

Curs 4: list, tuple, dict

1. Clasa list:

- Are la bază o construcție de tip array
- Are proprietăți asemănătoare cu a listelor dar și a vectorilor din C++
- Obiectele de tip list sunt MUTABLE (își pot schimba valoarea fără a fi nevoie să se genereze un nou obiect cu un alt id) și ITERABLE
- Obiectele de tip list sunt indexabile (pornind de la 0) folosind operatorul [].
- Spre deosebire de clasa string, putem atribui/schimba valorile unui element/slice din lista
- Fiind iterabila, putem folosi operatorul unar *, pentru “despachetarea” unei liste
- Atunci când facem un **slice** din lista, se generează un **nou obiect**, cu un nou id (shallow copy). Dacă sliceului vom atribui un nou obiect iterabil, modificarea se va vedea și în lista de proveniență a sliceului. Totuși, dacă în slice facem update la o singură valoare, sliceul fiind obiect separat, schimbarea nu se va “vedea” în lista originală. Vedeți [exemplul 4.1.1](#)

[Exemplul 4.1.1](#) Declararea unei liste, schimbarea unei valori din lista, verificarea dacă un element există în lista, slice-uri.

O listă are câteva metode din clasa list. A se observa că aceste metode pot modifica direct valoarea obiectului apelant, spre deosebire de cazul clasei str, unde obiectul apelant nu se modifică, ci metodele returnează un nou string cu valoarea dorită.

Nume metodă	Descriere	Complexitate	Exemplu
<code>list.append(x)</code>	Extinde (la dreapta) lista apelantă cu un element căruia i se atribuie valoarea lui x . Lungimea listei crește cu 1.	$O(1)$	4.1.2
<code>list.extend(iterable)</code>	concatenează la dreapta lui list toate elementele din iterable. Parametrul orimit nu trebuie să fie neapărat tot o listă, ci orice obiect iterabil (set, string, range, dicționar, tuple, etc) Echivalent ca rezultat cu: <code>list[len(list):]=iterable</code> sau <code>list+=iterable</code>	$O(\text{len}(\text{iterable}))$	4.1.3
<code>list.insert(i, x)</code>	inserează un element cu valoarea lui x pe poziția i . Dimensiunea listei crește cu 1.	$O(k)$ k - numărul de elemente ce sunt shiftate	4.1.4
<code>list.remove(x)</code>	Șterge prima apariție a lui x din list. Aruncă ValueError în cazul în care	$O(n)$	4.1.5

	elementul căutat nu există în list.		
<code>list.pop([i=-1])</code>	returnează și șterge elementul de pe poziția <i>i</i> din listă. Dacă nu este specificat indicele, atunci se elimină ultimul element din listă (indexul -1)	O(k) atunci când se specifică indicele O(1) când se elimină ultimul element	4.1.6
<code>list.clear()</code>	Șterge elementele din lista, dar nu dereferențiază lista (așa cum ar face-o <code>del(list)</code>). Echivalent cu <code>list=[]</code>	O(1)	-
<code>list.index(x[, start[, end]])</code>	Caută pe x în list și returnează indicele primei apariții. Opțional se pot adăuga 2 parametri de <i>start</i> și <i>end</i> pentru a căuta pe x în un anumit interval. În ambele cazuri indexarea se face de la primul element al lui list. În cazul în care x nu este în list, se aruncă un <code>ValueError</code>	O(n)	4.1.7
<code>list.sort(key=None, reverse=False)</code>	sortează lista crescător, cu condiția ca lista să conțină elemente comparabile. Elementele care sunt considerate egale își vor păstra ordinea originală între ele (stable sort)	O(n log n) timsort	4.1.8
<code>list.reverse()</code>	Inversează ordinea elementelor din listă	O(n)	-
<code>list.copy()</code>	Crează și întoarce un nou obiect egal în valoare cu obiectul apelant (dar id diferit)	O(n)	4.1.9

2. Clasa tuple:

- imutabil
- iterabil
- indexabil

Declararea unui tuplu, modificarea lui, iterarea printr-un tuplu, accesarea unui element.

[Exemplul 4.2.1](#)

Dece tuple si nu list?

- folosim tupluri pentru colecții de obiecte heterogene. Listele ar trebui să conțină obiecte de același tip (omogene), deși nu există o restricție în acest sens.
- Deoarece tuplurile sunt imutabile, iterația printr-o astfel de structură este mai rapidă
- Tuplurile care conțin la rândul lor doar elemente imutabile pot fi folosite pe post de chei în dicționare. Listele, fiind ele însele mutabile, nu pot fi folosite pe post de cheie (nu se poate face hashing pe ceva care își poate schimba valoarea)
- "Write protection"

Nume metodă	Descriere	Complexitate	Exemplu
<code>tuple.count(x)</code>	returnează numărul aparițiilor lui x în tuplet	O(n)	
<code>tuple.index(x)</code>	caută prima apariție a lui x în tuplet și returnează indexul ei. În cazul în care x nu se găsește în tuplet, se aruncă o <code>ValueError</code> . (Existența se poate verifica mai simplu folosind cuvântul cheie in)	O(n)	

3. Clasa dict():

- mutabil
- iterabil
- indexabil

Dicționarul este o enumerare de perechi de forma <key, value>, unde operațiile de inserție, interogare și update se fac în timp constant. În dicționar cheile trebuie să fie unice, iar elementele care sunt chei trebuie să fie de un tip **imutabil**. (ex: listele nu pot fi chei, dar tuplele pot)

Nume metodă	Descriere	Complexitate	Exemplu
<code>dict.clear()</code>	elimină toate valorile din dicționar	O(n)	
<code>dict.get(x, [val=None])</code> sau <code>dict[x]</code>	returnează valoarea de la cheia x . Dacă nu există cheia x , se returnează o valoare <i>val</i> setată implicit cu <code>None</code>	O(1)	4.3.1
<code>dict.update({key,value})</code>	dacă key nu se găsește printre cheile din dicționar, atunci se inserează în dicționar perechea (key,value). Dacă acea cheie se află deja în dicționar, se actualizează doar valoarea. Echivalent cu <code>dict[key]=value</code>	O(1)	4.3.2
<code>dict.keys()</code>	returnează o listă formată din mulțimea cheilor	O(n)	
<code>dict.values()</code>	returnează o listă formată din mulțimea valorilor	O(n)	

<code>dict.items()</code>	returnează o listă formată din perechile (cheie, valoare)	$O(n)$	4.3.3
---------------------------	---	--------	-----------------------