UNIVERSITATEA TEHNICĂ A MOLDOVEI

Generarea listelor prin sintaxa List comprehension

UNIVERSITATEA TEHNICĂ A MOLDOVEI

List comprehension

List comprehension reprezintă o sintaxă de creare a unor liste noi pornind de la un alt iterabil (listă, dicţionar, set etc).

```
lista = [1, 2, 3, 4, 5]

rezultat = []
for element in lista:
    rezultat.append(element)

print(rezultat)
```

```
lista = [1, 2, 3, 4, 5]
rezultat = [element for element in lista]
print(rezultat)
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

List comprehension

Putem efectua **verificări ale datelor înainte ca acestea să fie adăugate în lista cea nouă**.

```
rezultat = []
for element in lista:
    if element % 2 == 0:
        rezultat.append(element)
```

```
rezultat = [element for element in lista if element % 2 == 0]
```

Sintaxa este similară cu cea din slide-ul precedent, doar că s-a adăugat verificarea **`if`** la final

UNIVERSITATEA TEHNICĂ A MOLDOVEI

List comprehension

Dacă avem şi ramură **`else`**, atunci structura **`if`** nu se va mai găsi la final, **ci la început.**

```
rezultat = [element ** 2 if element % 2 == 0 else element * 2 for element in lista]
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

List comprehension

Dacă avem şi ramură **`else`**, atunci structura **`if`** nu se va mai găsi la final, **ci la început.**

```
rezultat = [element ** 2 if element % 2 == 0 else element * 2 for element in lista]
```

```
rezultat = []
for element in lista:
    if element % 2 == 0:
        rezultat.append(element ** 2)
    else:
        rezultat.append(element * 2)
```

UNIVERSITATEA TEHNICĂ A MOLDOVEI

List comprehension

```
l8= [c*3 for c in 'list' if c!='i']
```

```
l8
['lll', 'sss', 'ttt']
```

Sintaxa list comprehension are două mari avantaje:

1. Sintaxa **este mult mai uşor de citit** şi de înţeles deoarece poate fi citită precum o frază în limba engleză;
2. Această **sintaxă este mult mai rapidă** din punct de vedere al timpului de execuţie decât sintaxa obişnuită.

UNIVERSITATEA TEHNICĂ A MOLDOVEI

List comprehension

```
from timeit import timeit
print(timeit('for elem in range(100000):lista_noua.append(elem)', setup='lista_noua=[]', number=1000))

print(timeit('lista_noua=[elem for elem in range(100000)]', number=1000))
```

```
11.510976700003084
5.425535500002297
```

Sintaxa de list comprehension reuşeşte să efectueze aceste operaţii în jumătate din timpul necesar variantei clasice.
Bineînţeles, durata exactă a execuţiei liniilor de mai sus depinde şi de puterea de procesare a calculatoarelor pe care rulează, însă proporţiile se vor păstra.

List comprehension

Sintaxa list comprehension este destul de atrăgătoare. În majoritatea cazurilor listele cu care veţi lucra **vor avea dimensiuni relativ mici**, astfel că ***avantajul de timp nu va mai fi atât de important***.

Marele avantaj rămas va fi uşurinţa de a scrie şi înţelege o astfel de sintaxă. Din acest motiv, este recomandat să nu avem structuri de list comprehension prea complexe. **O regulă generală ar fi să avem doar două instrucţiuni în structura acesteia (un for şi un if, două for-uri).** *Dacă sintaxa list comprehension se întinde pe mai mult de două linii, avantajul de a fi uşor de citit s-ar putea să dispară*, să fie chiar mai greu de citit decât varianta clasică.

List comprehension nu este singurul tip de comprehension.
Cu aceleaşi reguli şi avantaje sunt sintaxele ***dict comprehension*** (pentru crearea de dicţionare), ***set comprehension*** (pentru crearea de seturi) şi ***generator comprehension*** (pentru crearea de generatoare).

UNIVERSITATEA TEHNICĂ A MOLDOVEI

List comprehension