Class Lab – Python

הוראות:
1. לצורך ביצוע המשימה יש ברשותכם 4 שעות.
2. יש להעלות את המשימה לאתר הקורס –בלשונית : מטלות
3. חומר עזר: אינטרנט/ספרות כתובה
4. המשימה מהווה מבחן!

In this Lab you will create a package with 3 modules and test them.

- Create a new project for this task
- Create a new package **utils**

<u>numbers module</u>

- Add module **numbers** to **utils** package. **numbers** module will contain functions that operate with numbers.
- Implement the following functions in **numbers** module:
    - dividable_by_10 - the function gets a number and returns True if the number can be divided by 10 without remainder, False otherwise. You can assume that the provided parameter is an integer number.

    - sum_of_digits – the function gets a number as a parameter and returns sum of its digits (for example 234 => 9). You can assume that the provided parameter is an integer number.

- is_palindrome – the function gets a number and returns True if the number is palindrome (1221 , 34543, ….). You can assume that the provided parameter is an integer number.

col module

- Add module **col** to **utils** package. This module will contain functions that operate on various collections.
- Implement the following functions in **col** module:
  - unique_values - function gets list and returns list without duplicates.
    For example, [34, 5, 1, 34, 6, 1] => [34, 5, 1, 6]
  - lists2dict - function gets 2 lists: **keys** and **values**, and creates a dictionary from them, such that elements of the **keys** list become keys in the dictionary, and the elements of the **values** list become values of the dictionary. The function returns the newly created dictionary.
    Note, the length of lists will not necessarily be equal. In this case, create the dictionary with the length of the list with a minimum amount of items.
    In addition, you should raise an exception if there are duplicates in the **keys** list, since in this case you will not be able to create a dictionary (a dictionary can not contain duplicate keys).
    For example, for the following lists:
    keys=[“a”, “b”, “c”, “d”]
    values=[1, 23, “ttt”]
    lists2dict will return the following dictionary:
    {“a”:1, “b”:23, “c”:”ttt”}
  - elems_count - function gets list of strings and returns a dictionary that counts how many times each string appeared in the list.
    For example: [“apple”, “pear”, “banana”, “apple”] =>

{“apple”: 2,

“pear”: 1,

“banana: 1}

birth_dates module

- Add module birth_dates in package utils. You will implement here a couple of functions that work with people's birthdays. You should use functions you wrote earlier.
- Implement the following functions:
  - is_lucky() - the function gets a year of birth, and checks the following: if the year is palindrome, or sum of all digits of the year can be divided by 10 without remainder, then the person born at this year is considered to be lucky, and the function should return True. Otherwise, return False.
    For example:
    1987 => False
    2002 => True
    2008 => True
  - lucky_people() - the function gets two lists - names and birth_dates. Names list contains names of people (strings), and birth_dates list contains strings in format dd-mm-yyyy that represents birth date for the people in the names list.
    The function should return 2 values:
    1. a list of people's names that are considered to be lucky (according to is_lucky function).
    2. a dictionary that stores how many times each birth date appears in the original birth_dates list.

You can assume that the lists contain strings, and the format of years is correct. However, it is not guaranteed that the length of the lists is the same, and there could be duplicates in the names list. In case of duplicate names you should raise an error. In case of different list lengths, go for the shortest one (exactly like in lists2dict function).

For example:

[“John”, “Mary”, “David”], [“20-02-2001”, “03-04-2002”, “30-11-2012”, “30-11-2012”]

=>

[“Mary”], {“20-02-2001”: 1, “03-04-2002”: 1, “30-11-2012”: 2}

Tests

- Add **tests** package that contains 3 test modules: test_numbers, test_col, test_birth_dates
- In every test module, implement test functions that test all the functions you coded earlier in that module. Think about edge cases!
- Add a simple python script main.py that runs all your tests (from test modules) and prints which tests passed and which failed in a user-friendly way.

Important note:

Your code should be ready to run. I will run your main.py and it should work!