Laboratory 08

Topics

1. Lists
2. Tables

Discussion

A. Discuss similarities and differences between lists and tables
B. Describe the use of the indexes in:
    a. strings;
    b. lists;
    c. tables.
C. How can you prevent an `out-of-range error`?

Esercizi

Part 1 – List and tables elaboration

***Task**: For each of the following exercises, write a program in Python that meets the given requirements. Complete <u>at least two</u> exercises during the laboratory session, and the remaining ones at home.*

08.1.1 Out-of-range. Write a program that contains an `out-of-range error`. What happens if you run the program? [R6.10]

08.1.2 Buffer. Write the pseudocode for an algorithm that, given a list of defined length, moves the elements "forward" one position (thus increasing their index by one unit), and moves the element at the last position to the first position. For example, the list `1 7 9 3 0 4`, after this operation, becomes the list: `4 1 7 9 3 0`. Write a program implementing the above described algorithm. [R6.17]

08.1.3 Play dice. Write a program that generates a sequence of `20` random rolls of a dice, stores them into a list and displays the generated values marking the longest set of identical values with the following format:

```
1 2 5 5 3 1 2 4 3 (2 2 2 2) 3 6 5 5 6 3 1
```

If there are multiple sequences of identical values of maximum length, use the formatting shown to highlight the first in order of occurrence. [P6.16]

08.1.4 Table. Write the instructions for the execution of the following operation for a table in Python of m rows and n columns (dimensions are inserted by the user):

I. Initialize the table with values equal to zero (`0`);
II. Fill all the cells with values equal to one (`1`);
III. Fill the cells alternating `0` and `1` in a chess scheme;
IV. Fill with `0` only the cells of the upper row and of the lower one, leaving the rest of the table unchanged;

V. Fill with `1` only the cells of the right column and of the left column, leaving the rest of the table unchanged;
VI. Calculate and display the sum of all the elements.

After each operation, display the table. [R6.28]

08.1.5 Lists union. Write a function `merge(a, b)` that merges the two lists `a` and `b`, alternating one element of the first and one element of the second. If one list is shorter than the other, the items are alternated as long as possible, then the items left in the longer list are added, in order, to the bottom. Starting lists should not be changed. If, for example, the content of `a` is `1 4 9 16` and the content of `b` is `9 7 4 9 11`, the invocation of `merge(a, b)` return a new list composed of the following values: `1 9 4 7 9 4 16 9 11`. [P6.30]

08.1.6 Sorted lists union. Write a function `merge_sorted(a, b)` that merges two lists (which are assumed to be already sorted ascending), returning a new, ascending sorted list. Manage a current index for each list to keep track of portions already processed. Starting lists should not be modified. If, for example, the content of `a` is `1 4 9 16` and the content of `b` is `4 7 9 9 11`, the invocation of `merge_sorted(a, b)` return a new list composed of the following values: `1 4 4 7 9 9 9 11 16`. The method `sort()` and the function `sorted()` must not be used. [P6.31]

Parte 2 – Algorithms using lists and tables

***Task**: For each of the following exercises, write a program in Python that meets the given requirements. Complete <u>at least one</u> exercise during the laboratory session, and the remaining ones at home.*

08.2.1 On average. Write the function `neighbor_average(values, row, column)` that, in a table `values`, calculates the average value of an element's neighbors in the eight directions, indexed as shown in the figure below (excluding the element itself). If, however, the element is on an edge of the table, the average should be calculated by considering only those neighbors that actually belong to the table. For example, if `row` and `column` are both `0`, the element has `3` neighbors. [P6.23]

`[i -1] [j - 1]`	`[i - 1] [j]`	`[i - 1] [j + 1]`
`[i] [j - 1]`	`[i] [j]`	`[i] [j + 1]`
`[i + 1] [j - 1]`	`[i + 1] [j]`	`[i + 1] [j + 1]`

08.2.2 Magic squares. A $n \times n$ matrix containing the integer numbers $1, 2, 3, \ldots, n^2$ is a "magic square" if the sum of its elements in each row, in each column, and in the two diagonals is the same. For example, this is a magic square of size 4:

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Write a program that acquires by the user 16 values, and places them in a table of 4×4 in order, one row at a time from top to bottom, and in each row from left to right, and check whether, after placing them, they form a magic square. Verify two properties:

I. All and only the numbers 1, 2, ..., 16 are present in the acquired data.
II. When the numbers are placed in the table, the sums of the rows, columns and diagonals are all equal to each other. [P6.21]

08.2.3 Tic-tac-toe. Write a program that plays the tic-tac-toe game. The game of tic-tac-toe is played on a 3×3 grid. The game is played by two human players taking turns. The first player marks the moves with a circle ('o'), the second one with a cross ('x'). The player who has formed a horizontal, vertical or diagonal sequence of 3 equal symbols. The program must, at each turn, display the game board, ask the user for the coordinates of the next move symbol (row and column, numbered from 1 to 3) as input, reverse the players after each move, and, when the game is over, decree the winner or a tie condition. [P6.28]

X		0
	X	0
0	0	X

08.2.4 Spring. Write a program that models and simulates the motion of an object of mass m attached to an oscillating spring. When the spring is displaced from its equilibrium position by a quantity x , Hooke's law states that the restoring force is given by the formula:

$$F = -kx$$

where k is a constant depending on the spring. For this simulation, use `k = 10 N/m`. Start with a given displacement x (i.e., `x = 0.5 m`). Set the initial speed to `v = 0`. Calculate the acceleration a based on Newton's law ($F = ma$) and Hooke's law, using a mass `m = 1 kg`. Use a small time interval `delta_t = 0.01 s` and, at each step, update the speed, by calculating a change of $a\Delta t$, and the displacement, by calculating a change of $v\Delta t$. [P6.38]