

[previous](#) [up](#) [next](#)

Tartalomjegyzék

- [1 Exercises](#)
 - ◆ [1.1 Square a List](#)
 - ◆ [1.2 Mean nearest](#)
 - ◆ [1.3 Increasing sublists](#)
 - ◆ [1.4 Name conflict](#)
 - ◆ [1.5 Pronunciation](#)
 - ◆ [1.6 Food courses](#)
 - ◆ [1.7 Pascal](#)
 - ◆ [1.8 Replacement](#)
 - ◆ [1.9 Name generator](#)

Exercises

Square a List

Write a function that only accepts a list as its parameter and the output is a new list with square components from the original list. Otherwise, the output: "http://wiki.math.bme.huThe parameter must be a list!"http://wiki.math.bme.hu

Mean nearest

Write a function that finds an element in a list which is the nearest to the mean of the list. There should be one parameter, the list.

Increasing sublists

Write a function that finds increasing sublists with a given length within a given list. The function should have two parameters:

- a list l
- and a natural number n
- return the list of n -long increasing sublists of l

Break down to subtasks:

- first return the list of all n -long sublists of l
- Check whether a sublist is increasing

Name conflict

We are throwing a party and there are a lots of unknown people there. We write their names in a list. Write a python function that decides whether there is a duplicate in the names (two person with the same name).

The function should have one parameter: the list of names.

Return `True` if there are at least two people with the same name, `False` if all the names are unique.

Hint:

Mind that do not compare ones name to itself, only to other's names.

Pronunciation

In Hungarian there are a lots of vowels and some words are hard to pronounce if there are a lots of consonants in them. For example "<http://wiki.math.bme.hu/lorozza>" <http://wiki.math.bme.hu> has a good number of vowels, but the Slovakian "<http://wiki.math.bme.hu/zmrzlina>" <http://wiki.math.bme.hu> has too many consonants.

Write a python function that decides whether a word has too many consonants or not.

- Call the function `pronunciation`
- with one parameter: `word`, the word in question
- return the string "<http://wiki.math.bme.hu>Hard<http://wiki.math.bme.hu> if the number of consonants are more (or equal) than twice the number of vowels.
- return "<http://wiki.math.bme.hu>Easy<http://wiki.math.bme.hu> otherwise.

Food courses

Suppose you have a really strict diet and you have a given number of calories what you should eat for dinner. You walk in a restaurant and they have the menu of soups and main dishes. Every item on the menu has a calorie value.

Can you choose the right dishes to exactly match your calorie intake?

Write a python function

- called `dishes`
- that have three parameters:
 - ◆ `calorie`, the number of calories you should eat
 - ◆ `soups`, a list containing calorie values for the soups
 - ◆ `mains`, a list containing calorie values for the main dishes
- You should return `True` if there is a two-course meal that fits your diet (`False` otherwise).

Pascal

The Pascal triangle consist of binomial coefficients, find details on [Wikipedia](#).

Write a function that calculates some lines of the triangle and returns it as a list of lists. First list is `[1]`, second is of length 2, and so on.

The function should have one parameter: n , the number of rows to calculate.

For example the result of `pascal(4)` should be:

```
[[1],  
 [1, 1],  
 [1, 2, 1],  
 [1, 3, 3, 1]]
```

Use the fact that a coefficient is the sum of the two elements above it.

Replacement

Write a function with two parameters: *word* is a string, and *replaces* a list of pairs where every pair is a number-character pair, like: (n, c) . You should replace the characters in *word* according to the *replaces*. On pair represents that you should replace the n^{th} character to the new letter *c*.

Return the new word after you performed the replacements.

For example replace $[(0, 'm'), (2, 'm'), (3, 'm')]$ in
"http://wiki.math.bme.hupuppy"http://wiki.math.bme.hu you get mummy.

Name generator

You write a computer game where you have to choose name of your player. The name consists of a first name and a last name where the names come from a given list of possibilities.

Generate all the possible names composed from the list of first names and list of last names.

[previous](#) [up](#) [next](#)