INFORMATICĂ · PYTHON · CLASA A VIII-A · TABLOURI UNIDIMENSIONALE · LECȚIA 7

Modificarea unui șir: inserare și ștergere

Durată: 50 min Nivel: gimnaziu · Python Înainte: lecțiile py8g-06

CE VEI ȘTI DUPĂ LECȚIE

- Modifici elementele unui șir pe parcursul unei parcurgeri după indice.
- Ștergi un element de pe o poziție dată, mutând celelalte elemente.
- Inserezi o valoare pe o poziție dată.

1. Schimbăm elemente după indice

Vrem ca toate numerele negative din șir să devină 0.

```
v = [3, -2, 8, -5, 1]
for i in range(len(v)):
    if v[i] < 0:
        v[i] = 0
print(*v)
```

```
3 0 8 0 1
```

ATENȚIE — GREȘELI FRECVENTE

Cu `for x in v:` **nu** poți schimba lista. Variabila `x` este doar o **copie** a valorii:

```
v = [1, 2, 3]
for x in v:
    x = x * 2
print(v)
```

Afișează tot `[1, 2, 3]`. Ca să modifici, parcurgi după indice și scrii `v[i] = ...`.

2. Ștergerea unui element

Vrem să ștergem elementul de pe poziția `p`. Gândește-te la o coadă la cinema: când pleacă cineva din mijloc, cei din spatele lui fac un pas înainte. În listă: **mutăm toate elementele de după `p` cu o poziție la stânga**, apoi scoatem ultima căsuță, care a rămas dublură.

```
v = [10, 20, 30, 40, 50]
n = len(v)
p = 1
for i in range(p, n - 1):
    v[i] = v[i + 1]
v.pop()
print(*v)
```

```
10 30 40 50
```

Pas cu pas (`p = 1`):

Moment	Lista
început	`10 20 30 40 50`
`v[1] = v[2]`	`10 30 30 40 50`
`v[2] = v[3]`	`10 30 40 40 50`
`v[3] = v[4]`	`10 30 40 50 50`
`v.pop()`	`10 30 40 50`

`v.pop()` scoate **ultimul** element al listei.

3. Inserarea unui element

Vrem să punem valoarea `x` pe poziția `p` . Întâi facem loc: mutăm elementele de la `p` încolo cu o poziție la **dreapta**, începând de la sfârșit (altfel le-am șterge unul pe altul).

```python
v = [10, 20, 30, 40]
n = len(v)
p = 1
x = 99
v.append(0)
for i in range(n, p, -1):
    v[i] = v[i - 1]
v[p] = x
print(*v)
```

```
10 99 20 30 40
```

Cu `v.append(0)` am creat o căsuță nouă la capăt. Apoi mutăm: `v[4] = v[3]` , `v[3] = v[2]` , `v[2] = v[1]` . Abia la final punem `99` în locul rămas liber, `v[1]` .

> **SFAT**
>
> **Scurtături (după ce ai înțeles ideea).** Python le face pe toate dintr-un rând: `del v[p]` șterge elementul de pe poziția `p` , iar `v.insert(p, x)` inserează `x` pe poziția `p` .

4. Păstrăm doar ce ne trebuie

Vrem să scoatem din listă toate valorile egale cu 0. Dacă ștergi în timp ce parcurgi, elementele „sar” și unele rămân. Mai simplu: construiești **o listă nouă** cu ce vrei să păstrezi.

```python
v = [4, 0, 7, 0, 0, 3]
w = []
for x in v:
    if x != 0:
        w.append(x)
print(*w)
```

```
4 7 3
```

5. Verifică-te

1. Ce valoare are `v` după `v[2] = 5`, dacă `v = [1, 1, 1, 1]`?
2. Ce afișează `v = [1, 2, 3]`, `v.pop()`, `print(*v)`?
3. De ce mutăm elementele de la sfârșit spre `p` când inserăm?
4. Ce se întâmplă dacă ștergi elementul de pe poziția `p = 0` din `[7, 8, 9]` cu metoda din secțiunea 2? Scrie rezultatul.
5. De ce programul din avertisment nu schimbă lista?

Răspunsuri

1. `[1, 1, 5, 1]`.
2. `1 2`.
3. Dacă am începe de la `p`, valoarea de pe `p` ar acoperi-o pe cea de pe `p + 1`, și toate ar deveni egale. De la sfârșit, fiecare valoare ajunge în căsuța deja „eliberată”.
4. `8 9`.
5. `x` este o copie: schimbarea ei nu atinge elementul din listă.