Exercises Week 1

For any function you haven't met before, you can use ? to find out information, e.g. print? You will need to hit *q* to exit from the help.

First you will need to open Pycharm and then import *numpy* to complete the following exercise. The following exercises can be completed in the terminal or written in a script.

Basic exercises

1. **Mathematics Operators**

    a. Print the product of 7 and 3
    b. Assign the answer of the previous product to a variable and print the type of that variable.
    c. Print the division of 7 and 3 and display its type.
    d. Print the integer division of 7 and 3 and display its type.
    e. Apply the cosine operator to the value 2 and then apply the inverse cosine to the answer.
    f. Apply the cosine operator to the value 4 and then apply the inverse cosine to the answer. How does this differ to what happened in e) and why?
    g. Print the square of 5.
    h. Find the remainder of 18 divided by 4 using the modulo operator.
    i. Here we are going to calculate the interest earned on your savings. Create a variable called *savings* and assign it a value of a 100. Write a small script to calculate the amount earnt after 7 years if the interest rate is 2%. Assign each number to a variable, such that the final calculation is written as '*profit=savings *interest_rate**years*' .

2. **Logic and Relation Operators**
    Using the logical operators, find the boolean values of the following
    a. Whether 9 is equal to 3.
    b. Whether 9 is equal to 9.
    c. Whether 9 is not equal to 3.
    d. Whether 9 is not equal to 9.
    e. Whether 9 is greater than 3
    f. Whether 9 is greater than 9.
    g. Whether 9 is greater than 3 and less than 13.
    h. Whether 9 is greater than 3 or less than 7.

3. **Strings**
    a. Assign the string 'hello' to a variable called greeting and the your name to a variable called name. Combine the two strings with the + operator and print the result.
    b. Create another string that says 'My age is ' , and a variable age and assign it your age. Try to combine the variables.
    c. Using the function *str(),* convert the age variable to a string and try combining both again.

4. Lists

a. Assign each of the following stellar masses to variables named after the stars. Using these variables, create a list of the star's masses from largest to smallest. These stars are some of the most massive stars known (mass given as solar masses):
R136a1 - 315 M_o, R136c - 230 M_o, BAT99-98 - 226 M_o, R136a2 - 195 M_o, Melnic-42 - 189 M_o .

b. If we just print the above list to screen, it is not very useful as it only contains masses. Create a list of lists, with each sub-list containing the name and mass of the star.

c. Print out the second element of list. You should get back [R136c, 230].

d. Print out the last element of the list using negative indices.

e. Use list slicing to create a new list of the 3 most massive stars.

f. From the list, print the mass of Melnic-42 only.

g. Add the details of HD15558A – 152 M_o, to the list.

h. Delete the value of R136a1 from the list using the *del()* function.

5. Arrays (Remember to import the numpy module before starting these exercises)

The following table gives the details of some stars to be used in the following exercises.

Class	**Radius (R_o)**	**Mass (M_o)**	**Luminosity (L_o)**	**Temp (K)**	**Name**
O6	18	40	500,000	38,000	Theta1 Orionis C
B0	7.4	18	20,000	30,000	Phi1 Orionis
B5	3.8	6.5	800	16,400	Pi Andromedae A
A0	2.5	3.2	80	10,800	Alpha Coronae Borealis A
A5	1.7	2.1	20	8,620	Beta Pictoris
F0	1.3	1.7	6	7,240	Gamma Virginis
F5	1.2	1.3	2.5	6,540	Eta Arietis

a. Create a list that contains the numbers radius of the stars, and then use the *numpy array* function to make this into a *numpy* array. Print out the type of the variable to check it is a *numpy* array.

b. Create another array that contains the mass of the stars.

c. Create a new array that takes the previous radius array and calculates the volume of each star (assuming sphericity). Print out the results.

d. Create a new array that contains the calculation of each star's density, using the volume and mass arrays. Print out the results.

e. Find the maximum, minimum and mean values of density using either the array methods or the *numpy* functions.

f. Create a boolean *numpy* array that contains the elements of *True* if the star's density is greater than the mean value and *False* otherwise (you will have to use relational operators). Print out the array.

- **g.** Use the boolean *numpy* array to select the values of the star's mass whose density is greater than the mean.
- **h.** Print out the mass value of the star at index 3. Then print out the mass value of the stars up to and including index 5.

6. <u>Arrays Pt II</u>

- **a.** Create an array of zeros that contains 20 elements. Print its shape and size.
- **b.** Create an array that contains elements from 0 to 19 in increasing order (hint: use the *numpy arange* function).
- **c.** Reshape this array into a 2d array that is 4 by 5. Print its shape and size.
- **d.** Print the value at the second column, third row.
- **e.** Create an array as you did in part b) and now make it a 5 by 4 array. Try adding this to the array created in part c).
- **f.** Create a copy of the array (a unique copy) from part c. Multiply the values by 2. Divide the previous array from part c) by this new array and print the results.

Harder exercises

You will have to use the help function or web-searches to find out how to use the new functions.

<u>Array concatenation</u>

7. Create two integer arrays of length 10, called a & b.
   - (i) Combine the two arrays to make a 20 by 1 array.
   - (ii) Combine them to make one array of dimension 10 by 2.
   - (iii) Combine this last array with itself to make an array of dimension 10 by 2 by 2.

<u>Array Manipulation</u>

8. Create an array of 1000 elements using *np.random.randn.* Use the *np.where* function to find out:
   - (i) How many elements are greater than 0.
   - (ii) How many elements are less than 0.
   - (iii) How many elements lie between -1 and 1. Is this number familiar?
   - (iv) What is the results when you look for numbers greater than 100?

9. Create an ordered array of length 40.
   - (i) Find the location of the even numbers using *modulo* function.
   - (ii) Find the location of the odd numbers.
   - (iii) Which elements have a factor of 3?
   - (iv) Which elements have a factor of 4?