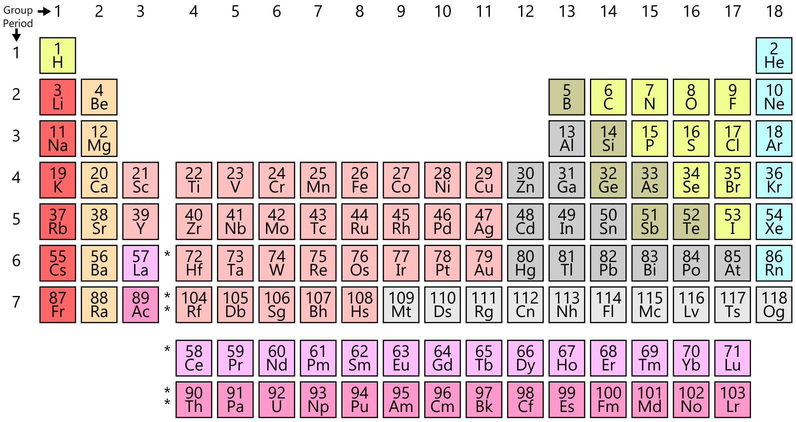
CS2204 Homework: Chemistry

Objectives

- Learn to use `namedtuple`
- Learn to process data from a text file
- Work with a list of (named)tuples
- Implement string processing algorithms


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

Background

This assignment is about chemical elements, compounds, and their modeling with computer software. You are given a database of the elements (`elements.txt`), which is a text file where each line describes one chemical element with four key properties using the following format:

```
<number> <symbol> <name> <weight>
```

where `<number>` is the atomic number (number of protons in the nucleus), `<symbol>` is the chemical symbol (one capital character, followed by zero, one or two lower case characters), `<name>` is the English name of the element (capitalized), and `<weight>` is the standard atomic weight (https://en.wikipedia.org/wiki/Standard_atomic_weight) of the element. The four fields are separated by one or multiple space characters. It is a good idea to open this file in Spyder to better understand the format.

You are going to read this database and store the information in Python objects to answer some simple and more complex questions about elements and compounds.

Tasks

You need to finish the `chemistry.py` source file. Note: you can use the simple tests provided at the end of the file (by uncommenting those lines). Feel free to change / add / delete any testing code. The outputs of this testing code are not going to be graded. The validator program will use your code directly.

1. Create a new data type , called `Element` , using the `collections.namedtuple` (meta)class. Each instance of this data type should have four fields in this order (6 pts):
   A. `number` : an integer value, the atomic number
   B. `symbol` : a string value, the chemical symbol
   C. `name` : a string value, the name of the element
   D. `weight` : a float value, the standard atomic weight
2. Finish the `load_elements` function. The existing code shows how to open a text file and read it line by line. Your task is to break up the line to four parts, convert some parts to the right data type (see above), create an `Element` instance with the parts, and add them to the global `elements` list. (12 pts)
3. Implement the `element_by_number` function. You can use the global `elements` list (created by `load_elements` function). Check the documentation string (and the example testing code) for the expected behavior. (8 pts)

4. Implement the `element_by_symbol` function. You can use the global `elements` list (created by `load_elements` function). Check the documentation string (and the example testing code) for the expected behavior. (8 pts)
5. Implement the `element_by_name` function. You can use the global `elements` list (created by `load_elements` function). Check the documentation string (and the example testing code) for the expected behavior. (8 pts)
6. Implement the `compound_elements` function. You can use the global `elements` list (created by `load_elements` function). This function should return a list of unique `Element` objects for each element present in the compound. The compound formula is a string containing a sequence of element symbols, followed by an optional number (number of atoms of a given element in the compound). If there is no number, there is one atom of the given element in the compound (25 pts).

   E.g. `"H2O"` has two Hydrogen and one Oxygen atoms, therefore you should return a list of two elements (Hydrogen and Oxygen).
7. Implement the `compound_weight` function. This function should return a float, which is the total weight of the molecule. You need to sum the atomic weights (considering the number of atoms of the given element) in the compound. The molecular formula is in the same format as described for the previous task. (25 pts)

Hints

The last two tasks (`compound_elements` and `compound_weight`) are a bit more challenging. You need to process strings character by character to find elements and an optional sequence of numbers behind these. You can use the following string methods to decide if a character is an upper case, lower case, or numeric

In [ ]:

```
ch1 = "A"
ch2 = "a"
ch3 = "1"

print(f'"ch1":', ch1.isupper(), ch1.islower(), ch1.isnumeric())
print(f'"ch2":', ch2.isupper(), ch2.islower(), ch2.isnumeric())
print(f'"ch3":', ch3.isupper(), ch3.islower(), ch3.isnumeric())
```

```
"ch1": True False False
"ch2": False True False
"ch3": False False True
```

Also, if you have a string of numeric character(s) you can convert this object to an integer, by the `int()` type/function:

In [3]:

```
str_num = "123"
num = int(str_num)
str_num, num
```

Out[3]:

```
('123', 123)
```

Grading

You can use the attached `validator.py` program to check your work (and the instructor's original mistakes). It will also estimate your final score for the homework. The program is in the same folder as your homework assignment. Open the `validator.py` script and run it in the Spyder environment to track your progress.

Penalties

Points will be deducted if you fail to set `__author__` variable (-10 pts) and for **each PEP 8 style errors** (-1 pt for each) in your program.

Submission

Please, upload the final version of the following file(s) (**and only those files**) to Brightspace:

- `chemistry.py`