Indian Institute of Technology Madras Zanzibar Campus

Chemistry Assignment I

# Manas Pandya - ZDA23B019

May 2024

## 1 Assignment Project Statement:

In this assignment we have been asked to write a Python program to find the electronic configuration of given chemical elements from the periodic table. This involves:

- Getting a list of chemical elements from the user as input. Eg: "O","H","He"
- Generate a table-like output containing subshell names (1s,2s,2p,etc.) on 'columns' and element symbol (eg: H) on 'rows'

## 2 The code

The following is the code used for the assignment:

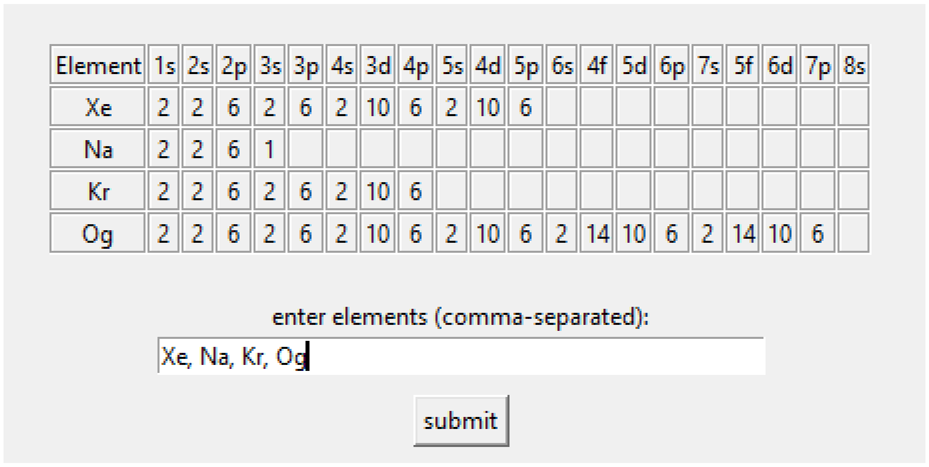
```
""" CHEM HW 1 of Manas Pandya - to display electronic configuration of user-put list of elements"""
import tkinter as tk
from tkinter import ttk

orbital_sequence = [
    "1s", "2s", "2p", "3s", "3p", "4s", "3d", "4p", "5s", "4d", "5p", "6s", "4f", "5d", "6p",
    "7s", "5f", "6d", "7p", "8s"
    #i have changed the code's list of orbitals from the question according to the aufbau principle.
    ]

max_electrons = {
    "s": 2, "p": 6, "d": 10, "f": 14, "g": 18
}

element_electrons = {
    "H": 1, "He": 2, "Li": 3, "Be": 4, "B": 5, "C": 6, "N": 7, "O": 8, "F": 9, "Ne": 10,
    "Na": 11, "Mg": 12, "Al": 13, "Si": 14, "P": 15, "S": 16, "Cl": 17, "Ar": 18,
    "K": 19, "Ca": 20, "Sc": 21, "Ti": 22, "V": 23, "Cr": 24, "Mn": 25, "Fe": 26,
    "Co": 27, "Ni": 28, "Cu": 29, "Zn": 30, "Ga": 31, "Ge": 32, "As": 33, "Se": 34,
    "Br": 35, "Kr": 36, "Rb": 37, "Sr": 38, "Y": 39, "Zr": 40, "Nb": 41, "Mo": 42,
    "Tc": 43, "Ru": 44, "Rh": 45, "Pd": 46, "Ag": 47, "Cd": 48, "In": 49, "Sn": 50,
    "Sb": 51, "Te": 52, "I": 53, "Xe": 54, "Cs": 55, "Ba": 56, "La": 57, "Ce": 58,
    "Pr": 59, "Nd": 60, "Pm": 61, "Sm": 62, "Eu": 63, "Gd": 64, "Tb": 65, "Dy": 66,
    "Ho": 67, "Er": 68, "Tm": 69, "Yb": 70, "Lu": 71, "Hf": 72, "Ta": 73, "W": 74,
    "Re": 75, "Os": 76, "Ir": 77, "Pt": 78, "Au": 79, "Hg": 80, "Tl": 81, "Pb": 82,
    "Bi": 83, "Po": 84, "At": 85, "Rn": 86, "Fr": 87, "Ra": 88, "Ac": 89, "Th": 90,
    "Pa": 91, "U": 92, "Np": 93, "Pu": 94, "Am": 95, "Cm": 96, "Bk": 97, "Cf": 98,
    "Es": 99, "Fm": 100, "Md": 101, "No": 102, "Lr": 103, "Rf": 104, "Db": 105,
    "Sg": 106, "Bh": 107, "Hs": 108, "Mt": 109, "Ds": 110, "Rg": 111, "Cn": 112,
    "Nh": 113, "Fl": 114, "Mc": 115, "Lv": 116, "Ts": 117, "Og": 118
}
```

```python
def electronic_configuration(element):
    """
    compute the electronic configuration of a given element
    parameters:
    element: the chemical symbol of the element. fetches the corresponding atm no from the above dictionary
    returns:
    nested dictionary representing the electronic configuration of the element: dictionary with the
    configuration of the electrons in the format of : {shell: {subshell: no. of electrons filled} }
    """
    electrons = element_electrons[element]
    config = {}
    for orbital in orbital_sequence:
        if electrons <= 0:
            break
        shell, subshell = int(orbital[0]), orbital[1]
        capacity = max_electrons[subshell]
        filled = min(electrons, capacity)
        if shell not in config:
            config[shell] = {}
        config[shell][subshell] = filled
        electrons -= filled
    return config

def display_configurations(elements):
    """
    display the electronic configurations of a list of elements in a table.
    parameters:
    elements: A list of chemical symbols of elements.
    returns: uses electronic_configuration to find the configuration of the elements in
    the list and displays it as a table using tKinter.
    """
    for widget in frame.winfo_children():
        widget.destroy()
    headers = ["Element"] + orbital_sequence
    for col, header in enumerate(headers):
        lbl = tk.Label(frame, text=header, borderwidth=2, relief="groove")
        lbl.grid(row=0, column=col, sticky="nsew")
    for row, element in enumerate(elements, start=1):
        config = electronic_configuration(element)
        element_label = tk.Label(frame, text=element, borderwidth=2, relief="groove")
        element_label.grid(row=row, column=0, sticky="nsew")
        for col, orbital in enumerate(orbital_sequence, start=1):
            shell, subshell = int(orbital[0]), orbital[1]
            value = config.get(shell, {}).get(subshell, "")
            cell = tk.Label(frame, text=value, borderwidth=2, relief="groove")
            cell.grid(row=row, column=col, sticky="nsew")

def on_submit():
    """ function to call previous function smoothly (ie, handle user clicks) """
    elements = entry.get().split(",")
    elements = [el.strip() for el in elements]
    display_configurations(elements)

root = tk.Tk()
root.title("Electronic Configuration")

frame = tk.Frame(root)
```

```
frame.pack(pady=20)

entry_label = tk.Label(root, text="enter elements (comma-separated):")
entry_label.pack()

entry = tk.Entry(root, width=50)
entry.pack()

submit_btn = tk.Button(root, text="submit", command=on_submit)
submit_btn.pack(pady=10)

root.mainloop()
```

## 3 Results

The following is a sample result of the above code:



| Element | 1s | 2s | 2p | 3s | 3p | 4s | 3d | 4p | 5s | 4d | 5p | 6s | 4f | 5d | 6p | 7s | 5f | 6d | 7p | 8s |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Xe | 2 | 2 | 6 | 2 | 6 | 2 | 10 | 6 | 2 | 10 | 6 | | | | | | | | | |
| Na | 2 | 2 | 6 | 1 | | | | | | | | | | | | | | | | |
| Kr | 2 | 2 | 6 | 2 | 6 | 2 | 10 | 6 | | | | | | | | | | | | |
| Og | 2 | 2 | 6 | 2 | 6 | 2 | 10 | 6 | 2 | 10 | 6 | 2 | 14 | 10 | 6 | 2 | 14 | 10 | 6 | |

enter elements (comma-separated):

Xe, Na, Kr, Og

Figure 1: Sample results - Electronic configuration of Elements

**Thank you**