Uli Wortmann Finnigan MAT 253

# Assignment

Copy this assignment and the associated image files into your submissions folder.

# Learning outcomes

* Become familiar with function arguments and return values
* Practice doc-string writing
* Practice type hinting
* Bridge the gap from seeing an equation to thinking about a task
* Bridge the gap from a task described in practical terms to python code
* Learn how to reduce complexity
* Practice testing code fragments

# Instructions

Functions have the following characteristics:

* They allow us to group code sequences and refer to this group by name. This is useful for decluttering your code.
* Code inside a function does not have access to variables defined outside of the function. This helps to isolate code sections and prevent naming conflicts or accidental overwriting of, e.g., a counter.
* The value(s) of a variable(s) can be passed into a function as arguments to the function call (see below)
* The result of computation inside the function can be returned to the calling code with the return statement.
* Functions must always be defined before you can use them. This is best done at the beginning of the code

When solving this assignment, be strategic about it. You can probably see that there are three independent challenges here:

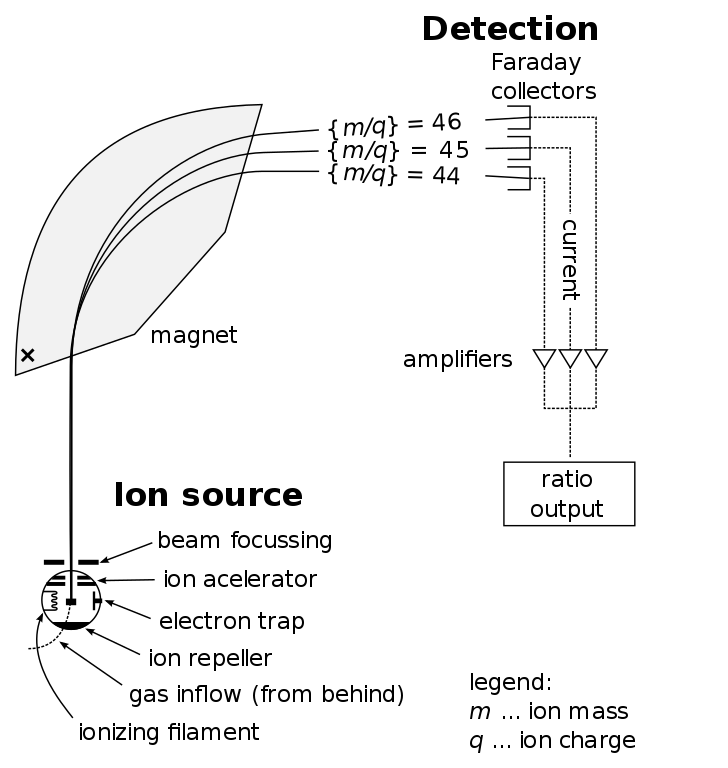
1. To create functions.
2. To embed the functions into a loop
3. To add data to an empty list.

Solve each problem individually before you combine it into a more complex program!

# Converting data from a Mass spectrometer into delta notation

Most elements have atomic structure variations, affecting their weight but not their chemical characteristics. In other words, they have the same number of protons and electrons but a different number of neutrons. You will likely have heard of oxygen or carbon isotopes, both featuring prominently in the current climate change debate and isotopes featuring prominently in almost all geoscience research.

Stable isotopes ratios are measured with a mass spectrometer where the oxygen atoms are ionized, then accelerated, and the beam of accelerated ions is then sent through a magnetic field, which will bend this beam. The ions with additional neutrons are heavier and thus will have a different curve radius than the lighter ones (see Fig. <fig:IRMS>). Therefore we can split the beam into two beams. The two beams will then be collected by Faraday cups, which count the number of arriving ions (i.e., they register a voltage). Based on these voltages, we can establish the ratio between these isotopes.

 Figure 4.1: Schematic drawing of an isotope ratio monitoring mass spectrometer (IRMS). Source: <https://commons.wikimedia.org/wiki/File:Mass_Spectrometer_Schematic.svg>, 2019

Most geological processes change the isotope ratio of a given element only by a small fraction. Consider the following example, which uses sulfur isotopes from seawater sulfate. Sulfur has 4 stable isotopes (32S,  33S,  34S,  36S), and one unstable (i.e., radiogenic) isotope (35S). Here we will stick to the two most abundant isotopes  32S and  34S. Each measurement generates a value that corresponds to the concentration of ^{32}S and ^{34}S in a given sample. The following data is from actual measurements of seawater sulfate samples. The index position equals the sample number.

# remember that variable names cannot start with a number!

S32 :list = [0.956825467106151, 0.956824254162342, 0.956831127551253,

0.956806868972346, 0.956808486172672, 0.95680282599545,

0.957705256379378, 0.956814955028641, 0.957705256379378,

0.957705256379378, 0.956929791779426, 0.957705256379378,

0.957705256379378, 0.956975491533205, 0.957000163125976,

0.956964976158995]

S34 :list = [0.043174532893849, 0.043175745837658, 0.043168872448747,

0.043193131027654, 0.043191513827328, 0.04319717400455,

0.042294743620622, 0.043185044971359, 0.042294743620622,

0.042294743620622, 0.043070208220574, 0.042294743620622,

0.042294743620622, 0.043024508466795, 0.042999836874024,

0.043035023841005]

From the above, you can see that there is more S32 than S34. You can also see that if we only look at the ratios between S32and S34, the numbers are unwieldy, and it is hard to spot the change between two values. I.e.,

print(f"34S/32S [0] = {S34[0]/S32[0]}")

print(f"34S/32S [4] = {S34[4]/S32[4]}")

It is, therefore, customary to express the change in isotope ratio as a difference relative to a standard value. The unit of the delta notation is "per mil" which translates as "per thousand" (or 0.1 %)

\begin{equation}

\delta^{34}S = \left(

\frac{

\left(\frac{34S}{32S}\right) \_{Sample}}

{

\left(\frac{34S}{32S}\right) \_{VCDT}}

-1

\right) \times 1000 \quad [^0/\_{00}]

\end{equation}

For sulfur, the standard value is a meteorite, the Canyon Diabolo Troilite. Since this standard has long been depleted, we nowadays use a virtual value, the so-called "Vienna Canyon Diabolo Troilite" (VCDT). The reference ratio of  34S/32S for VCDT is

R :float = 0.044162589 # Reference ratio of 34S/32S for VCDT

# Question 1

Create a function that takes two numbers as arguments (say S32[i] & S34[i]) and returns the respective delta value to the calling program. To keep the function universal, pass the reference ratio as the argument. So your function interface will look like this

def v2d (s32 :float, s34:float, R:float) -> float:

Note, that the above implies that s32 and s34 are single values, rather than a list. We will explore passing a list later.

Write this function (including doc-strings) and embed it into a loop that iterates over each element of S32 and S34 and then calls v2d for each isotope pair. Store the returned delta values in the new list delta. Print the resulting values using this template before starting the next iteration.

S34 = and S32= yield a delta value of = XX.XX permil

Note the explicit print format for delta. Also, remember that the print statement should be in your main code, not inside your function!

To solve this assignment, you will have to create a new (and empty) list, and then append the results to this list. The following code snippet will be helpful for this. See the previous module on lists if you do not recall how to append data onto a list

delta: list = [] # this will create an empty list

delta: list = list() # this will do the same

The goal of this exercise is to make you think about the function arguments, how to use multiple arguments, and how to go from an equation to a function. Remember the tips from the last module: Keep it simple, test individually, and use a test case with only a few values.

Before you go ahead and start coding consider the following:

* Does the assignment ask to pass the list to the function, or an individual value? Look at the type hints in the above function definition.
* What steps do you need to take to solve this assignment? Write this down without worrying about how to do this in python. Your notes could look like this:
  + Define v2d and test that values are computed as intended
  + Declare an empty list that will hold the results
  + Loop over all list elements
    - Access the individual list elements by index
    - Call v2d(list1[i],list2[i]) for each list element
    - append the result to the empty list created above

Each of the techniques has been used before. If need be, make good use of the textbook, and test each step before going to the next one. Some delta values will be minimal (basically zero), but most should be around 18 to 22 permil.

# Question 2

We can invert the above equation and calculate the respective isotope concentrations from the delta value we stored in delta (lets call this function d2i). To keep the function universal, I will call the respective isotopes simply li for 'light isotope' and hi for 'heavy isotope'

\begin{equation}

li = \frac{1000}{(\delta +1000) \times R + 1000}

\end{equation}

\begin{equation}

hi = \frac{(\delta + 1000) \times R}{(\delta + 1000) \times R + 1000}

\end{equation}

write a function that will take a delta value and returns the light and heavy isotope values. Then write some code that will use a loop to iterate over each delta value in the delta list and call your function to compute the light and heavy isotopes. Store the returned values in the new lists S32\_new and S34\_new.

Note that you should use a single function that returns both values. If need be, review the intro module.

# Question 3

Use a single loop to compute the element-by-element difference between S32 and S32\_new. Within the same loop, do the same for S34 and S34\_new. Add the difference to errS32 (errS34 respectively) to calculate the total accumulated error.

\begin{equation}

err\_{S32} = \sum\_{0}^{n} S32(i)-S32\_{new}(i)

\end{equation}

where n denotes the number of elements in S32. Your total error should be a small number (i.e., 1-16 etc.).

# Marking Scheme

* Your code uses the correctly defined functions (docstrings, and type-hinting, parameters) 2 \* 3 = 6 pts
* Your code calculates the requested quantities 3 \* 2 = 6 pts.

Total: 12 points

Create a new (or copy and existing) notebook in your submissions folder before editing it. Otherwise, your edits may be overwritten the next time you log into syzygy. Please name your copy "Assignment-Name-FirstName-LastName":

* Replace the Assignment-Name with the name of the assignment (i.e., the filename of the respective Jupyter Notebook)
* FirstName-Last-Name with your own name.

Note: If the notebook contains images, you must also copy the image files!

Your notebook/pdf must start with the following lines

Assignment Title

Date:

First Name:

Last Name:

Student: Id

Before submitting your assignment:

* Check the marking scheme and ensure you have covered all requirements.
* re-read the learning outcomes and verify that you are comfortable with each concept. If not, please speak up on the discussion board and ask for further clarification. I can guarantee that if you feel uncertain about a concept, at least half the class will be in the same boat. So don't be shy!

To submit your assignment, you need to download it as ipynb notebook format and pdf format. The best way to export your notebook as pdf is to use your browser's print function (Ctrl-P) and then select Save as pfd. Please submit both files on Quercus. Note that the pdf export can fail if your file contains invalid markup/python code. So you need to check that the pdf export is complete and does not miss any sections. If you have export problems, don't hesitate to contact the course instructor directly.

Notebooks typically have empty code cells in which you must enter python code. Please use the respective cell below each question, or create a python cell where necessary. Add text cells to enter your answers where appropriate. Your responses will only count if the code executes without error. It is thus recommended to run your solutions before submitting the assignment.

Note: Unless specifically requested, do not type your answers by hand. Instead, write code that produces the answer. Your pdf file should show the code and the results of the code execution.