INFORMATICĂ · PYTHON · CLASA A VIII-A · TABLOURI UNIDIMENSIONALE · LECȚIA 6

Căutarea unui element și verificări

Durată: 50 min Nivel: gimnaziu · Python Înainte: lecțiile py8g-05

CE VEI ȘTI DUPĂ LECȚIE

- Cauți un element într-un șir (căutare liniară) și afli poziția lui.
- Verifici dacă toate elementele respectă o regulă.
- Verifici dacă un șir este ordonat crescător.

1. Căutare liniară: te uiți pe rând

Cauți o carte pe raft: iei cărțile una câte una până o găsești. Asta este **căutarea liniară**. În Python folosim un „steag” (o variabilă `gasit`) care are valoarea `False` la început (nu am găsit încă) și devine `True` când găsim elementul.

```
v = [12, 7, 9, 20, 15]
x = 9
gasit = False
for e in v:
    if e == x:
        gasit = True
if gasit:
    print("există")
else:
    print("nu există")
```

```
există
```

2. Poziția elementului

Parcurgem după indice și ne oprim când îl găsim, cu `break` (ieșire din buclă).

```
v = [12, 7, 9, 20, 15]
x = 20
poz = -1
for i in range(len(v)):
    if v[i] == x:
        poz = i
        break
print(poz)
```

```
3
```

`poz = -1` înseamnă „nu am găsit nimic”. Un indice nu poate fi `-1` în acest scop, deci nu apar confuzii. Dacă pentru `x = 5` programul afișează `-1`, știm că elementul lipsește.

> **SFAT**
>
> **Scurtătură (opțional).** Python are `x in v`, care dă `True` sau `False`. E același lucru cu bucla din secțiunea 1.

3. Sunt toate elementele pozitive?

Aici folosim steagul invers: presupunem că **da** (`ok = True`) și, dacă găsim **un singur** element care strică regula, punem `False`.

```
v = [4, 8, 15, 16]
ok = True
for x in v:
    if x <= 0:
        ok = False
if ok:
    print("toate sunt pozitive")
else:
    print("cel puțin unul nu este")
```

```
toate sunt pozitive
```

Întrebarea	Steag la început	Când îl schimbi
Există un element cu proprietatea?	`gasit = False`	când găsești unul → `True`
Au toate elementele proprietatea?	`ok = True`	când găsești unul care **nu** o are → `False`

4. Este șirul ordonat crescător?

Un șir este crescător dacă **fiecare element este mai mic sau egal cu următorul**. Comparăm vecinii: `v[i]` cu `v[i + 1]`. Ca să nu ieșim din listă, `i` ajunge doar până la `n - 2`.

```
v = [2, 5, 5, 9, 11]
n = len(v)
ok = True
for i in range(n - 1):
    if v[i] > v[i + 1]:
        ok = False
if ok:
    print("crescător")
else:
    print("nu este crescător")
```

```
crescător
```

`range(n - 1)` dă indicii de la 0 la `n - 2`, deci `v[i + 1]` există mereu.

> **ATENȚIE — GREȘELI FRECVENTE**
>
> Dacă scrii `range(n)`, la ultimul pas ceri `v[n]`, care nu există: `IndexError: list index out of range`.

Pentru `v = [2, 7, 5, 9]` compararea `7 > 5` strică regula, deci `ok` devine `False`.

5. Mai multe întrebări, aceleași unelte

Întrebare	Idee
Există elemente negative?	steag `gasit`, condiția `x < 0`
Sunt toate numerele pare?	steag `ok`, condiția `x % 2 != 0` strică regula
Este șirul descrescător?	ca la crescător, dar `v[i] < v[i + 1]` strică regula
Toate elementele sunt egale?	compari fiecare cu `v[0]`

6. Verifică-te

1. Ce afișează căutarea poziției pentru `v = [3, 6, 6, 9]` și `x = 6`? Dar pentru `x = 4`?
2. De ce `range(n - 1)` și nu `range(n)` la verificarea ordinii?
3. Cu ce valoare pornește steagul `ok` la „toate elementele...”?
4. Scrie condiția care strică regula „șirul este strict crescător” (nu sunt permise elemente egale).
5. Poate un șir cu un singur element să fie crescător?

Răspunsuri

1. `1` (prima apariție, apoi `break`) și `-1`.
2. Pentru că `v[i + 1]` trebuie să existe; ultimul `i` este `n - 2`.
3. `ok = True`.
4. `v[i] >= v[i + 1]`.
5. Da: nu are nicio pereche de vecini care să strice regula, deci bucla nu face nimic și `ok` rămâne `True`.