

### Răspunsuri exerciții seminar 3:

#### Exercițiu A.

```
a = [1, 2, 4, 10]
b = a
a[2] = 'hello!'
print(b)
```

A.1. \_\_\_\_\_

Se afișează [1, 2, 'hello!', 10], prin atribuirea `b = a`, `b` ajunge să refere aceeași listă ca și `a`, astfel orice modificare făcută prin `a` asupra listei se va putea vedea și accesând `b`.

```
a = [1, 2, 4, 10]
b = a
b[0] = 'hello!'
print(a)
```

A.2. \_\_\_\_\_

Analogue cu A.1. `b` și `a` referă aceeași listă din memorie.

#### Python Tutor: Visualize code in [Python](#), [JavaScript](#), [C](#), [C++](#), and [Java](#)

The image displays two screenshots of the Python Tutor visualization tool, illustrating the execution of Python code and the state of memory frames and objects.

**Top Screenshot:**

- Code:** Python 3.6, known limitations. The code is:

```
1 a = [1, 2, 4, 10]
2 b = a
3 a[2] = 'hello!'
4 print(b)
```

The line number 4 is highlighted, indicating it is the next line to execute.
- Print output:** [1, 2, 'hello!', 10]
- Frames:** Global frame. Variables `a` and `b` are shown, both pointing to the same list object.
- Objects:** A list object with elements [1, 2, 'hello!', 10]. The index 2 is highlighted, corresponding to the value 'hello!'.

**Bottom Screenshot:**

- Code:** Python 3.6, known limitations. The code is:

```
1 a = [1, 2, 4, 10]
2 b = a
3 #a[2] = 'hello!'
4 b[0] = "hello!"
5 print(b)
```

The line number 5 is highlighted, indicating it is the next line to execute.
- Print output:** ['hello!', 2, 4, 10]
- Frames:** Global frame. Variables `a` and `b` are shown, both pointing to the same list object.
- Objects:** A list object with elements ['hello!', 2, 4, 10]. The index 0 is highlighted, corresponding to the value 'hello!'.

#### A.3. List slicing: <https://www.pythonmorsels.com/slicing/>

Se creează și afișează o nouă listă care conține elementele de la indexul 2 până la indexul 5 din lista originală: [1, 2, 4, 10]. Indexul de start se ia în considerare, dar nu și cel de final.

Python 3.11  
known limitations

```

1 lst = [-3, 6, 1, 2, 4, 10, 18, 23]
2 slice = lst[2:6]
3
4 print(id(lst))
5 print(id(slice))

```

[Edit this code](#)

⇒ line that just executed  
→ next line to execute

Done running (4 steps)

Print output (drag lower right corner to resize)

```

140468461397568
140468461391744

```

Frames

Global frame

lst

slice

Objects

list

0	1	2	3	4	5	6	7
-3	6	1	2	4	10	18	23

list

0	1	2	3
1	2	4	10

A. 4. Se afișează ['a', 'a', 'a', 'a', 'a']. Se creează o listă în care obiectul de tip str 'a' este multiplicat de 5 ori.

! Atenție: ce se întâmplă dacă obiectele din interiorul listei inițiale sunt **mutable**?

Python 3.11  
known limitations

```

1 lst = ['cuvant1700'] * 5
2 print(lst)
3 for i in range(len(lst)):
4     print(f'id(lst[{i}])={id(lst[i])}')
5
6
7 print("Dupa lst[0] = 'cuvant9':")
8 lst[0] = 'cuvant9'
9 print(f'id(lst[{0}])={id(lst[0])}')
10 print(f'id(lst[{1}])={id(lst[1])}')
11 print(lst)
12
13
14 lst_nr = [[1,2,3]]*3
15 print(f'Lista initiala de numere este {lst_nr}.')
16
17 for i in range(len(lst_nr)):
18     print(f'id(lst_nr[{i}])={id(lst_nr[i])}')
19 lst_nr[0][0] = 18
20 print(f'Lista de numere este acum {lst_nr}.')

```

[Edit this code](#)

⇒ line that just executed  
→ next line to execute

Print output (drag lower right corner to resize)

```

['cuvant1700', 'cuvant1700', 'cuvant1700', 'cuvant1700', 'cuvant1700']
id(lst[0])=139659397983984
id(lst[1])=139659397983984
id(lst[2])=139659397983984
id(lst[3])=139659397983984
id(lst[4])=139659397983984
Dupa lst[0] = 'cuvant9':
id(lst[0])=139659397982960
id(lst[1])=139659397983984
['cuvant9', 'cuvant1700', 'cuvant1700', 'cuvant1700', 'cuvant1700']
Lista initiala de numere este [[1, 2, 3], [1, 2, 3], [1, 2, 3]].
id(lst_nr[0])=139659534494720
id(lst_nr[1])=139659534494720
id(lst_nr[2])=139659534494720
Lista de numere este acum [[18, 2, 3], [18, 2, 3], [18, 2, 3]].

```

Frames

Global frame

lst

i

lst\_nr

Objects

list

0	1	2	3	4
"cuvant9"	"cuvant1700"	"cuvant1700"	"cuvant1700"	"cuvant1700"

list

0	1	2
18	2	3

list

0	1	2
18	2	3

A.5. Se afișează [5, 4, 3, 2, 1].

[Read more about range here](#)

A. 6. Se afișează ['Ana', 'are', 'mere'].

`list.extend(iterable)`

Extend the list by appending all the items from the iterable. Equivalent to `a[len(a):] = iterable`.

A. 7. Se afișează 3.

[Read more about tuple unpacking here.](#)

A. 8. Se afișează numărul de la ultimul index al listei: -5. Dacă am încerca să afișăm `lst[-2]`, s-ar afișa valoarea 20, întrucât acesta este penultimul element al listei.

[Read more about negative indexing here.](#)

A. 9. Se afișează True, se verifică dacă 'E' există în tuplu.

Extra: ce se afișează în urma executării următoarelor instrucțiuni?

```

participants = ('A', 'B', 'V', 'E')
print('E' in participants)

participants_l = ['A', 'B', 'V', 'E']
print('E' in participants_l)

participants_s1 = 'ABVE'
print('E' in participants_s1)

participants_s2 = 'ABVCDEFG'
print('E' in participants_s2)

participants_s3 = 'ABCDEF6'
print('CDE' in participants_s3)

participants_modified = ['A', 'B', 'V', 'Eva']
print('E' in participants_modified)

```

A.10. Se șterge ultimul element, 9.

`list.remove(x)`

Remove the first item from the list whose value is equal to x. It raises a [ValueError](#) if there is no such item.

Cum ar trebui să modificăm codul pentru a se șterge lista de la indexul 1 din listă?

B.

```

def count_character_frequencies(sentence: str) -> dict:
    """
    Functia calculeaza frecventa caracterelor dintr-o propozitie data
    :param sentence: propozitia data
    :return un dictionar care contine perechi de caracter-frecventa
    """
    frequency_dict = {}
    for character in sentence.lower():
        if character in frequency_dict:
            frequency_dict[character] += 1
        else:
            frequency_dict[character] = 1

    return frequency_dict

test_sentence1 = "Ana are mere."
freq_dict_test1 = count_character_frequencies(test_sentence1)
assert (freq_dict_test1['a'] == 3)
assert (freq_dict_test1[' '] == 2)

test_sentence2 = ""
freq_dict_test2 = count_character_frequencies(test_sentence2)
assert (len(freq_dict_test2) == 0)

```