Tartalomjegyzék

Irányított gráfok Sage-ben

BST

```
sage: g = DiGraph({8:[3,10], 3:[1,6], 6:[4,7], 10:[14], 14:[13]})
sage: g
  Digraph on 9 vertices
```

Lokális és globális változók

A lokális változók csak a függvényen belül használhatók, "http://wiki.math.bme.hukívül"http://wiki.math.bme.hu-r?l, közvetlenül azok nem érhet?k el. A globális változók az alkalmazás "http://wiki.math.bme.huf?"http://wiki.math.bme.hu részében léteznek

```
sage: def concatenate(L1,L2):
    for x in L2:
        L1.append(x)
    return L1

sage: L1 = [1,3,5,7]
sage: L2 = [2,4,6,8]
sage: concatenate(L1,L2)
[1, 3, 5, 7, 2, 4, 6, 8]
```

Gyorsrendez? algoritmus

```
sage: def quicksort(T):
     less=[]
     equal=[]
     greater=[]
     if len(T) <= 1:
         return T
     middle = T[floor(len(T)/2)]
     for x in T:
         if x < middle:
             less.append(x)
         if x == middle:
             equal.append(x)
         if x > middle:
             greater.append(x)
     retList = quicksort(less)
     concatenate(retList,equal)
```

```
    concatenate(retList,quicksort(greater))
    return retList
```

Python

- Guido van Rossum
- Monty Python
- magas szint? programozási nyelv
- interpreteres nyelv
- objektumorientált
- free and open source
- 2000. okt. 16 Python 2.0

Interaktív mód

```
oktato:~$ python
Python 2.7.3 (default, Aug  1 2012, 05:14:39)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 3
5
>>> 5/3
1
>>> 5.0/3
1.6666666666666667
>>> 5.0//3
1.0
>>>
```

```
>>> sqrt(16)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sqrt' is not defined
>>> 16^2
18
>>> 16**(1/2)
1
>>> 16**(1.0/2)
4.0
>>>
```

Változatok gyök1-témára

Valósítsuk meg a gyökfüggvényt!

```
>>> def sqrt(x):
...     y = x**(1.0/2)
...     return y
...
>>> sqrt(16)
4.0
>>>
```

Rengeteg ún. module létezik a Python-hoz. Ezek általában feladat orientáltak. Ilyen pl. a `math` module is. Ahhoz, hogy egy module függvényeit használni tudjuk, el?ször importálnunk kell azt.

```
>>> import math
>>> math.sqrt(16)
```

```
4.0
>>>

>>> y = 16
>>> sqrt(y)
4.0
>>> y
16
>>>
```

Python scriptek

A Python script file kiterjesztése (hagyományosan, de nem kötelez?en) `.py`
A fájl els? sora: `#!/usr/bin/python` vagy `#!/usr/bin/env python`
A foo.py fájl futtathatóvá tétele: `chmod a+x foo.py`
Futtatása: `./foo.py`

A `gyok1.py` file tartalma:

```
#!/usr/bin/python

def sqrt(x):
     y = x**(1.0/2)
     return y

y = 16
sqrt(y)
```

Futtatása és eredménye:

```
oktato:~$ ./gyok1.py
oktato:~$
```

A `gyok2.py` a második verzió... (return?)

```
#!/usr/bin/python

def sqrt(x):
     y = x**(1.0/2)
     return y

y = 16
return sqrt(y)
```

Futtatása és eredménye:

```
oktato:~$ ./gyok2.py
  File "./gyok2.py", line 8
    return sqrt(y)
SyntaxError: 'return' outside function
oktato:~$
```

A `gyok3.py` az eredmény kiíratása

```
#!/usr/bin/python
```

```
def sqrt(x):
     y = x**(1.0/2)
     return y

y = 16
print sqrt(y)
```

Futtatása és eredménye:

```
oktato:~$ ./gyok3.py
4.0
oktato:~$
```

A `gyok4.py` szöveg kiíratása.
Karakterkódolás kell: #coding: utf-8

```
#!/usr/bin/python
#coding: utf-8

def sqrt(x):
     y = x**(1.0/2)
     return y

y = 16
print '16 négyzetgyöke:'
print sqrt(y)
```

Futtatása és eredménye:

```
oktato:~$ ./gyok4.py
16 négyzetgyöke:
4.0
oktato:~$
```

`gyok5.py` Interakció: kérdezzük meg a felhasználót!
A megvalósításhoz a `raw_input()` függvényt használjuk.

```
#!/usr/bin/python
#coding: utf-8

def sqrt(x):
     y = x**(1.0/2)
     return y

y = raw_input("Adj meg egy egész számot!\n")
print y,' négyzetgyöke:\n',sqrt(float(y))
```

Futtatása és eredménye:

```
oktato:~$ ./gyok5.py
Adj meg egy egész számot!
25
25  négyzetgyöke:
5.0
oktato:~$
```

A beolvasott adat string tipusú, float-tá kellett alakítani!

Python scriptek II.

Argumentumok használata

Az `arg1.py`, a kapott argumentumok használatához importálni kell a sys module-t:

```
#!/usr/bin/python
#coding: utf-8

import sys

for arg in sys.argv:
    print arg
```

Futtatása és eredménye:

```
oktato:~$ ./arg1.py 1 argumentum
./arg1.py
1
argumentum
oktato:~$
```

Az argumentumok egy listában tárolódnak.
Az `arg2.py`-nal a 0. és az 1. argumentum kiíratása:

```
#!/usr/bin/python
#coding: utf-8

import sys

print sys.argv[0]
if  len(sys.argv) > 1:
  print sys.argv[1]
```

Futtatása és eredménye:

```
oktato:~$ ./arg2.py 1 argumentum
./arg2.py
1
oktato:~$
```

Az argumentumok listájának 0. eleme maga a script!

Még egyszer gyök, most argumentumként kapott értékkel

```
#!/usr/bin/python
#coding: utf-8

import sys
import math

def gyok(a):
    return math.sqrt(a)

if  len(sys.argv) > 1:
  x = sys.argv[1]
```

```
  print 'x tipusa: ',type(x)
  y = float(x)
  print 'y tipusa: ',type(y)
  a = gyok(y)
else:
  sys.exit(0)

print '\neredmény: ',a
```

Futtatása és eredménye:

```
oktato:~$ ./gyok_arg.py 36
x tipusa:  <type 'str'>
y tipusa:  <type 'float'>

eredmény:  6.0
```

Normál kilépés hibaüzenet nélkül: sys.exit(0)