

## Homework 2

Use docstrings and good style, and don't forget to test your code! Write your solutions in a file named `yourname_homework2.py` and submit to [m.spacek@lmu.de](mailto:m.spacek@lmu.de) before class 04 (May 8).

Hint: the solutions to all of these are no more than a few lines each. If your solution is long and complicated, you're probably doing it the hard way!

1. Your experiment collects  $N$  measurements. Write a function called `norm()` that accepts a list of values of arbitrary length  $N$ , and returns a list of the normalized values (i.e., each value divided by the sum of the values).

e.g. `norm([1, 6, 0, 3])` returns `[0.1, 0.6, 0.0, 0.3]`

- do you have to do anything different for your function to also accept a tuple instead of a list? What about to make it return a tuple instead of return a list?
- try to write your function using only two lines of code (not including the `def` line and the docstring)

2. Take the following data (a list of lists, one experiment per row) and apply your `norm()` function to create a normalized version of the data in a new list of lists called `normdata` :

```
data = [[9.1, 2.1, 0.9, 1.5, 1.1],
        [4.4, 2.2, 3.3, 5.5, 6.6],
        [0.1, 0.2, 0.3, 0.4, 0.5]]
```

- can you do it in one line?
- check that the normalized values for each experiment in `normdata` really do add up to 1.

3. Write a function called `vectorsum()` that returns the vector sum of two lists, i.e., the sum of the values at the corresponding positions in two input lists. Example:

```
x = [2, 3, 4, 5, 0, 0, 0, 2, 2, 0]
y = [0, 4, 2, 4, 5, 1, 0, 5, 3, 5]
```

`vectorsum(x, y)` returns `[2, 7, 6, 9, 5, 1, 0, 7, 5, 5]`

Hint: use the function `zip()` (added to class 03 notes) to iterate over both lists at the same time, e.g. `for xx, yy in zip(x, y):`

- what happens with `zip()` if the lists aren't the same length?

4. The measurements in your experiment in question 1. now have exciting names, e.g. `'a'`, `'b'`, `'c'`, `'d'`. Write a function called `normd()` that takes a dictionary with an arbitrary number of key:value pairs, and returns a dictionary with the same keys, but with normalized values. Example:

`normd({'a':1, 'b':6, 'c':0, 'd':3})` returns `{'a':0.1, 'b':0.6, 'c':0.0, 'd':0.3}`

Hint: some dict methods like `.keys()`, `.values()` and `.items()` will be very useful

- try to write your function using only two lines of code (not including the `def` line and the docstring)