Python exercises for Chapter 3. Session 1

Instructions for uploading the exercises

1. File names:
   - **Names of python scripts** are given according to the numbering of the list of exercises. Like `exercise_1.py`, `exercise_2.py`, etc.
   - **Names of output files** where the outputs are written to follow a similar naming format:
     - `exercise_1.txt`, if using the functions `print`, and `open` and `close`,
     - `exercise_1.npz`, if using the function `numpy.savez`, etc.
   - The **name of the zip file** must be `Surname1Surname2Name`, without white spaces, and excluding non-ASCII characters, such as tildes and *ñ*. For instance,

     Lucía Martín Cañas must write `MartinCanasLucia.zip`

     Include only the `exercise_*.py` files in your zip.
2. Ckeck that:
   - **Each script runs without errors**. To do this, in Spyder, or in any other IDE, restart the kernel (to clean variables) and run the script in the command window.
   - **The solution, and only the solution, is printed to the required output file.** Do not print intermmediate results in the final version of the script.

Exercises

1. Write a script for computing the Lagrange interpolating polynomial through the Lagrange Fundamental polynomials. To do this, create two functions: `lagrange_fundamental` with
   - Input: index i of the Fundamental polynomial, a point for evaluation, x, and the nodes of the interpolation problem.
   - Output: the value of the i-th Fundamental polynomial corresponding to the nodes at point x,

   and `lagrange_polynomial` with
   - Input: a point for evaluation, x, and the nodes and values of the interpolation problem.
   - Output: the value of the Lagrange interpolating polynomial.

   Use them for the following data:

   Nodes: $2,3,4,5,6,$ Values: 2, 6, 5, 5, 6

   and evaluate the Lagrange interpolating polynomial in a mesh of the interval $(2,6)$ with 100 equidistant points. If your evaluation is `v = lagrange_polynomial(mesh, nodes, values)` save the array `v` through the Numpy function `numpy.savez('exercise_1', v)`, which will generate a file `exercise_1.npz` in your folder (to be uploaded).

   To check your result, make two plots containing: (i) the Lagrange fundamental polynomial corresponding to the third node ($i = 2$), and (ii) the interpolating polynomial and the values at the nodes. Compare them to Figure 1.

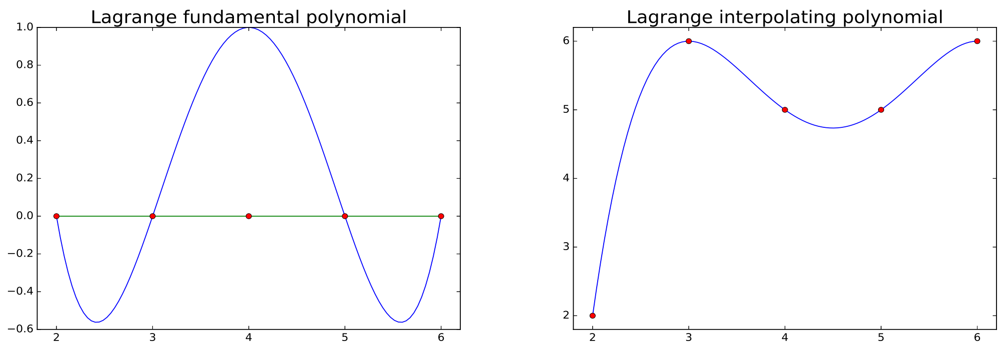

Figure : Exercise 1. Left: An example of Lagrange fundamental polynomial, corresponding to the third node. Right: The Lagrange interpolating polynomial.

Hint: Implement formulas (3.3) and (3.4) of the Handbook.

2. Write a script for computing the Lagrange interpolating polynomial through the divided differences. To do this, create two functions: `divided_differences` with

   - Input: the nodes and values of the interpolation problem.
   - Output: the matrix (or table) containing the divided differences of all orders,

   and `newton_polynomial` with

   - Input: a point for evaluation, x, and the nodes and values of the interpolation problem.
   - Output: the value of the Lagrange interpolating polynomial.

   Use them for the following data:

   $$\text{Nodes: } 2,3,4,5,6, \quad \text{Values: } 2,\ 6,\ 8,\ 7,\ 6$$

   and evaluate the Lagrange interpolating polynomial in a mesh of the interval $(2,6)$ with 100 equidistant points. If your evaluation is `v = newton_polynomial(mesh, nodes, values)` save `v` through `numpy.savez('exercise_2', v)`.

   To check your result:

   (a) Compare your table of divided differences to the following

   $$\begin{pmatrix} 2. & 4. & -1. & -0.16666667 & 0.16666667 \\ 6. & 2. & -1.5 & 0.5 & 0. \\ 8. & -1. & 0. & 0. & 0. \\ 7. & -1. & 0. & 0. & 0. \\ 6. & 0. & 0. & 0. & 0. \end{pmatrix}$$

   (b) Make a plot containing the interpolating polynomial and the values at the nodes and compare it to Figure 2.

   Hint: Implement formulas (3.7)-(3.9) of the Handbook.

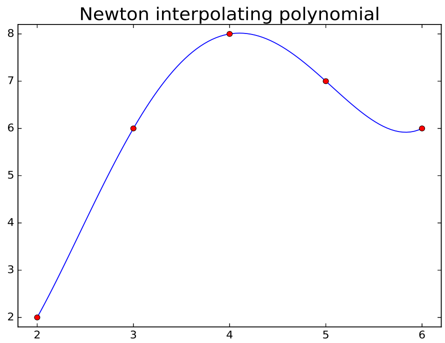

Figure : Exercise 2