Homework 1

Please write your code cleanly and tidily. Clarify your code with readable and pronouncable variable and function names. Use comments to provide information that cannot be expressed or obtaiend with the programming language. Try to use names rather than comments to express what the code is doing.
Please write your functions and submission code into a file `specom_homework_1.py`.

Exercise 1 Polarize a matrix (3 Points)

Make a function `polarize_matrix(matrix: numpy.ndarray) -> numpy.ndarray`.

It takes a 2D array of any shape as input and converts each of its elements to either -1 or 1 according to the following rule: if the element is a negative number, convert to -1, else convert to 1. The result should be a matrix consisting of only ones and minus ones.
Tip: you only need one `numpy` function.

As a simplistic test case, please create a random matrix with values shifted into the range of -0.5 and 0.5. Call `polarize_matrix` with a 5×10-matrix as input.

Exercise 2 Simulation of coin tosses (3 Points)

Please write the function `coin_toss_probability(number_of_trials: int) -> float` that simulates a number of consecutive coin tosses.

One coin toss is a random variable whose value is eiher 0 ("tails") or 1 ("head"). Evaluate one coin toss by generating a random decimal number R. If the random number is ≥ 0.5, it will count as "heads". Therefore, the probablistic events are defined as $\text{tails}_R = [0, 0.5]$ and $\text{heads}_R = [0.5, 1]$.

Return the relative frequency of having tossed "heads" over the `number_of_trials`-many coin tosses.

What is the value of `coin_toss_probability(500)`?

Exercise 3 Pascal's Triangle (4 Points)

Pascal's Triangle names a scheme for computing binomial coefficients. For this exercise, implement a function `pascals_triangle(rows_count: int) -> numpy.ndarray` which computes `Pascal's Triangle` as a matrix M where

$$M_{i,j} = \begin{cases} 0 & \text{if } j > i \\ 1 & \text{if } j = 0 \vee i = j \\ M_{i-1,j-1} + M_{i-1,j} & \text{else} \end{cases}$$

Start at $i, j = 0, 0$ and fill the matrix row by row. Tip: use `numpy.ndindex(number_of_rows, number_of_columns)` to iterate over the indices. Use `dtype=int` as argument when you create numpy arrays otherwise your results will be using `float` which loses precision compared to `int`.

a) Generate a Pascal's Triangle with 50 rows. In the result, shift each column down by the number of elements equal to the column index and afterwards compute the sums of each of the remaining 50 rows. You could also sum up the elements equivalently without shifting the columns, e.g. using `numpy.diagonal()`.

   What is the name of the result sequence?

b) With the sequence of sums F from a), compute the ratio $\frac{F(n)}{F(n-1)}$ between adjacent values.

   It has been found that this ratio converges to the value of the golden ratio $\varphi = \frac{1+\sqrt{5}}{2}$.
   Tip: the golden ratio is available in `scipy` as `scipy.constants.golden`.

   After how many ratios in the sequence do we get a ratio which coincides with the golden ratio by the first 5 digits?