# Lab. 2 Functions; IF and FOR instructions

For the exercises below, use the Spyder IDE. Make sure you use a working directory in your computer (preferably that you created in the previous lab) and select it in the File Explorer window of Spyder. In the Editor window create a file “lab2.py” and define the functions with the signatures below. Test the functions created from the console, after importing the file with command “import lab2”.

1. **Classification of a triangle**

   Specify function **triangle_type/3** that takes as arguments three non-negative numbers, interpreted as the sizes of the three sides of a triangle, and returns the type of such rectangle encoded as

   0 – not a triangle
   1 – scalene triangle
   2 – isosceles triangle
   3 – equilateral triangle

   **Examples:**
   ```
   triangle_type(6,6,6) -> 3
   triangle_type(9,2,4) -> 0
   triangle_type(9,5,5) -> 2
   triangle_type(3,4,5) -> 1
   ```

2. **Redo functions for arbitrary vectors**

   Specify **length(u)**, **vec_sum(u,v), def dot_product(u,v**) and **angle(u,v)** addressed in the previous lab, but now assume that vectors **u** and **v** may have any arbitrary length (of course the length is the same for both vectors).

3. **Vector Mean**

   Specify function **vec_mean/1** that takes a vector of real numbers as an argument and returns the mean of its elements.

   **Example:** `vec_mean([3,5,6,4,7]) -> 5.0`

4. **Vector Standard Deviation**

   Specify function **vec_std/1** that takes a vector of real numbers as an argument and returns its standard deviation.

   **Example:** `vec_std([3,5,6,4,7]) -> 1.4142`

5. **Matrix Statistics**

   Specify function mat_stat /1 that takes a matrix of real numbers as an argument and returns a vector with the **mean** and **standard deviation** of the elements of the matrix.

   **Example:** `mat_stat([[3,5,6],[4,5,7]]) -> [5.0, 1.2910]`

6. **Averaging rows and columns**

   a) Specify function **row_mean/1** that takes as input matrix of numbers and returns as a result a matrix with the same number of rows, each with one element representing the average of the elements of the matrix in that row

   **Example:** `col_mean([[1,7,2,4],[5,9,0,8]]) -> [[3.5],[5.5]]`

   b) Specify function **col_mean/1** that takes as input a matrix of numbers (encoded with lists, with any number of elements) and returns a vector with the mean value of each of the columns

   c) **Example:** `col_mean([[1,7,2,4],[5,9,0,8]]) -> [3 8 1 6]`

7. **Matrix Multiplication**

   Specify function **mat_mult/2** that takes as input two matrices with real numbers and returns their product. Note: if the matrices are not *compatible* return an empty array.

   **Example:** Given `A =[[4,3],[1,2],[7,8]], B = [[0,3,4],[2,1,4]`

   `mat_mult(A,B) -> [[6,15,28],[4,5,12],[16 29 60]]`

8. **Boolean Matrix Multiplication**

   Specify function **bool_mat_mult/2** that takes as input two Boolean matrices and returns their Boolean product (i.e. similar to the numeric case, but replacing multiplication by conjunction and sum by disjunction. Note: if the matrices are not *compatible* return an empty array.

   **Example:** Given `A =[1,0],[0,1],[1,1], B = [[0,1,0],[1,1,0]]`

   `bool_mat_mult(A,B) -> [[0,1,0],[1,1,0],[1,1,0]]`