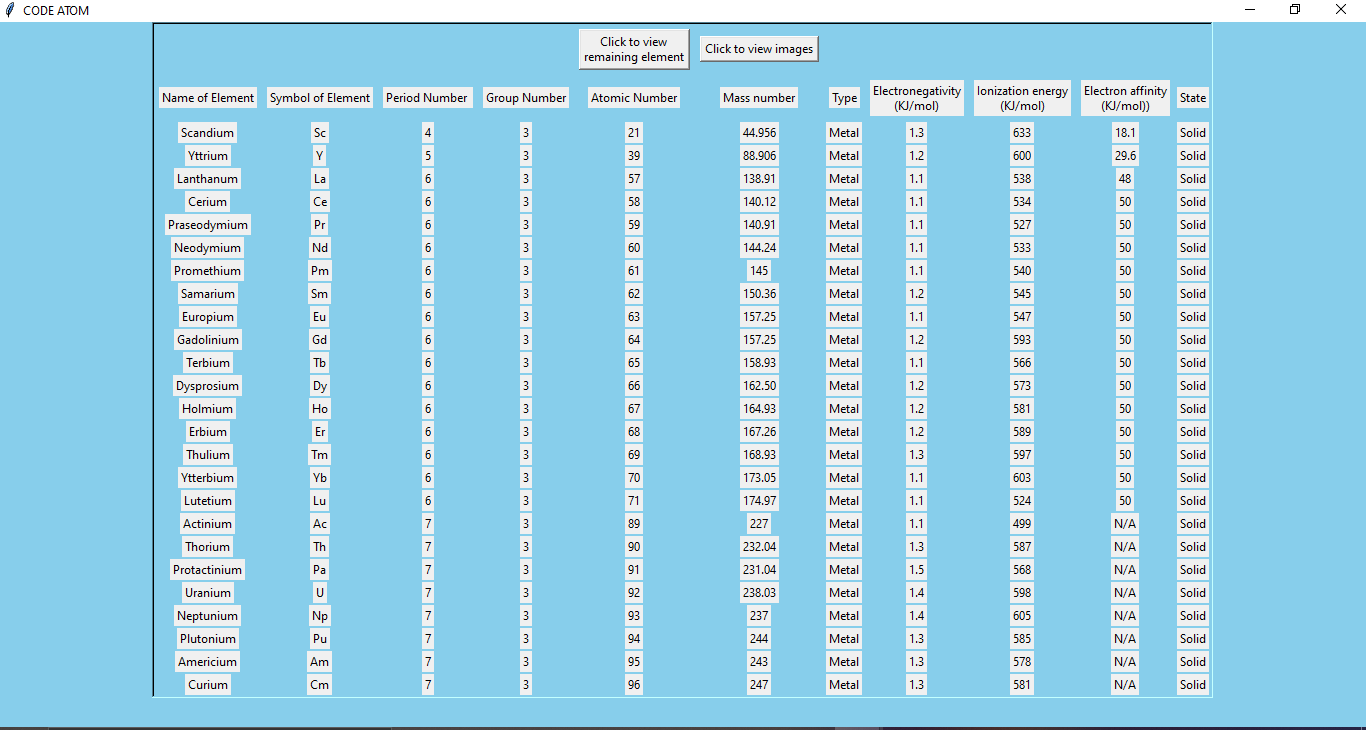


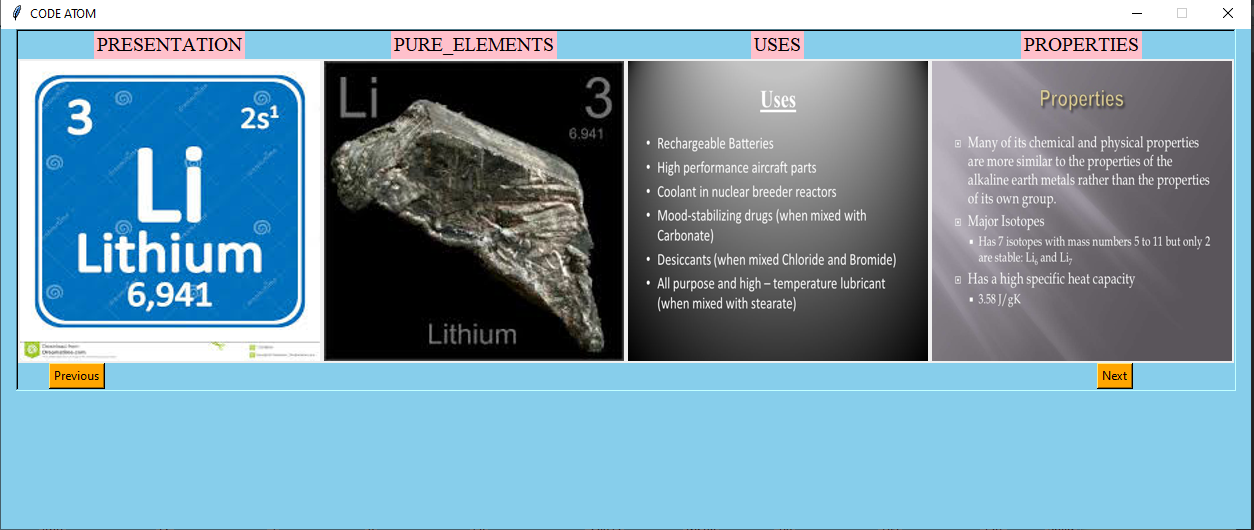
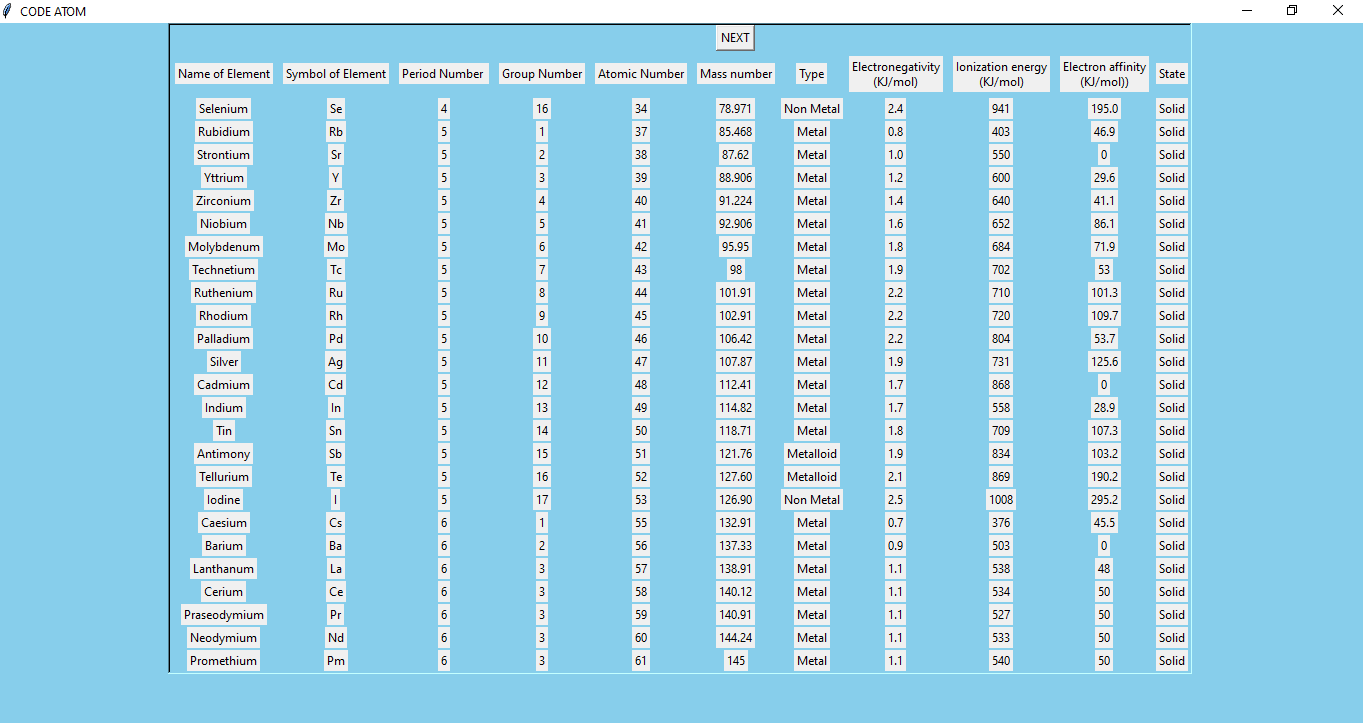


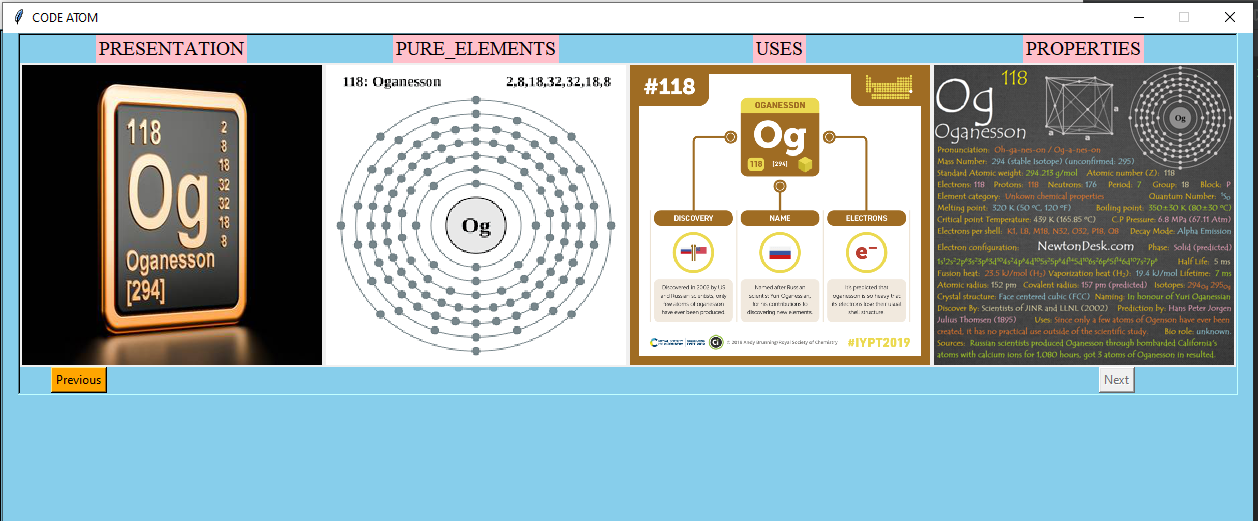








****

****