**The Longest Substring in Order**

**Learning goals**

Learning to design a somewhat more challenging algorithm for processing strings.

Implement the function longest\_substring\_in\_order, which takes a string as its parameter and searches for the longest substring with its characters in alphabetic order and then returns it. You can assume the string contains only lower-case letters, so you can compare the alphabetical order of the strings by using the comparison operator <, for instance.

If the string contains several equally long substrings in alphabetic order, the program returns the substring that is the closest to the beginning of the string.

The function returns its parameter value, if it is an empty string "" or a string made of a single character.

The longest substring in alphabetical order is character *x*, for strings that consist only of occurrences of *x* and *x* has at least two occurrences. For example, if the parameter value is "aaa", the return value is "a".

Example of how the function operates, when tested in the interactive mode of the Python interpreter (PyCharm's Python console):

>>> longest\_substring\_in\_order("abcabcdefgabab")

'abcdefg'

>>> longest\_substring\_in\_order("acdkbarstyefgioprtyrtyx")

'efgioprty'

Programming tips:

* You should go through the string using a for .. in range.. loop.
* For every index, calculate the length of the alphabetically ordered substring found starting from there.
* Strings can be compared using the operators < and > . For example the comparison "aaa" < "aab" returns True, because the string "aaa" is before the string "aab" if the strings are ordered alphabetically.
* If the found substring is longer than the longest one found earlier, take note of the information it contains. This way, after reaching the end of the string, you have stored information on the longest found substring.