Imperatív programozás

6. Előadás

Python típusok (folytatás)

Függvények

Típusok + műveleteik

Listák

- mutable (változtatható)
- heterogén lista

```
>>> lista = ["szo", 12, 3.5]
>>> lista[1]
12
>>> lista[1:3]
[12, 3.5]
>>> lista[0:-1]
['szo', 12]
```

```
>>> lista[1] = 24
>>> lista
['szo', 24, 3.5]
>>> lista[2] = lista[2] + 4
>>> lista
['szo', 24, 7.5]
>>> lista[1:2] = [12, "abc", 8]
>>> lista
['szo', 12, 'abc', 8, 7.5]
>>> lista[-2:] = []
>>> lista
['szo', 12, 'abc']
```

Listák további műveletei

lista = [3, 4, 5, 2, 1]

- új elem hozzávétele:
    - lista.append(6)

        lista == [3, 4, 5, 2, 1, 6]
- megfordítás:
    - lista.reverse()

        lista ==[1, 2, 5, 4, 3]
- rendezés:
    - lista.sort()

        lista ==[1, 2, 3, 4, 5]

Immutable vs Mutable

s = "peLDa"

- s.upper() == 'PELDA'
  - s == "peLDa"

l = [5,4,2]

- l.sort() == None
  - l == [2,4,5]

Magasabbrendű függvények listákra

lista = [1, 2, 3, 4, 5, 6]

- filter:
    - filter(lambda x: x%3==0, lista) == [3, 6]

- map:
    - map(lambda x: x**2, lista) == [1, 4, 9, 16, 25, 36]

      map(lambda x, y : x+y, lista, lista) == [2, 4, 6, 8, 10, 12]

      map(lambda x, y: (x, y), [1, 2], [1, 2, 3, 4])

      == [(1, 1), (2, 2), (None, 3), (None, 4)]

      map(None, [1, 2, 3], [2, 4, 5]) == [(1, 2), (2, 4), (3, 5)]

Lista konstruktorok

- [x**2 for x in range(1,5)] == [1, 4, 9, 16]

- [x**2 for x in range(1,5) if x%2 == 0] == [4, 16]

- [x*y for x in [2,4] for y in [3,5]] == [6, 10, 12, 20]

- [x*y for x, y in zip([2,4],[3,5])] == [6, 20]

Megjegyzés

zip([1,2,3],[5,6]) == [(1, 5), (2, 6)]

Típusok + műveleteik

Rendezett n-esek (tuple-k)

- immutable
- heterogén elemek

```
>>> tuple = (1, 2, 'abc')
>>> tuple
(1, 2, 'abc')
>>> tuple = 1, 2, 'xyz'
>>> tuple
(1, 2, 'xyz')
```

```
>>> tuple[2]
'xyz'
>>> tuple[0:1]
(1,)
>>> tuple[0:2]
(1, 2)
```

Típusok + műveleteik

Rendezett n-esek (tuple-k)

Immutable!

```
>>> tuple[2] = 'abc'
Traceback (most recent call last):
  File "<pyshell#22>", line 1, in <module>
    tuple[2] = 'abc'
TypeError: 'tuple' object does not support
  item assignment
```

Típusok + műveleteik

Halmaz

- mutable érték
- létrehozás

```
lista = [1,3,2,1,4,2,3]
s = set(lista)
        s == set([1, 2, 3, 4])
```

- eleme-e reláció:

```
s = set([1, 2, 3, 4])
(4 in s) == True
(5 in s) == False
```

Típusok + műveleteik

Halmaz

s == set([1, 2, 3, 4])

- új elem hozzávétele:

    s.add(5)

        s == set([1, 2, 3, 4, 5])

- elem törlése

    s.discard(3)

        s == set([1, 2, 4])

Típusok + műveleteik

Halmaz

Halmazműveletek

a = set([1, 2, 3]), b = set([2, 3, 4])

- kivonás:
  a - b == set([1])

- unió:
  a | b == set([1, 2, 3, 4])

- metszet:
  a & b == set([2, 3])

- xor, (a|b) - (a&b):
  a ^ b == set([1,4])

Típusok + műveleteik

Szótár adatszerkezet

- mutable érték

tel = {'Peter': 1234, 'Janos': 3456}

- lekérdezés

    tel['Peter'] == 1234

- módosítás

    tel['Janos'] = 1287

        tel == {'Peter': 1234, 'Janos': 1287}

    tel['Jozsef'] = 6543

        tel == {'Peter': 1234, 'Janos': 3456, 'Jozsef': 6543}

Típusok + műveleteik

Szótár adatszerkezet

tel = {'Peter': 1234, 'Janos': 3456, 'Jozsef': 6543}

- törlés

  del tel['Peter']

  tel == {'Janos': 3456, 'Jozsef': 6543}

- kulcsok lekérdezése

  - tel.keys() == ['Peter', 'Janos', 'Jozsef']
  - tel.has_key('Janos') == True

    tel.has_key('Miklos') == False

Típusok + műveleteik

Szótár adatszerkezet

- létrehozás listából:

```
dict([('a',2),('b',3),('c',42)]) == {'a': 2, 'c': 42, 'b': 3}

dict([('a',2),('b',3),('c',42), ('a',43),('b',12)]) == {'a': 43, 'c': 42, 'b': 12}

dict([(x,x**3) for x in [1,2,3] ]) == {1: 1, 2: 8, 3: 27}
```

Függvénydefiníció

```
def lnko(x, y):
    "Két szám legnagyobb közös osztója."

    while not (x == y) :
        if x > y :
            x = x - y
        else :
            y = y - x

    return x
```

Függvénydefiníció

```
def lnko(x, y):
    "Két szám legnagyobb közös osztója."

    ...
```

Megjegyzés: dokumentáció

lnko.__doc__ == "Két szám legnagyobb közös osztója."

Default paraméter értékek

```
def lnko(x=10, y=15):
    "Két szám legnagyobb közös osztója."

    while not (x == y) :
        if x > y :
            x = x - y
        else :
            y = y - x

    return x
```

Függvény hívás, függvény objektum

```
x = lnko()

x = lnko(122)

x = lnko(324, 248)

x = lnko(y=120)

x = lnko(y=120, x=22)
```

```
myfun = lnko

y = myfun(128, 24)
```

Paraméterátadás
objektum referencia szerint
(immutable vs. mutable)

```
def myDouble(x):
    x = 2*x
    return x

y = 125
z = myDoble(y)
```

```
def mySort(l):
    l.sort()
    return l

ly = [3, 4, 2]
lz = mySort(ly)
```

Eredmény:

y == 125
z == 250

ly == [2, 3, 4]
lz == [2, 3, 4]

Default paraméter értékek

(immutable vs. mutable)

```
x = 10

def myFun1(arg=x) :
    return arg


x = 20

    myFun1() == 10
```

```
lista = [10, 20]

def myFun2(arg=lista) :
    return arg


lista[1] = 30

    myFun2() == [10, 30]
```

Láthatóság

```
x = 20
y = 30

def myFun() :
    global x

    x = 10     # global
    y = 10     # local


myFun()
print(x, y) # 10  30
```

Láthatóság

```
x = 20
y = 30

def myFun() :
    y = x        # y local, x global
    print(y)     # 20

myFun()
print(x, y) # 20  30
```

Láthatóság

```
x = 20
y = 30

def myFun() :
    y = x        # y local, x ???  --- ERROR  x local and unbound!!!
    print(y)
    x = 10     # x local

myFun()  --- ERROR
```

Speciális függvénydefiníció

```
def fv(x, y="egy", *args, **keyargs) :
    print ("x:", x)
    print ("y:", y)
    for arg in args: print (arg);
    for kw in keyargs.keys(): print (kw, ":", keyargs[kw]);
```

Jó hívások:

fv(5, 12, 3, z=24, h=21),

fv(5, z=24, h=21)

Rossz hívás:

fv(5, 12, z=3, 4)