Homework - Serie 08

Kevin Sturm
Python 3

Test your code with examples! Below the word function refers to python function.

Problem 1. Define the decorators of exercise 4/5 of the previous exercise sheet with the help of the @ operator. Explain what the syntax does. What do `@staticmethod` and `@classmethod` do ? What is the difference to a normal class method with a self argument. Write a class of your choice illustrating the difference between these three types of class methods.

Problem 2. Write a class `poly` which gets a list `coeff` containing the coefficients of the polynomial $p(x) = a_0 + a_1 x + \cdots + a_n x^n$.

(a) Write a class method `poly_eval` which gets a value $x \in \mathbf{R}$ and return $p(x)$.

(b) Write a second method `poly_der_coef(k)` which returns the kth derivative of the polynomial. Test your code with the polynomial $p(x) = 1 - 2x + x^4 - 10x^5$.

Problem 3. Write a class `vector([a,b,c,d,...])` which stores a vector `[a,b,c,d,...]`. The class should have the functions `norm(p)` which returns the p-norm of the vector, a function `sum` which computes the sum off all elements of the vector and `ncount` which counts the number of elements of the vector. Also implement the function `__add__` and `__sub__` which adds and subtacts componentwise two vectors. This sould work as follows: if `a = vector([1,2,3,4])` and `b = vector([2,3,4,5])` this should implement `a+b` and `a-b`.

Problem 4. Define a class `vectorcomplex` which is derived from the vector class of the previous exercise. The new class should add the functionality of the vector class for complex vectors. The complex vector shall be stored in a list of two lists containing the real and complex part of the complement of the vector. Also extend the `__add__` and `__sub__` functions to complex numbers.

Problem 5. Write a python script which, for given dimension n, returns the matrix $A \in \mathbf{R}^{n\times n}$ with ones on the diagonal and anti-diagonal, while all other entries are zero. Example: for $n = 5$, this matrix reads as

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Avoid loops, and use only appropriate array functions and indexing instead.

Problem 6. Write a python-script which generates and displays for given dimension n an arrow-matrix $A \in \mathbf{R}^{n \times n}$ of the form

$$A = \begin{pmatrix} 1 & \cdots & 1 & 1 & 1 \\ & & & 1 & 1 \\ & & 1 & & 1 \\ & \iddots & & & \vdots \\ 1 & & & & 1 \end{pmatrix},$$

where all entries which are not shown have to be initialized with 0. Avoid loops! Instead, use matrix functions and matrix indexing!

Problem 7. Let $1 \leq p < \infty$. Write a Python-function `norm` which computes and returns the norm of a given matrix $A \in \mathbf{R}^{m \times n}$ with entries a_{ij} defined by

$$\|A\|_p := \Big(\sum_{j=1}^{m} \sum_{k=1}^{n} a_{jk}^p \Big)^{1/p}.$$

Avoid loops, and use only appropriate array functions and indexing instead.

Problem 8. Write a Python-script which generates, for given $n \in \mathbf{N}$, the following block diagonal matrix $A \in \mathbf{R}^{2n \times 2n}$.

$$A := \begin{pmatrix} 1 & 1 & & & & & \\ 1 & 1 & & & & & \\ & & 1 & 1 & & & \\ & & 1 & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & 1 \\ & & & & & 1 & 1 \end{pmatrix}$$

All entries which are not shown have to be initialized with 0. Avoid loops, and use only appropriate array functions and indexing instead.