Practice 2.1

Practice sheets are not assessed. The intention is to use material from lectures in preparation for Competence tests and Assignments. You are encouraged to use `thing.[TAB]` and `thing?` in IPython.

Data structures [EASY]

tuples and list

Make a list of tuples, where each tuple contains somebody's name and their age.

dictionaries

Make a dictionary which associates people's names with their ages.

Add an entry to the dictionary. Change an entry. Remove an entry (using `thing.pop`).

sets

Here are two sets:

```
s = {1,2,3}
r = {2,4,6}
```

Add/remove elements using `s.add` / `s.remove`. Find the intersection using the method or the operator `-`. Find the union using the method or the operator `|`.

What does `s ^ r` *do?*

For other operations, such as the Cartesian product, see `itertools` *library.*

Functions and multiple returns [EASY]

Write a function which returns both the first, second, third, and fourth power of a given number i.e.

$$f(x) = (x, x^2, x^3, x^4).$$

Calculate this for $x = 2$ and store the result in `a,b,c,d`.

Integration [MEDIUM]

Use `trapz` or `quad` to evaluate the definite integral $\int_0^\infty e^{-x^2} \cos x \, dx$.

Write a *function* which computes the integral $F(x) = \int_0^x e^{-x'} \, dx'$ and plot $F(x)$ for $0 \leq x \leq 10$.

Lists and methods [MEDIUM]

self reference

Make a list and use the `append` method to make the list contain itself. Try indexing it. Does anything break? Could this ever be a useful thing to do?

sorting

Make a list of numbers. Sort it using the `sort` method. What is different between the `sort` and `sorted` methods?

Make a list of people's names. Sort it by length using the `key` optional argument to `sort` and the function `len` to measure the length of a string.

Primes [HARD]

Using Python sets, write a function `sieve` which implements the Sieve of Eratosthenes and returns all prime numbers less than a given positive integer N.

Triangle sums [VERY HARD]

Project Euler problems 18 and 67 can be solved with a functional programming approach: the maximum sum, for a given starting node, is the value of that node plus the greater of the maximum sum which one can reach from either of the two nodes below. Be sure to **record and reuse results** when you calculate them, otherwise you'll end up the same way as the functional implementation of the Fibonacci sequence.

The attached code may help to get you started.