Programmazione ad Alto livello con Python
Homework 7

Lezione 7: Introduzione alla libreria per il calcolo numerico NumPy.

Svolgere i seguenti esercizi. Le soluzioni possono essere discusse in orario di ricevimento, da concordare con il docente. Il testo degli esercizi viene fornito in lingua inglese per facilitare il riuso del materiale didattico.

1 Excercises

Exercise 1. Define the following vectors and matrices:

\[
\text{vec1} = \text{np.array([-1., 4., -9.])}
\]
\[
\text{mat1} = \text{np.array([[1., 3., 5.], [7., -9., 2.], [4., 6., 8.]])}
\]

Solve the following problems:

- You can multiply vectors by constants. Compute:
  \[
  \text{vec2} = (\text{np.pi}/4) \times \text{vec1}
  \]
- The cosine function can be applied to a vector to yield a vector of cosines. Compute:
  \[
  \text{vec2} = \text{np.cos( vec2 )}
  \]
- You can add vectors and multiply by scalars. Compute:
  \[
  \text{vec3} = \text{vec1} + 2 \times \text{vec2}
  \]
- The Euclidean norm of a matrix or a vector is available using \text{la.norm}. Compute:
  \[
  \text{la.norm(vec3)}
  \]
• You can do row-column matrix multiplication. Compute the product of `mat1` and `vec3` and set `vec4` equal to the result. Compute the transpose of `mat1`. Compute the determinant of `mat1`. Compute the trace of `mat1`. Find the smallest element in `vec1`. What function would you use to find the value of `j` so that `vec1[j]` is equal to the smallest element in `vec1`? What expression would you use to find the smallest element of the matrix `mat1`?

**Exercise 2.** As you know, a magic square is a matrix all of whose row sums, column sums and the sums of the two diagonals are the same. (One diagonal of a matrix goes from the top left to the bottom right, the other diagonal goes from top right to bottom left.) Show by direct computation that if the matrix `A` is given by:

```python
A=np.array([[17, 24,  1, 8, 15],
            [23,  5,  7, 14, 16],
            [ 4,  6, 13, 20, 22],
            [10, 12, 19, 21,  3],
            [11, 18, 25,  2,  9]])
```

The matrix `A` has 5 row sums (one for each row), 5 column sums (one for each column) and two diagonal sums. These 12 sums should all be exactly the same, and you could verify that they are the same by printing them and seeing that they are the same. It is easy to miss small differences among so many numbers, though. Instead, verify that `A` is a magic square by constructing the 5 column sums and computing the maximum and minimum values of the column sums. Do the same for the 5 row sums, and compute the two diagonal sums. Check that these six values are the same. If the maximum and minimum values are the same, the yswatteer principle says that all values are the same.

**Exercise 3.** Investigate the behavior of the statements below by looking at the values of the arrays `a` and `b` after assignments:

```python
a =  np.arange(5)
b = a
b[2] = -1
b = a[:]
b[1] = -1
b = a.copy()
b[0] = -1
```
Exercise 4. Create a 4x4 array with values generated by a random exponential variable with mean 0.5. Extract every element from the second row. Extract every element from the third column. Assign the value 0.21 to upper left 2x2 subarray.

Exercise 5. Construct an array containing only prime numbers greater than 100 and less than 200.

Exercise 6. Construct a 5x5 array in which each cell contains the distance of that cell from the center of the array. The concept of broadcasting will be very useful here. A 3x3 version would look like this:

```
array([[ 1.41421356,  1. ,       ,  1.41421356],
       [ 1. ,       ,  0. ,       ,  1. ]],
       [ 1.41421356,  1. ,       ,  1.41421356]])
```