Laborator 1 - aplicatii

Context

Temele se rezolva individual si se predau in tema de elearning, pana in 18 octombrie ora 23. Se acorda 2 puncte din oficiu.

Numele fisierului va avea structura: laborator1_tema_nume_prenume.ipynb.

Tema 1

1. (1 p). Se da o lista `lista` de numere intregi. Sa se depuna numerele pare in lista `lista_pare` numerele pare si in `lista_impare` cele impare.

Exemplu:

In [ ]:
```
lista = [223, 1019, 4, 658, 8, 9, -4]

lista_pare = ...
lista_impare = ...

# in mod normal, lista numerelor pare/impare
assert lista_pare == [4, 658, 8, -4]
assert lista_impare == [223, 1019, 9]
```

Pentru exemplele proprii adaptati corespunzator cele doua asertiuni.

1. (1 p) Se da o lista de stringuri. Se cere ca in fata fiecarui string cu lungime para sa se insereze lungimea acelui string. Nu se va folosi lista suplimentara.

Exemplu:

In [ ]:
```
my_string = "Atunci când citești un text trebuie să fii capabil să identifici ideea pr
my_list = my_string.split(' ')

...
print(my_list)

assert my_list == [6, 'Atunci', 4, 'când', 'citești', 2, 'un', 4, 'text', 'trebuie', 2
```

1. (1 p) Se da o lista de cel putin 3 numere in virgula mobila. Care este al treilea cel mai mic numar?

Exemplu:

In [ ]:
```
lista_numere = [1, -1, 2, -2, 3, -3, 4, -4, 100, 0]

al_treilea = ...
print(al_treilea)

assert al_treilea == -2
```

1. (1 p) Se da o lista de stringuri. Sa se determine frecventa de aparitie a fiecarui string, intr-un dictionar: cheia dintr-un dictionar este cuvantul, iar valoarea este numarul de aparitii.

Optional: dictionarul va contine cuvinetele in ordine descrescatoare a numarului de aparitii.

Exemplu:

In [ ]:
```
my_string = "Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantiu
my_string = my_string.lower().replace(',', '').replace('.', '').replace('?', '')
my_list = my_string.split(' ')

dict_frecvente = {}
...

print(dict_frecvente)

assert dict_frecvente == {'sed': 3, 'ut': 4, 'perspiciatis': 1, 'unde': 1, 'omnis': 1,
```

1. (1 p) Fiind data o lista de cuvinte scrise cu litere mici, să se returneze lista care contine cuvintele care NU incep cu litera b, sortate in ordine alfabetica.

Exemplu:

In [ ]:
```
lista_cuvinte=["broasca", "maxim", "alfabet","carte", "inima", "barca", "trunchi", "ar

...

assert cuvinte_fara_b_alfabetic == ['alfabet', 'are', 'carte', 'inima', 'maxim', 'trur
```

1. (2p) Se da un dictionar in care cheile sunt nume de studenti, iar valorile sunt liste de carti preferate. Sa se determine:
   A. Care sunt studentii care au in lista de preferinte o carte specificata?
   B. Care sunt perechile de studenti care au aceleasi preferinte de carti (ordinea in lista de preferinte nu e relevanta)
   C. Pentru un student dat, care sunt studentii cu care are cele mai multe carti comune in liste de preferinte; daca sunt mai multi astfel de studenti cei mai apropiati, se vor enumera toti.
   D. Pentru doi studenti, A si B, care e lista de stergeri si adaugari prin care lista lui A devine identica cu lista lui B (ordinea in lista nefiind importanta)
   E. Toate perechile de studenti A, B pentru care lista de preferinte a lui A este inclusa in (dar nu coincide cu) lista de preferinte a lui B.

Indicatie: considerati tipul de date `set`.

Exemplu:

```
preferinte = {
    'Popescu': ['carte2', 'carte1', 'carte3', 'carte4'],
    'Ionescu': ['carte1', 'carte2', 'carte3', 'carte7', 'alta carte'],
    'Georgescu': ['carte1', 'alta carte'],
    'Xulescu': ['carte2', 'carte5', 'carte6', 'carte7'],
    'Dragomir': ['carte4', 'carte1', 'carte2', 'carte3']
}
```

```
1. carte = 'alta carte' -> ['Ionescu', 'Georgescu']
2. Popescu si Dragomir au aceleasi preferinte: ['carte2', 'carte1',
'carte3', 'carte4']. Se remarca faptul ca nu se afiseaza si perechea
echivalenta: Dragomir si Popescu.
3. Xulescu -> Suprapunere maxima de preferinte: 2 cu colegii:
['Ionescu']
4. A='Popescu', B='Ionescu' -> Se vor adauga cartile: {'carte7',
'alta carte'}, se vor scoate cartile: {'carte4'}
5. Preferintele lui Georgescu sunt subset propriu al preferintelor
lui Ionescu
```

7 (1p). Se da o lista de stringuri si o lista de intregi; nicio lista nu contine valori duplicate. Sa se determine toate combinatiile de elemente din prima lista cu elemente din a doua lista, cu valorile despartite prin '_'.

Exemplu:

In [ ]:
```
lista_1 = ['latop', 'carte', 'telefon']
lista_2 = list(range(1, 5))

...

assert lista_combinatii == ['latop_1', 'latop_2', 'latop_3', 'latop_4',
                            'carte_1', 'carte_2', 'carte_3', 'carte_4',
                            'telefon_1', 'telefon_2', 'telefon_3',
                            'telefon_4']
```

In [ ]:
```

```