Main tasks

|  |
| --- |
| **Task №1**  In this task you will create a function that takes a list of non-negative integers and strings and returns a new list with the strings filtered out.  **Examples:** |
| **Task №2**  Write a function named first\_non\_repeating\_letter that takes a string input, and returns the first character that is not repeated anywhere in the string.  For example, if given the input 'stress', the function should return 't', since the letter t only occurs once in the string, and occurs first in the string.  As an added challenge, upper- and lowercase letters are considered the **same character**, but the function should return the correct case for the initial letter. For example, the input 'sTreSS' should return 'T'.  If a string contains all repeating characters, it should return an empty string ("") or None -- see sample tests. |
| **Task №3**  [Digital root](https://en.wikipedia.org/wiki/Digital_root) is the recursive sum of all the digits in a number.  Given n, take the sum of the digits of n. If that value has more than one digit, continue reducing in this way until a single-digit number is produced. The input will be a non-negative integer.  **Examples:** |
| **Task №4**  There is an array of numbers - arr [1, 3, 6, 2, 2, 0, 4, 5]  there is a number target = 5.  Count the number of pairs in the array, the sum of which will give target |
| **Task №5**  Den has invited some friends. His list is:    Could you make a program that   * makes this string uppercase * gives it sorted in alphabetical order by last name.   When the last names are the same, sort them by first name. Last name and first name of a guest come in the result between parentheses separated by a comma.  So the result of function meeting(s) will be:  **Examples:**    It can happen that in two distinct families with the same family name two people have the same first name too. |

Extra tasks

|  |
| --- |
| **Task №1**  Create a function that takes a positive integer and returns the next bigger number that can be formed by rearranging its digits.  **Examples:**    If the digits can't be rearranged to form a bigger number, return -1 |
| **Task №2**  Take the following IPv4 address: 128.32.10.1  This address has 4 octets where each octet is a single byte (or 8 bits).   * 1st octet 128 has the binary representation: 10000000 * 2nd octet 32 has the binary representation: 00100000 * 3rd octet 10 has the binary representation: 00001010 * 4th octet 1 has the binary representation: 00000001   So 128.32.10.1 == 10000000.00100000.00001010.00000001  Because the above IP address has 32 bits, we can represent it as the unsigned 32 bit number: 2149583361  Complete the function that takes an unsigned 32 bit number and returns a string representation of its IPv4 address.  **Examples:** |