



# Szkriptyelvek

Szathmáry László  
Debreceni Egyetem  
Informatikai Kar

## 3. Gyakorlat

- írás a standard kimenetre
- listák (folyt.)
- ciklusok (for, while)

(utolsó módosítás: 2026. jan. 16.)

2025-2026, 2. félév



```

>>> a = [0, 1, 2, 3, 4]
>>> a
[0, 1, 2, 3, 4]
>>> for e in a:
...     print(e)
...
0
1
2
3
4
>>> print(2, 'a', 5, '12')
2 a 5 12
>>> print(2, 'a', 5, '12', sep='')
2a512
>>> print(1, 'a'); print(2, 'b')
1 a
2 b
>>> print(1, 'a', end=''); print(2, 'b')
1 a2 b
>>>
>>> import sys
>>> print('Warning! Reactor meltdown!', file=sys.stderr)
Warning! Reactor meltdown!
>>>

```

függvény lett

*sep*: elemek közti elválasztó jel

*end*: utolsó elem utáni végjel

*file*: hova, melyik file-ba kerüljön a kimenet

Help on built-in function print in module builtins:

```

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

```

alapértelmezett értékek

# Írás a standard kimenetre



```

1  >>> a = range(5)
2  >>> a
3  [0, 1, 2, 3, 4]
4  >>> for e in a:
5  ...     print e
6  ...
7  0
8  1
9  2
10 3
11 4
12 >>> for e in a:
13 ...     print e,
14 ...
15 0 1 2 3 4
16 >>>
17 >>> import sys
18 >>>
19 >>> for e in a:
20 ...     sys.stdout.write(e)
21 ...
22 Traceback (most recent call last):
23   File "<stdin>", line 2, in <module>
24 TypeError: expected a character buffer object
25 >>>
26 >>> for e in a:
27 ...     sys.stdout.write(str(e))
28 ...
29 01234>>>
  
```

The code demonstrates different ways to print elements from a list 'a' (containing integers 0-4) to standard output. It shows three methods: 1. Using the built-in print function with a trailing newline, which adds an extra space before the next command. 2. Using the built-in print function without a trailing newline, which prints all elements on the same line separated by spaces. 3. Using the sys.stdout.write method, which gives full control over the output, resulting in the expected output of '01234'.

1

(„\n”)

2

(szóköz)

3

(„full control”)

## Néhány lista művelet

```
1  >>> a = [1, 2, 3]
2  >>> a
3  [1, 2, 3]
4  >>> a.append(20)
5  >>> a
6  [1, 2, 3, 20]
7  >>> a.pop(0)
8  1
9  >>> a
10 [2, 3, 20]
11 >>> del a
12 >>> a
13 Traceback (most recent call last):
14   File "<stdin>", line 1, in <module>
15     NameError: name 'a' is not defined
16 >>> a = [1, 2, 3]
17 >>> del a[1]
18 >>> a
19 [1, 3]
```

**Verem használata:**

*lista.append(elem)*  
*lista.pop()*

## Extra: **sor** adatszerkezet

```
>>> from collections import deque
>>>
>>> q = deque([3, 4, 5])
>>> q
deque([3, 4, 5])
>>> q.append(6)
>>> q.append(7)
>>> q
deque([3, 4, 5, 6, 7])
>>> q.popleft()
3
>>> q
deque([4, 5, 6, 7])
```

További kollekciók:

<http://docs.python.org/3/library/collections.html>

```
1  >>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8]
2  >>> a
3  [0, 1, 2, 3, 4, 5, 6, 7, 8]
4  >>> a[2:5]
5  [2, 3, 4]
6  >>> a[2:5] = []
7  >>> a
8  [0, 1, 5, 6, 7, 8]
9  >>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8]
10 >>> a[2:5] = [10, 20, 30, 40]
11 >>> a
12 [0, 1, 10, 20, 30, 40, 5, 6, 7, 8]
```

több elem törlése

több elem cseréje

# Néhány gyakori lista metódus

- `list.append(elem)`  
Elem beszúrása a lista végére. Nem tér vissza a listával, a listát helyben módosítja.
- `list.insert(index, elem)`  
Elem beszúrása az adott index pozíción lévő elem előtt. A tőle jobbra lévő elemeket eggyel jobbra mozgatja.
- `list.extend(list2)`  
A list2-ben lévő elemeket a lista végére beszűrja. A + ill. a += operátorok hasonlóan működnek.
- `list.index(elem)`  
Adott elem keresése a listában. Ha benne van, akkor az elem indexével tér vissza. Ha nincs benne, akkor ValueError kivételt dob. (Ha el akarjuk kerülni ezt a kivételt, használjuk az „in” operátort.)
- `list.remove(elem)`  
Az adott elem első előfordulását eltávolítja a listából. Ha nincs benne, akkor ValueError kivétel lép fel.
- `list.sort()`  
Helyben rendezi a listát (nem tér vele vissza).
- `list.reverse()`  
Helyben megfordítja az elemek sorrendjét (nem tér vissza a listával).
- `list.pop(index)`  
Az adott indexű elemet eltávolítja s ezzel az elemmel tér vissza. Ha az indexet nem adjuk meg, akkor a legjobboldalibb elemmel tér vissza.

# Lista rendezése

1

```
>>> a = [8, 5, 1, 3]
>>> a
[8, 5, 1, 3]
>>> sorted(a)
[1, 3, 5, 8]
>>> help(sorted)
```

`Help on built-in function sorted in module builtins:`

```
sorted(iterable, key=None, reverse=False)
    Return a new list containing all items from the iterable in ascending order.

    A custom key function can be supplied to customise the sort order, and the
    reverse flag can be set to request the result in descending order.
```

```
>>> sorted(a, reverse=True)
[8, 5, 3, 1]
>>> a
[8, 5, 1, 3]
>>> a = sorted(a)
>>> a
[1, 3, 5, 8]
>>>
>>> a = ['bela', 'aladar', 'denes', 'cecil']
>>> sorted(a)
['aladar', 'bela', 'cecil', 'denes']
```

2

```
>>> a.sort()
>>> a
['aladar', 'bela', 'cecil', 'denes']
>>>
```

helyben rendez

## Néhány gyakori művelet listákkal

```
1  >>> li
2  [9, 8, 1, 4, 8, 2, 3, 2]
3  >>> max(li)
4  9
5  >>> min(li)
6  1
7  >>> sum(li)
8  37
```



ezek beépített függvények  
(lásd még L függelék)

**Feladat:** írunk függvényt, mely kap egy listát s visszaadja a listában lévő elemek szorzatát.

## split / join

```

1  >>> a = ['aa', 'bb', 'cc', 'dd']
2  >>> a
3  ['aa', 'bb', 'cc', 'dd']
4  >>> ':'.join(a)
5  'aa:bb:cc:dd'
6  >>> ','.join(a)
7  'aa,bb,cc,dd'
12 >>> print '\n'.join(a)
13 aa
14 bb
15 cc
16 dd
17 >>>
18 >>> b = 'aa:bb:cc:dd'
19 >>> b
20 'aa:bb:cc:dd'
21 >>> b.split(':')
22 ['aa', 'bb', 'cc', 'dd']
23 >>> s = 'aladar      bela  cecil'
24 >>> s.split()
25 ['aladar', 'bela', 'cecil']

```

lista → sztring

valamilyen szeparátor  
mentén

sztring → lista

# range / xrange



```

4  >>> range(20)
5  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
6  >>> for i in range(10):
7      ...     print i,
8  ...
9  0 1 2 3 4 5 6 7 8 9
10 >>> for i in xrange(10): ← kevesebb memória kell neki
11 ...     print i,
12 ...
13 0 1 2 3 4 5 6 7 8 9
14 >>>
15 >>> range(5,20)
16 [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
17 >>>
18 >>> range(5,20,2) ← harmadik paraméter:
19 [5, 7, 9, 11, 13, 15, 17, 19]           lépésköz
20 >>>
21 >>> range(10, 0, -1) ← csökkenő sorozat
22 [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
23 >>>

```

## range



A Python 3-ban már csak a „range” található meg, de a Python 2 „xrange”-éhez hasonló módon működik.

```
>>> xrange(5)
Traceback (most recent call last):
  File "<input>", line 1, in <module>
NameError: name 'xrange' is not defined
>>>
>>> range(5)
range(0, 5)
>>>
>>> list(range(5))
[0, 1, 2, 3, 4]
>>>
>>> for i in range(5):
...     print(i)
...
...
0
1
2
3
4
>>>
```



## Feladat

Számoljuk ki az egész számok összegét 1-től 100-ig.

Rendelkezésre álló idő: 30 másodperc.

Link: <https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=Py3.20121001b>

# for és while ciklus

```

>>> for i in range(10):
...     print(i, end=' ')
...
...
0 1 2 3 4 5 6 7 8 9
>>>
>>> i = 0
>>> while i < 10:
...     print(i, end=' ')
...     i += 1
...
...
0 1 2 3 4 5 6 7 8 9
>>>
>>> li = ['aladar', 'bela', 'cecil']
>>>
>>> for e in li:
...     print(e, end=' ')
...
...
aladar bela cecil
>>>
>>> i = 0
>>> size = len(li)
>>> while i < size:
...     print(li[i], end=' ')
...     i += 1
...
...
aladar bela cecil
>>>

```

*for ciklus*

ugyanez *while* ciklussal

**HF:** list1.py és list2.py kiegészítése.



házi feladat



# Feladatok

1. [\[20120905b\]](#) lista elemeinek szorzata
2. [\[20121001b\]](#) egész számok összege 1-től 100-ig (másik változat is)
3. [\[20120818bc\]](#) listák #1
4. [\[20120922a\]](#) listák #2
5. [\[20120815h\]](#) a-z
6. [\[20130225a\]](#) sztring tisztítása
7. [\[20120815d\]](#) ASCII táblázat
8. [\[20120820b\]](#) decimális → bináris konverter
9. [\[20120818e\]](#) ezernél kisebb pozitív egész számok (PE #1)
10. [\[20120815l\]](#) rejtélyes üzenet
11. [\[20120815e\]](#) palindróm (iteratív módszer)

