

# 05\_PythonBev\_Rendezes\_Comprehension

February 12, 2025



## 1 Pythonos “ügyességek”

Rendezés; Haladó indexelési technikák; Comprehension; Kicsomagolás, értékcsere és haladó iterációk

### 1.1 Még néhány hasznos lista eljárás:

rendezés, minimum, maximum, összeg

#### 1.1.1 Lista rendezése helyben

`.sort()` eljárás

```
[61]: # Lista rendezése helyben.  
l = [10, 2, 11, 3]  
l.sort()  
print(l) # az eredeti lista megváltozott!
```

[2, 3, 10, 11]

```
[62]: # Rendezés csökkenő sorrendbe - reverse paraméter  
l = [10, 2, 11, 3]  
l.sort(reverse = True)  
print(l)
```

[11, 10, 3, 2]

```
[63]: # Fontos, hogy az elemek összehasonlíthatók legyenek!  
l = [2, 1, 'alma']  
l.sort()
```

```
-----  
TypeError                                Traceback (most recent call last)  
/tmp/ipykernel_175381/1268567011.py in <module>  
      1 # Fontos, hogy az elemek összehasonlíthatók legyenek!  
      2 l = [2, 1, 'alma']
```

```
----> 3 l.sort()
```

```
TypeError: '<' not supported between instances of 'str' and 'int'
```

```
[64]: # Ha csak sztringeket tartalmaz a lista, akkor lehet rendezni.  
l = ['alma', 'szilva', 'körte']  
l.sort()  
print(l)
```

```
['alma', 'körte', 'szilva']
```

Sztringeket a lexikografikus rendezésnek megfelelően, azaz az “ABC sorrend” szerint rendezi.

### 1.1.2 Gyűjtemény rendezése új listába.

`sorted()` beépített függvény

```
[ ]: # Kollekciónak rendezése új listába.  
l1 = [10, 4, 20, 5]  
sorted(l1)
```

```
[ ]: [4, 5, 10, 20]
```

```
[66]: # Tuple elemeit is rendezhetjük új listába.  
t1 = (10, 4, 20, 5)  
sorted(t1)
```

```
[66]: [4, 5, 10, 20]
```

```
[67]: # ...és halmaz elemeit is.  
sorted({3, 676, 11, 42})
```

```
[67]: [3, 11, 42, 676]
```

```
[68]: # Szótár esetén a sorted a kulcsokat rendezi.  
d1 = {"barack": 10, "alma": 3, "mandula": 25}  
sorted(d1)
```

```
[68]: ['alma', 'barack', 'mandula']
```

```
[69]: # Párok listájának rendezése (lexikografikusan).  
l = [('sör', 10), ('bor', 20), ('pálinka', 30), ('bor', 5)]  
sorted(l)
```

```
[69]: [('bor', 5), ('bor', 20), ('pálinka', 30), ('sör', 10)]
```

### 1.1.3 minimum, maximum, összeg

`min(), max(), sum()`

```
[1]: l1 = [10, 4, 20, 5]
```

```
[70]: # próbáljuk ki mindet
print("maximum", max(l1))
print("minimum", min(l1))
print("összeg", sum(l1))
```

maximum 20

minimum 4

összeg 39

## 1.2 Haladó indexelés (szeletelés/[slicing](#))

- A slice jelölésmód szintaxisa [alsó határ: felső határ: lépésköz].
- A kiválasztás intervalluma felülről nyitott, azaz a felső határ adja meg az első olyan indexet, amelyet már éppen nem választunk ki.

```
[2]: data = ['alma', 'banán', 10, 20, 30]
```

```
[3]: # első 3 elem kiválasztása
print(data [0:3])
print(data [ :3])
```

['alma', 'banán', 10]

['alma', 'banán', 10]

```
[4]: # Minden második elem kiválasztása:
data [::2]
```

```
[4]: ['alma', 10, 30]
```

```
[5]: # Minden kivéve az utolsó elemet
# Használhatunk negatív indexeket is
data [:-1]
```

```
[5]: ['alma', 'banán', 10, 20]
```

```
[6]: # utolsó 3 elem
data[-3 : ]
```

```
[6]: [10, 20, 30]
```

```
[7]: # lista elemek fordított sorrendben
data[::-1]
```

```
[7]: [30, 20, 10, 'banán', 'alma']
```

Szövegre is működik a haladó indexelés!

```
[ ]: text = "abcde"
```

```
[77]: # Írassuk ki a text sztringet fordított sorrendben!  
text[::-1]
```

```
[77]: 'edcba'
```

## 1.3 Comprehension

- A comprehension gyűjtemények tömör megadását teszi lehetővé.
- [Comprehension a Python dokumentációban](#)
- Hasonlít a matematikában alkalmazott, tulajdonság alapján történő halmazmegadásra (példa: a páratlan számok halmaza megadható  $\{2k + 1 \mid k \in \mathbb{Z}\}$  módon).

### 1.3.1 Feltétel nélküli comprehension

**Példa:** Állítsuk elő az első 10 négyzetszám listáját / halmazát!

```
[9]: # Ismert módszer - gyűjtőváltozó használatával  
N = 10  
l = []  
for elem in range(1, N+1):  
    l.append(elem**2)  
print(l)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
[79]: # Ugyanez tömörebben, lista comprehension-nel is megoldható  
N = 10  
l = [ elem**2 for elem in range(1, N + 1) ]  
print(l)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
[80]: # Állítsuk elő az első 10 négyzetszám halmazát!  
N = 10  
h_nsz = { elem**2 for elem in range(1, N + 1) }  
print(h_nsz)
```

```
{64, 1, 4, 36, 100, 9, 16, 49, 81, 25}
```

De lehet könnyen szótárat is készíteni.

**Példa:** Párosítsuk össze a számokat a négyzetükkel szótárat használva

```
[81]: # szótár készítése comprehenssionnel
N = 10
{ elem: elem**2 for elem in range(1, N+1) }
```

```
[81]: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

**Feladat:** Állítsunk elő egy szótárat, amely az angol kisbetűs magánhangzókhoz hozzárendeli az ASCII-kódjukat!

Segítség: a kód előállításához használjuk az `ord()` függvényt!

```
[82]: # Használjunk először gyűjtőváltozót!
vowels = "aeiou"
d = {}

for v in vowels:
    d[v] = ord(v)

print(d)
```

```
{'a': 97, 'e': 101, 'i': 105, 'o': 111, 'u': 117}
```

```
[83]: # Ugyanez tömörebben, szótár comprehension-nel:
vowels = "aeiou"
{ v: ord(v) for v in vowels }
```

```
[83]: {'a': 97, 'e': 101, 'i': 105, 'o': 111, 'u': 117}
```

**Feladat:** Párok tagjainak megcserélése egy listában.

```
[17]: pairs = [('alma', 10), ('körte', 20), ('barack', 30)]
```

```
[18]: # cseréljük meg a párok tagjait!
[ (p[1], p[0]) for p in pairs]
```

```
[18]: [(10, 'alma'), (20, 'körte'), (30, 'barack')]
```

### 1.3.2 Feltételes comprehension

**Példa:** Állítsuk elő az 1-től 10-ig előforduló négyzetszámok listáját!

```
[85]: # Páros négyzetszámok gyűjtőváltozó használatával
N = 10
l = []
for elem in range(1, N + 1):
    if elem % 2 == 0:
        l.append(elem**2)
print(l)
```

[4, 16, 36, 64, 100]

```
[86]: # Páros négyzetszámok comprehension:
N = 10
l = [ elem**2 for elem in range(1, N +1) if elem%2 == 0 ]
print(l)
```

[4, 16, 36, 64, 100]

**Feladat:** Állítsunk elő egy szótárat, amely az angol kisbetűs magánhangzókhoz, az 'u'-t leszámítva hozzárendeli az ASCII-kódjukat!

```
[11]: # Feltételes szótár comprehension.
vowels = "aeiou"
{ v: ord(v) for v in vowels if v != "u" }
```

[11]: {'a': 97, 'e': 101, 'i': 105, 'o': 111}

**Feladat:** Magánhangzók megszámlálása angol kisbetűs szövegben Comprehensionnel

```
[12]: text = 'This is a short text for testing the algorithm.'
vowels = 'aeiou'
```

```
[13]: # feltételes comprehensiont használva a szöveg karaktereire
sum(1 for elem in text.lower() if elem in vowels)
```

[13]: 12

```
[14]: # másik megoldás count() eljárást használva az összes magánhangzóval
sum(text.lower().count(mgh) for mgh in vowels)
```

[14]: 12

**Feladat:** Magánhangzók statisztikája angol kisbetűs szövegben Comprehensionnel

```
[15]: text = 'This is a short text for testing the algorithm.'
vowels = 'aeiou'
```

```
[16]: # comprehension + count() eljárás
{ mgh: text.count(mgh) for mgh in vowels.lower() }
```

[16]: {'a': 2, 'e': 3, 'i': 4, 'o': 3, 'u': 0}

## 1.4 Kicsomagolás (unpacking)

- többszörös értékadásra
- értékcsereire !!!
- for ciklusban is alkalmazható

```
[19]: ## Általános eset: 'alma' ; 2 és [30,40] listához hozzáférés?
[x, (y, z)] = ['alma', (2, [30, 40])]
print(x)
print(y)
print(z)
```

```
alma
2
[30, 40]
```

```
[20]: # Ha a bal és jobb oldal nem illeszthető egymásra, hibát kapunk.
x, y, z = 1, 2
```

```
-----
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_78500/3325170505.py in <module>
      1 # Ha a bal és jobb oldal nem illeszthető egymásra, hibát kapunk.
----> 2 x, y, z = 1, 2

ValueError: not enough values to unpack (expected 3, got 2)
```

```
[21]: # Többszörös értékadásra is jó
x, y = 1, 2
print(f"x = {x} és y = {y}")
```

```
x = 1 és y = 2
```

**Fontos alkalmazás:** Értéket cserélni is lehet így változók közt!

```
[97]: x, y = 1, 2      # értékadás
print(f"x = {x} és y = {y}")

x, y = y, x          # értékcseré
print(f"x = {x} és y = {y}")
```

```
x = 1 és y = 2
x = 2 és y = 1
```

**Példa** Írjuk ki a képernyőre az alábbi párok listájából a pár első elemét!

Majd gyűjtsük ki listába is comprehensiont használva!

```
[24]: pairs = [('sör', 10), ('bor', 20), ('rum', 30)]
```

```
[25]: # Kicsomagolás alkalmazható for ciklusnál is
for x, y in pairs:
    print(x)
```

```
sör  
bor  
rum
```

```
[26]: # gyűjtsük listába az előző párok listájából a sztringeket  
[ x for x, y in pairs]
```

```
[26]: ['sör', 'bor', 'rum']
```

## 1.5 Haladó iterálási technikák

### 1.5.1 enumerate

- [enumerate a dokumentációban](#)
- Alkalmazás: Sorindex nyilvántartása szövegfájl feldolgozásnál.

```
[27]: x = ['alma', 'körte', 'szilva']
```

```
[28]: ## Iterálás az elemeken és indexeken egyszerre, hagyományos megoldás.  
for elem in range(len(x)):  
    print(elem , x[elem])
```

```
0 alma  
1 körte  
2 szilva
```

```
[ ]: ## Ugyanez elegánsabban, enumerate-tel:  
for i, xi in enumerate(x):  
    print(i, xi)
```

```
0 alma  
1 körte  
2 szilva
```

```
[ ]: ## Az enumerate eredménye egy iterálható objektum  
# átalakítható párok listájává  
list(enumerate(x))
```

```
[(0, 'alma'), (1, 'körte'), (2, 'szilva')]
```

```
[ ]: ## Az enumerate kicsomagolás nélkül is használható.  
for elem in enumerate(x):  
    print(elem[0], elem[1])
```

```
0 alma  
1 körte  
2 szilva
```



### 1.5.2 zip

- [zip a dokumentációban](#)
- könnyű adatpárokat készíteni

```
[37]: x = ['alma', 'körte', 'barack']  
      y = [10, 20, 30]  
      z = ['X', 'Y', 'Z']
```

```
[38]: ## Iterálás több szekvencián egyszerre, hagyományos megoldás.  
      for i in range(len(x)):  
          print(x[i], y[i])
```

```
alma 10  
körte 20  
barack 30
```

```
[39]: ## Ugyanez elegánsabban, zip-pel:  
      for elem in zip(x, y):  
          print(elem[0], elem[1])
```

```
alma 10  
körte 20  
barack 30
```

```
[40]: # Ugyanez elegánsabban, zip-pel kicsomagolt változókkal  
      for szoveg, szam in zip(x, y):  
          print(szoveg, szam)
```

```
alma 10  
körte 20  
barack 30
```

```
[41]: ## A zip eredménye egy iterálható objektum.  
      # Ami átalakítható listává.  
      list(zip(x, y))
```

```
[41]: [('alma', 10), ('körte', 20), ('barack', 30)]
```

```
[42]: # Ha szekvenciák hossza nem azonos, akkor az eredmény a rövidebb hosszát veszi  
      ↳ fel.  
      alist = [10, 20]  
      blist = ['a', 'b', 'c']  
      list(zip(alist, blist))
```

```
[42]: [(10, 'a'), (20, 'b')]
```

```
[44]: # A zip kettőnél több szekvenciára is alkalmazható.  
list(zip(x, y ,z))
```

```
[44]: [('alma', 10, 'X'), ('körte', 20, 'Y'), ('barack', 30, 'Z')]
```