Tartalomjegyzék

Visszapillantó

- = vs == értékadás vs 'ellen?rzés'
- függvény vs metódus
- változó vs 'szimbólum'

Egyszer? adattípusok

A Sage (és kb. egyben a Python) beépített egyszer? típusai:

Típus	Leírás	Példa
None	"http://wiki.math.bme.husemmi"http://wiki.math.bme.hu, null típus	None
int	egész szám (32-bites)	6
long	hosszú egész (tetsz?legesen hosszú)	2354099L
float	lebeg?pontos szám (törtek)	3.75
complex	komplex szám	3-2*i

Összetett adattípusok

A Sage (és kb. egyben a Python) beépített egyszer? típusai:

Típus	Leírás	Példa
str	karakterlánc (string)	'alma'
list	lista	[2,y,'bb']
tuple	tuple	(2,x,'aa')
set	halmaz	set(['a','c'])
dict	szótár (kulcs:érték)	dict({'one':1,'two':2})

Listák

- Listát de?niálhatunk úgy, hogy megadjuk az elemeit (akármilyen típusúak lehetnek) szögletes zárójelek között:

```
sage: L1 = [pi, 'abc', 35, pi, 12]
```

- Hivatkozhatunk a lista egy elemére:

```
sage: L1[2]
35
```

Figyeljünk arra, hogy 0-tól kezd?dik az elemek számozása!

- Kiválaszthatunk egy rész-listát:

```
sage: L1[1:4]
['abc', 35, pi]
```

- Létrehozhatunk egy listát a range függvény segítségével:

```
sage: L2 = range(5)
sage: L2
[0, 1, 2, 3, 4]
```

- Lekérdezhetjük a lista hosszát (hány eleme van):

```
sage: len(L2)
5
```

- A listának bármely elemét felülírhatjuk:

```
sage: L2[1] = 'egy'
sage: L2
[0, 'egy', 2, 3, 4]
```

A tuple típus

A tuple sokmindenben hasonlít a listához, de az elemei utólag nem módosíthatóak.

- Kerek zárójellel de?niáljuk:

```
sage: T = (pi, 'abc', 35, pi, 12)
```

- Hivatkozhatunk a tuple egy elemére:

```
sage: T[2]
```

```
35
```

- Ha meg akarnánk változtatni a tuple egy elemét:

```
sage: T[1]='s'
TypeError: 'tuple' object does not support item assignment
```

A halmaz típus

A `set` típus megfelel a matematikai halmaz fogalomnak: nem rendezett elemek gyüjteménye.

- De?niálható a `set` kulcsszóval és egy lista megadásával:

```
sage: S = set([pi, 'abc', 35, pi, 12])
```

- Megkérdezhetjük, hogy valami benne van-e a halmazban:

```
sage: 35 in S
True
```

- Lekérdezhetjük a halmaz méretét (hány eleme van):

```
sage: len(S)
4
```

- Törlés a halmazból:

```
sage: S.remove(pi)
```

A szótár típus

A szótár arra való, hogy kulcs-érték párokat egymáshoz rendelhessünk.

- Kapcsos zárójellel definiálható a következ? módon:

```
sage: D = {'one':1, 'two':2, 'three':3}
```

- Vagy a `dict` kulcsszó használatával:

```
sage: D = dict({'one':1, 'two':2, 'three':3})
```

- Kulcshoz tartozó érték lekérdezése:

```
sage: D['one']
1
```

- Lekérdezhetjük a szótár kulcsait:

```
sage: D.keys()
['three', 'two', 'one']
```

- Megkérdezhetjük, hogy valami benne van-e a szótárban:

```
sage: 'two' in D
True
```

- Az értékek között is kereshetünk a `values` metódussal:

```
sage: 2 in D.values()
True
```

Még a típus-okról

Egy változó típusát a `type` függvénnyel nézhetjük meg.
Például (a változók az órán már korábban szerepelt változók):

```
sage: type(L1)
<type 'list'>
sage: type(D)
<type 'dict'>
```

Vagy adat esetén:

```
sage: type(5)
<type 'sage.rings.integer.Integer'>
sage: type(5.0)
<type 'sage.rings.real_mpfr.RealLiteral'>
sage: type('5')
<type 'str'>
```

Algoritmusok

Algoritmuson vagy eljáráson olyan megengedett lépésekb?l álló módszert, utasítás(sorozato)t, részletes útmutatást, receptet értünk, amely valamely felmerült probléma megoldasára alkalmas.
Az algoritmust leírhatjuk pszeudokóddal vagy folyamatábrával, illetve implementálhatjuk (megvalósíthatjuk) egy konkrét programnyelven, hogy m?köd? programot kapjunk.

Algoritmus példa

Válasszuk ki az a; b; c számok közül a legkisebbet!

Flowchart

```
IF a < b:
    IF a < c:
        RETURN a
    ELSE:
        RETURN c
ELSE:
    IF b < c:
        RETURN b
    ELSE:
        RETURN c
```

Folyamatábra (Flowchart) Pszeudokód (Pseudocode)

Feltételes utasítás

Egy feltételes utasítás így néz ki Sage-ben, ill. Python-ban:

```
def k(x):
  if x%10 == 0:
    return '0-ra végz?dik'
  elif x%5 == 0:
    return '5-re végz?dik'
  elif x%2 == 0:
    return 'Nem 0-ra végz?d? páros'
  else:
    return 'Nem 5-re végz?d? páratlan'
```

Flowchart

While ciklus

A while-ciklus belseje addig fog ciklikusan újra és újra végrehajtódni, amíg a feltétel része teljesül.
Sage-ben:

```
x = 30
while x > 15:
  x = x-3
  print x
```

Flowchart

Figyelni kell, hogy ne kerüljön a program végtelen ciklusba!

For ciklus

A `for` ciklus valamilyen lista vagy szekvencia elemein megy végig egyesével.

```
for x in range(4):
    print str(x) + '. elemnél járunk'
```

A fenti kód kimenete:

```
0. elemnél járunk
1. elemnél járunk
2. elemnél járunk
3. elemnél járunk
```

További olvasnivalók

Hasznos olvasnivalók, néhol a tananyagon túl:

- Programozás bevezet? (angolul):
  http://johnstachurski.net/book/sample2.pdf
- Sage Tutorial:
  http://www.sagemath.org/pdf/SageTutorial.pdf
- Algoritmus (Wikipedia oldal):
  http://hu.wikipedia.org/wiki/Algoritmus
- Folyamatábra (angol Wikipedia oldal):
  http://en.wikipedia.org/wiki/Flow_chart