MAT101 Programming – Homework 7

Deadline: Monday, 20.11.2023, 22:00

Login to `https://w3.math.uzh.ch/my` with your UZH credentials to submit your solved exercises for grading. You can find more information on how to upload/submit your exercises on `https://wiki.math.uzh.ch/public/studentUpload`.

💡 For submission, please upload **at most 1** Python file **per exercise**. You could even just upload 1 Python file for the whole exercise sheet. You can use comments and/or print statements to answer non-programming tasks.

Exercise 1. **10** P.

(Warm-up)

💻 In mathematics, the *Fibonacci numbers*, commonly denoted by F_n, form a sequence, the *Fibonacci sequence*, in which each number is the sum of the two preceding ones. The sequence commonly starts from 0 and 1. In other words, the n-th Fibonacci number F_n is *recursively* defined by

$$F_0 := 0, \quad F_1 := 1$$
$$F_n = F_{n-1} + F_{n-2}, \quad \text{for } n \geq 2.$$

a) Write a function `fibonacci_recursive(n : int)` which takes as an argument an integer $n \geq 0$ and returns the n-th Fibonacci number using the recursive definition described above. (**3** P.)

b) Using `fibonacci_recursive`, what is the value of F_{19}? What is the value of F_{38} and what do you observe with respect to the time it takes to compute this number? (**3** P.)

c) Design and implement an iterative algorithm (using a for- or while-loop) that returns for a given integer n the n-th Fibonacci number. Call your function `fibonacci_iterative`. (**4** P.)

Exercise 2. **10** P.

💻 Let F_n denote again the n-th *Fibonacci number*. We have $F_0 := 0$, $F_1 := 1$ and $F_{n+1} = F_n + F_{n-1}$ for all $n \geq 1$. Let us now define

$$A := \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \qquad \text{then } \begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = A \begin{pmatrix} F_n \\ F_{n-1} \end{pmatrix}.$$

Thus we can produce a vector whose coordinates are two consecutive Fibonacci numbers by applying A several times to the vector $u_1 = (1, 0)^{\mathrm{T}}$, i.e.

$$\begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix} = A^n \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Using this insight and the Numpy package, write a function `fibonacciMatrix(n : int)` that takes as an input an integer $n \geq 0$ and returns the n-th Fibonacci number.

Which output do you observe for `fibonacciMatrix(101)`? Can you explain said output?

Exercise 3. **10** P.

💻 This exercise is intended to make you familiar with multidimensional np.ndarrays.

a) use np.ones() to create a 3×3 matrix filled with 1. **1** P.

b) use np.eye() to create the identity matrix of size 3. **1** P.

c) add the two matrices from a) and b). **1** P.

d) multiply the identity matrix of size 3 with the vector $(7, 11, 2022)$ **1** P.

e) use NumPy functions to create the following matrix as multi-dimensional np.array **6** P.

$$\begin{matrix} 1 & 0 & 4 & 4 \\ 0 & 1 & 4 & 4 \\ 0 & 0 & 19 & 0 \\ 0 & 0 & 0 & 2 \end{matrix}$$

Exercise 4. **10** P.

Monte Carlo methods are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use *randomness* to solve problems that might be deterministic in principle. One of the basic examples of getting started with the Monte Carlo algorithm is the estimation of π.

Consider the square of side-length 2 centered around $(0, 0)$, i.e. $D = \{(x, y) \in \mathbb{R}^2 : -1 \leq x, y \leq 1\}$ and the unit disk centered around $(0, 0)$ of radius $r = 1$, i.e. $R = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leq 1\}$. It then holds that

$$\frac{\text{Area of circle}}{\text{Area of square}} = \frac{\pi}{4}. \tag{1}$$

Generating a very large number of *random* points in D, equation 1 can be interpreted as

$$\frac{\text{Total number of points generated inside the circle}}{\text{Total number of points generated}} \approx \frac{\pi}{4},$$

where with *inside the circle* we mean all $x, y \in \mathbb{R}^2$ such that $x^2 + y^2 \leq 1$.

a) Write a function `monteCarloPi(N : int)` that for a given integer $N \geq 1$ returns π_{approx} according to the Monte Carlo method described above. (**5** P.)

b) Using the same approximation method as outlined above, generalise your function of part a) to a function called `monteCarloSphere(N : int, d :  int)` that approximates the volume of a d-dimensional sphere. Sampling with $N = 10'000$ points, report your findings for dimensions $d \in \{3, 4, 10, 100, 300\}$. (**5** P.)

Hint: Read the documentation for `np.random.uniform`