Functional Programming Exercise

Shared Exercises 3

These exercises test your ability to write modular, functional code. Each exercise requires you to write a function that may or may not take arguments, and may or may not return a result. The number of arguments may be fixed or variable.

Note, that each of these exercises (as pretty much any programming problem) has multiple correct solutions. It can be very instructive to implement several of them and discuss what the advantages and disadvantages they might have.

Exercise 1

Write a function called intCounter, that takes in a list of numbers (integers). The list can have any length. Count how many times each number appears in the list, and print it out. If you come across an element that is not a number, print an error message instead.
Funtion intCounter takes in a variable length of arguments, and has no return value.
Some examples of what the output should look like are:

intCounter(1, 1, 1, 3, 1, 4, 1, 1, 3)
The number 1 appears 6 times.
The number 3 appears 2 times.
The number 4 appears 1 times.

intCounter(1, 1, 1, 'a', 'b', 'c')
The number 1 appears 3 times.
a is not a number.
b is not a number.
c is not a number.

Hint 1.1

You have to check how many times each UNIQUE element appears in the list. One way of doing that could be to make a new list of uniques, and use this new list to count how many times each of the elements appears in the original list.

Hint 2.2

To count how many times an element appears in a list, you can use the .count() method. Look this up.

Exercise 2

Write a function called mode. Similar to the previous function, this function takes in a list of numbers (integers). The list can have any length. Count how many times each number appears in the list. Return the number that appears the most amount of times.
Some examples of the output:
result = mode(1, 1, 1, 3, 1, 4, 1, 1, 3)

print(result)

1
result = mode(5, 5, 5, 2, 3, 'a', 'a', 'a', 'a')

print(result)
5

Exercise 3

Write a function called neatPlotter. This function takes in two Python lists as arguments. The function checks that both lists have numbers in them, and both lists must have the same size. If this is the case, the function uses the matplotlib library, to plot one list against the other. The function then adds the following:

- a label to the x axis "X values".
- and a label to the y axis "Y values".
- a title "Neat plot".
- a grid.

So, the function neatPlotter takes in two arguments, both lists of integers, and has no return value.

Exercise 4

Write a function called circleFrame. This function has two arguments - the path to an image file, and a number "r". Use the openCV library methods to read the image at that path. Read the image in black and white mode (i.e. only one channel). If the image does not exist, print an error message. If the image exists, and you can read it into memory without errors, do the following:

- Make a grid using the numpy mgrid function. The grid must go from -img.shape[0]/2 to img.shape[0]/2 in the first dimension and -img.shape[1]/2 to img.shape[1]/2 in the second dimension. For example, let's say the image is 100 pixels wide and 80 pixels tall. Then the grid must go from -50 to 49 in the wide dimension, and -40 to 39 in the tall dimension.
- Create a new array, which contains a circle with the equation $x^2 + y^2 < r^2$. Here, "r" is the second argument that you pass to the function. Call this array "mask".
- Multiply the mask array with the image. This should mask out all of the image except for a circle at the center. Display this image using matplotlib. If all goes well, you should see the central part of the image in a circle, and black/empty pixels around this circle.

In other words, function circleFrame takes two arguments, a string containing the image path and a number. It has no return value.