Python warm-up/recap

-Getting used to python and numpy.

-Creating, using and modifying objects in python.

To get ready for the IO calculations, in this exercise, more emphasis is placed on integers, floats and lists.

0: Install python and numpy (NumPy is a python package that was created specifically scientific computing in python. It is optimized for creating, handling and calculating with n-dimensional arrays.)

1.

a) Divide the integer 3 by the integer 5 in python. What is the object type of the result? What is the object type if you multiply instead?
b) Create a float and multiply it by an integer. What is the object type of the result?
c) Create a list of three integers and a float and calculate the sum of the list.
d) Retrieve the second item in the previously created list.

2.

a) Create an array (array_one) from a list of integers.
b) What happens when you do array_one + 3? And array_one / 5?
c) Create an array (array_two) with two rows and three columns of integers and floats.
d) Find the sum of all values in array_two.
e) Find the sum of values in each row of array_two. How did the dimensions of the resulting array change?
f) Find the sum of values in each row of array_two while retaining the dimensions.

Python code guide for the exercises after the Week 3 lecture

Display an array ***A***	print(***A***)
Import numpy library as '***np***'	import numpy as ***np***
Create an array ***A*** = $\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$	***A*** = np.array(***[[1, 2, 3], [4, 5, 6]]***)
Display the shape of an array ***A***	np.shape(***A***)
Matrix multiplication, e.g. ***A***×***B***	np.dot(***A***, ***B***) or ***A*** @ ***B***
Diagonalizing a vector *a, e.g.* $\begin{bmatrix}1\\2\end{bmatrix}$ to $\begin{bmatrix}1 & 0\\0 & 2\end{bmatrix}$	np.diag(***a***)
Invert a matrix ***A***	numpy.linalg.inv(***A***)
Transpose an array ***A***	numpy.transpose(***A***) or ***A***.transpose()
Collapse a 2D array ***A*** to 1D	***A*** = ***A***.flatten() or ***A***.flatten()
Create an identity matrix, e.g. $\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$	np.identity(***2***)
Create a vector of '1', e.g. $\begin{bmatrix}1\\1\end{bmatrix}$	np.ones((***2,1***))
Element-wise matrix division, e.g. $\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$/[2 2 3] = $\begin{bmatrix}0.5 & 1 & 1\\2 & 2.5 & 2\end{bmatrix}$	***A***/***B***