



# Bevezetés a Python programozási nyelvbe

Szathmáry László  
Debreceni Egyetem  
Informatikai Kar

## 5. Gyakorlat

- halmaz (set)
- szótár (dictionary)
- zip

(utolsó módosítás: 2018. márc. 11.)

2017-2018, 2. félév



# set

```
4 >>> kosar = ['alma', 'ananasz', 'banan', 'alma', 'narancs', 'banan']
5 >>> gyumolcs = set(kosar)
6 >>> gyumolcs
7 set(['ananasz', 'banan', 'alma', 'narancs'])
8 >>> type(gyumolcs)
9 <type 'set'>
10 >>> li = list(gyumolcs)
11 >>> li
12 ['ananasz', 'banan', 'alma', 'narancs']
13 >>> type(li)
14 <type 'list'>
15 >>> sorted(li)
16 ['alma', 'ananasz', 'banan', 'narancs']
17 >>> gyumolcs
18 set(['ananasz', 'banan', 'alma', 'narancs'])
19 >>> 'kiwi' in gyumolcs
20 False
21 >>> 'alma' in gyumolcs
22 True
```

ismétlődések megszüntetése

adott elem szerepel-e a halmazban

<https://docs.python.org/3/library/stdtypes.html#set-types-set-frozenset>

## Feladat:

Legyenek adottak a következő elemek: [5, 2, 3, 5, 1, 4, -200, 5, 1, 3, 2, 2, 5].  
Távolítsuk el a duplikátumokat, vagyis egy elem csak 1x szerepeljen az eredményben. Az eredményben az elemek legyenek rendezve. ([halmaz01](#))

listából halmaz

```
11 >>> a = ['alma', 'banan', 'citrom']
12 >>> a = set(a)
13 >>> a
14 set(['banan', 'citrom', 'alma'])
15 >>> b = set()
16 >>> b.add('banan')
17 >>> b.add('narancs')
18 >>> b
19 set(['banan', 'narancs'])
20 >>> a.union(b)
21 set(['banan', 'citrom', 'alma', 'narancs'])
22 >>> a.intersection(b)
23 set(['banan'])
24 >>> a.difference(b)
25 set(['citrom', 'alma'])
26 >>> a
27 set(['banan', 'citrom', 'alma'])
28 >>> a.remove('citrom')
29 >>> a
30 set(['banan', 'alma'])
```

üres halmaz;  
halmaz bővítése

klasszikus  
halmazműveletek

elem törlése

# dictionary

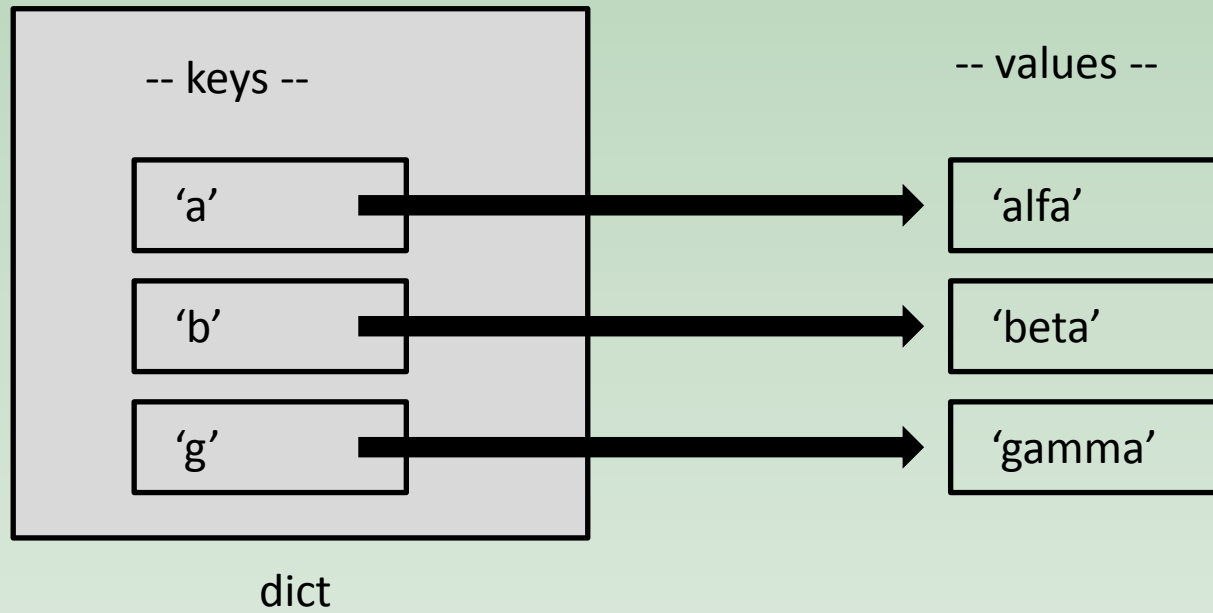
kulcs / érték  
párok tárolása



```
1  >>> d = { }
2  >>> d['a'] = 'alfa'
3  >>> d['b'] = 'beta'
4  >>> d['g'] = 'gamma'
5  >>> d
6  {'a': 'alfa', 'b': 'beta', 'g': 'gamma'}
7  >>> d['a']
8  'alfa'
9  >>> d['o']
10 Traceback (most recent call last):
11     File "<stdin>", line 1, in <module>
12     KeyError: 'o'
13 >>> d.get('o')
14 >>> d.get('a')
15 'alfa'
16 >>> 'a' in d
17 True
18 >>> 'o' in d
19 False
20 >>> d['a'] = 'ALFA'
21 >>> d['a']
22 'ALFA'
23 >>> d
24 {'a': 'ALFA', 'b': 'beta', 'g': 'gamma'}
```

üres szótár  
( vagy: `d = dict()` )

adott kulcsú elem  
szerepel-e benne



```
>>> d
{'b': 'beta', 'g': 'gamma', 'a': 'alfa'}
>>> d['o'] = 'omega'
>>> d
{'o': 'omega', 'b': 'beta', 'g': 'gamma', 'a': 'alfa'}
>>> d.keys()
dict_keys(['o', 'b', 'g', 'a'])
>>> type(d.keys())
<class 'dict_keys'>
>>> d.values()
dict_values(['omega', 'beta', 'gamma', 'alfa'])
>>> type(d.values())
<class 'dict_values'>
>>> for k in sorted(d.keys()):
...     print("key:", k, "=>", d[k])
...
key: a => alfa
key: b => beta
key: g => gamma
key: o => omega
>>> d.items()
dict_items([('o', 'omega'), ('b', 'beta'), ('g', 'gamma'), ('a', 'alfa')])
>>> type(d.items())
<class 'dict_items'>
>>> for k, v in d.items():
...     print(k, v)
...
o omega
b beta
g gamma
a alfa
>>> list(d.keys())
['o', 'b', 'g', 'a']
```

← az elemek sorrendje  
tetszőleges

### iterátorok:

d.keys()  
d.values()  
d.items()

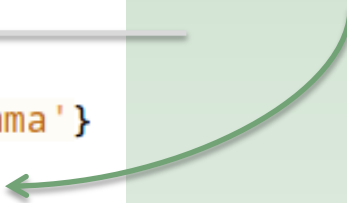
iterátorok használata  
ciklusban

← Iterátorból lista. Az elemek  
sorrendje tetszőleges.

HF: [dict1.py](#)

## elem törlése szótárból

```
1  >>> a = 6
2  >>> a
3  6
4  >>> del a
5  >>> a
6  Traceback (most recent call last):
7    File "<stdin>", line 1, in <module>
8  NameError: name 'a' is not defined
9  >>>
10 >>> li = list(range(5))
11 >>> li
12 [0, 1, 2, 3, 4]
13 >>> del li[-1]
14 >>> li
15 [0, 1, 2, 3]
16 >>>
17 >>> d
18 {'a': 'alfa', 'b': 'beta', 'o': 'omega', 'g': 'gamma'}
19 >>> del d['b']
20 >>> d
21 {'a': 'alfa', 'o': 'omega', 'g': 'gamma'}
```



# Haladó

## zip

A paraméterül kapott iterálható objektumok elemeit összepárosítja egy-egy tuple-ben. A zip-nek kettőnél több paraméter is megadható. A zip egy iterátort ad vissza.

```
In [1]: a = [1, 2, 3]

In [2]: b = ['one', 'two', 'three']

In [3]: c = ['x', 'y', 'z']

In [4]: list(zip(a, b))
Out[4]: [(1, 'one'), (2, 'two'), (3, 'three')]

In [5]: list(zip(a, b, c))
Out[5]: [(1, 'one', 'x'), (2, 'two', 'y'), (3, 'three', 'z')]

In [6]: list(zip("abc", "def"))
Out[6]: [('a', 'd'), ('b', 'e'), ('c', 'f')]
```

Több információ itt: <https://goo.gl/nY551i>





# Feladatok

1. [[20120904a](#)] duplikátumok eltávolítása (halmaz)
2. [[20120905a](#)] dictionary #1
3. [[20120921a](#)] ékezetek eltávolítása
4. [[20130218b](#)] bizonyos karakterek
5. [[20120818h](#)] 100 db 50-jegyű szám (PE #13) [csak az **A** változatot]
6. [[20120816a](#)] 8 királynő
7. [[20180306b](#)] AoC2017, Day 1, Part 1 (Inverse Captcha)



```
1 >>> d
2 {'a': 'alfa', 'b': 'beta', 'g': 'gamma'}
3 >>> d['o'] = 'omega'
4 >>> d
5 {'a': 'alfa', 'b': 'beta', 'o': 'omega', 'g': 'gamma'}
6 >>> d.keys()
7 ['a', 'b', 'o', 'g']
8 >>> d.values()
9 ['alfa', 'beta', 'omega', 'gamma']
10 >>> for k in sorted(d.keys()):
11 ...     print 'kulcs:', k, '->', d[k]
12 ...
13 kulcs: a -> alfa
14 kulcs: b -> beta
15 kulcs: g -> gamma
16 kulcs: o -> omega
17 >>> d.items()
18 [('a', 'alfa'), ('b', 'beta'), ('o', 'omega'), ('g', 'gamma')]
19 >>> for k, v in d.iteritems():
20 ...     print k, v
21 ...
22 a alfa
23 b beta
24 o omega
25 g gamma
```

az elemek sorrendje  
tetszőleges

kulcsok listája;  
értékek listája

ciklusban inkább  
az iteritems() fv.-t  
használjuk

`d.items()` : tuple-ök listája (mint a `range` függvény)

`d.iteritems()` : generátor, a tuple-öket egyenként állítja elő (mint az `xrange` függvény)