Training Assignment 13

NUMA01: Computational Programming with Python

Malin Christersson, Robert Klöfkorn

In this exercise we work with tests, both `unit tests` and `pytest`.

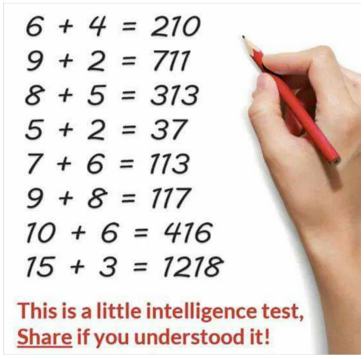
Tasks

This assignment has 6 tasks.

Task 1

Implement a function `crazzyPlus(a,b)` which sums two integers in a way as described in the figure a friend found in a newspaper[1] a while ago.

Decide for which range of arguments a, b you want to design your function. Raise



an ValueError if your function is called with wrong arguments.

Task 2

What is the time complexity of the algorithm that you designed?

Task 3

Design and write tests for this function. Either use `unittests` or `pytest` for this (see lecture notes or book). You should test:

[1] Unfortunately I got this without any citation.

- The correct result for several test cases
- That the exception is raised, when wrong arguments are used.

Task 4

Use the command `%timeit` and `%%timeit` to obtain some timings about the code that you wrote. Profile your program using the `%lprun` command in the IPython shell. Consult the lecture slides for explanation about `%lprun`.

Extra Task 5 (in case you want an extra challenge)

You are given an `array` of N random positive integers, i.e. $i \in \mathbb{N}$. Find the smallest positive number missing from the array.

The expected time complexity is $\mathcal{O}(N)$. Check that your implementation fulfills this.

Extra Task 6 (in case you want an extra challenge)

Create an $n \times m$ matrix randomly filled with the values 0 or 1. Imagine that 0 stands for water and 1 means land. Find the number of islands.

Note: An island is either surrounded by water or at the boundary of the matrix and is formed by connecting adjacent lands horizontally or vertically or diagonally i.e., in all 8 directions.

Write a code that can find all islands given in a domain (matrix).

Example: This domain has 3 islands.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{1}$$

The expected time complexity is $\mathcal{O}(n\,m)$. Experimentally check that your implementation fulfills this.