Mentor Klub

Python tanfolyam

5. alkalom

Elemenkénti feldolgozás

Mentor Klub

- Sokszor előfordul, hogy valamilyen lista szerű adatszerkezet minden egyes elemét fel kell dolgozni
  - Szakszóval végig kell rajtuk iterálni.
- Ezekre az esetekre a `for`, vagy más néven `for-each` ciklusok alkalmasak.
- A for ciklus mindig kiváltható **while** ciklussal, legfeljebb többet kell hozzá kódolni.

For ciklus – Szintaxis és szemantika

Mentor Klub

```
for VÁLTOZÓ in ITERÁLHATÓ_KIFEJEZÉS:
        #Ezeket az utasitás1-et és utasitas2-t addig ismétli a program,
        #amig van elem az ITERÁLHATÓ_KIFEJEZÉS-ben.
        #A for ciklus hatókörében a VÁLTOZÓ elérhető, és mindig az aktuális
        #elemet jelenti
        utasitas1
        utasitas2
utasitas3
#utasitas3 csak akkor lesz végrehajtva, ha az összes elemet
#feldolgoztuk az ITERÁLHATÓ_KIFEJEZÉSBEN
```

Range

- Számsorozatok generálására a range() függvényt használhatjuk.
- Használata:
  - `range(5)`
    - 0,1,2,3,4
  - `range(5,10)`
    - 5,6,7,8,9
  - `range(0,10,2)`
    - 0,2,4,6,8
- Használható for ciklusban is:

```
for v in range(5,10):
    print(v)
```

Speciális utasítások ciklusok esetén

Mentor Klub

`break`

- A ciklus futását befejezi, a végrehajtás a ciklus után folytatódik
- Például ha egy keresés során megtaláltuk az elemet, amit kerestünk, akkor a break utasítás segitségével megszakítjuk a ciklus futását, kiugrunk a ciklusból.
  - Természetesen azt a tényt, hogy megtaláltuk, amit kerestünk, jelölni kell valahogy.

`continue`

- A ciklusmag futását félbeszakítja, a vezérlés a ciklusfeltétel kiértékelésével folytatódik.
- Például adjuk össze a 8-cal osztható számokat az [1,1000] intervallumon. Ahelyett, hogy bonyolult if-else vezérlési szerkezetet irunk, a 8-cal nem osztható számok esetén a continue utasítással a következő iterációval folytatjuk.
  - Azért oda kell figyelni a ciklusváltozó növelésére, mert könnyen végtelen ciklus lehet a vége

Alapvető algoritmusok - Összegzés

Mentor Klub

- Összegzés:
  - Egy adott, számokat tartalmazó, iterálható adatszerkezet összes elemét sorra vesszük, és összeadjuk őket.
  - Az algoritmus kimenete a végeredmény.
  - Például: Adjuk össze a számokat 1-től 1000-ig.
- Általánositás:
  - Tetszőleges műveletet végzünk el a számokon.
  - Tetszőleges adattipusokon végzünk tetszőleges műveletet.

Alapvető algoritmusok - Számlálás

Mentor Klub

- Számlálás:
  - Egy adott, számokat tartalmazó, iterálható adatszerkezet összes elemét sorra vesszük, és megszámoljuk azokat, amelyek egy tetszőleges feltételnek megfelelnek.
  - Az algoritmus kimenete azon elemek száma, melyek megfelelnek a feltételnek.
  - Például: Hány darab 8-cal osztható szám van 1 és 1000 között?
- Általánositás:
  - Tetszőleges adattipusra vonatkozik a feltétel.
  - Nem számlálunk, hanem például összegzést végzünk.
    - Adjuk össze az 1 és 1000 közötti, 8-cal osztható számokat.

Alapvető algoritmusok - Lineáris keresés

Mentor Klub

- Lineáris keresés:
  - Egy adott, számokat tartalmazó, iterálható adatszerkezetben keressük azt a számot, ami megfelel egy feltételnek.
  - Az algoritmus kimenete megválaszolja, hogy szerepel-e az adatszerkezetben olyan elem, amely megfelel a feltételnek.
    - True/False vagy None/Az elem amit kerestünk
  - Például: A [1,2,3,4,5,6] listában van 3-mal osztható szám?

- Azért nevezzük lineáris keresésnek, mert legrosszabb esetben az egész listán végig kell menni 1x, hogy megtaláljuk a keresett elemet.
  - A legrosszabb eset azt jelenti, hogy nincs ilyen elem a listában. Ilyenkor a lista elemszáma mennyiségű feltételvizsgálatot végzünk.

Alapvető algoritmusok – Maximum keresés

Mentor Klub

- Maximum keresés:
  - Egy adott, számokat tartalmazó, iterálható adatszerkezetben keressük a legnagyobb elemet.
  - Az algoritmus kimenete megválaszolja, hogy melyik a legnagyobb elem az adatszerkezetben.
  - Például: A [1,2,7,4,5,6] listában mi a legnagyobb érték?