MAT101 – Programming with Python
Mock exam 1 - 27.11.23 (due date: 11.12.23)

Gauthier Wissocq

Institute of Mathematics, University of Zurich

For each exercise, produce a script named 'Exercise_n.py', with n number of the exercise. Appropriately comment your code and add documentation in functions to explain what they do, specifying inputs and outputs. Note that in your final grade, 4 points are attributed to the overall comments of your code and documentation of functions.

Exercise 1 (12p)

a) (6p) Write a function called `'is_bigger'` which takes as input a vector of numbers (list or one-dimensional ndarray) v and a threshold t. The function returns the list of the indices of the elements strictly bigger than t in the input vector v.

b) (6p) Write a function called `'average_rows'` which takes as input a matrix M (numpy bidimensional ndarray) and returns a one-dimensional ndarray with size equal to the number of rows of the matrix M, containing the average values of each row of M.

Exercise 2 (36p)
Consider the sequence

$$a_k = \frac{(-1)^k}{k+1}, \tag{1}$$

defined for $k \in \mathbb{N}$, and the corresponding partial sum $S_n = \sum_{k=0}^{n} a_k$. It can be shown that S_n converges to $\log(2)$ when $n \to \infty$. The aim of this exercise is to use this sequence to have an approximation of $\log(2)$.

a) (7p) Write a function called `'partial_sum'` which takes as input a variable n, supposed to be an integer, and returns the value of the partial sum S_n. If $n \notin \mathbb{N}$, the function should return the following message error: `"Error:  the argument should be a natural number"`.

b) (5p) Write a function called `'convergence_vector'` which takes as input a list v of integers and gives as output another list containing the partial sums whose indices correspond to the elements of v. You should use the function `'partial_sum'` for this question.

c) (10p) When calling the function `'partial_sum'` for each integer in the list v, the function `'convergence_vector'` is far from optimized. For example, with $v = [100, 200]$, the computation of S_{200} involves computations that are already done in S_{100}. Write a function called `'convergence_vector_opt'` which is similar to `'convergence_vector'`, but with a single loop until the maximum index contained in v. This would reduce (very) much the computational cost.
Note: suppose that the indexes in v are ordered.

d) (5p) Write a function `'time_comparison'` which takes as input a list v of integers and prints the respective computational times of the functions `'convergence_vector'` and `'convergence_vector_opt'`, when called with v as argument.
Note: this question can be addressed even if question c) was not answered.

e) (9p) Plot the partial sums S_n corresponding to the indices $n = 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024$ (that can be computed through the function `'convergence_vector'` or `'convergence_vector_opt'`) with respect to their indices. In particular

- plot these values as discrete points (no lines between them);
- use the logarithmic scale for the x-axis;
- in the same figure, plot the horizontal line of $\log(2)$ in dashed line;
- add a title on the top of the plot;
- add a label for the x-axis;
- add a legend;
- add a grid;
- save the plot as `'plot_convergence.pdf'`.

Exercise 3 (28p)
We consider polynomials $P(X)$ defined as

$$P(X) = \sum_{k=0}^{d} c_k X^k,$$

where d is the degree of the polynomial and c_k are its coefficients. We recall that the derivative of P can be computed as

$$P'(X) = \sum_{k=0}^{d-1} c_{k+1}(k+1)X^k,$$

For example, with $P(X) = 1 + 2X^2$, we have $d = 2$, $c_0 = 1$, $c_1 = 0$, $c_2 = 2$ and $P'(X) = 4X$. In this exercise, we denote a polynomial P by the list of its coefficients, *i.e.* $P = [c_0, c_1, \dots c_d]$. With the above example, $P = [1, 0, 2]$.

a) (6p) Write a function `'degree'` which takes as input the list of coefficients P and returns the degree of the polynomial.

b) (4p) Write a function `'eval_polynomial'` which takes as input a polynomial P (list of coefficients) and a float x, and returns $P(x)$, the evaluation of the polynomial in x.

c) (6p) Write a function `'polynomial_derivative'` which takes as input a polynomial P (list of coefficients) and returns the list of coefficients of its derivative P'.

d) (12p) (This question is about classes and objects, which will be covered on the 04.12.23). Define a class `'Polynomial'` whose constructor takes as input a list of coefficients and defines two attributes:

- an integer d referring to the degree of the polynomial,
- a list $coefs$ referring to the list of coefficients of the polynomial.

Furthermore, add the following methods in the class:

- a method called `'derivative'` which returns an object of the class `'Polynomial'`, whose coefficients are the coefficients of the derivatives of the considered polynomial;
- a method designed to overload the operator $+$, such that the addition of two polynomials returns a polynomial whose coefficients are the sum of the coefficients. For this question, we assume that the two polynomials that we sum always have the same number of coefficients.