Bevezetés a Python programozási nyelvbe

Szathmáry László
Debreceni Egyetem
Informatikai Kar

6. Gyakorlat (2012. nov. 8.)

- globális változók
- fájlkezelés

globális változók

```
3  PI = 3.14159     # "konstans"
4
5  counter = 0
6
7
8  def f2():
9      global counter
10     counter = 42
11
12
13 def f1():
14     counter = 5
15     print 'f1:', counter
16
17
18 def f0():
19     print 'f0:', counter
20
21
22 def main():
23     print 'PI:', PI
24     f0()
25     f1()
26     f2()
27     print 'main:', counter
28
29 ############################
30
31 if __name__ == "__main__":
32     main()
```

Megállapodás: a „konstansok” nevét csupa nagybetűvel írjuk.

globális változók

Egy globális változó módosítható, de ezt a „global” kulcsszóval külön jelezni kell!

counter itt egy lokális változó a globális counter-t elrejti

A globális változók minden függvényből látszanak. Alapesetben az értékük nem módosítható.

A globális változók használatát (amiket módosítunk is) ne vigyük túlzásba.

„Konstansokat” pedig lehetőleg NE módosítsunk.

olvasás fájlból

```
Elso sor.
Masodik sor.
Harmadik sor.
~
```

szoveg.txt

```
1  >>> f = open('/tmp/szoveg.txt', 'r')
2  >>> for line in f:
3  ...     print line,
4  ...
5  Elso sor.
6  Masodik sor.
7  Harmadik sor.
8  >>> f.close()
```

a beolvasott soroknak része a '\n', ezért a vessző

vagy:

```
for line in f:
    line = line.rstrip('\n')
    print line
```

Megnyitási módok:

r	--	read
w	--	write
a	--	append

ne felejtsük el bezárni a fájlt a végén

```
1  >>> f = open('/tmp/szoveg.txt', 'r')
2  >>> lines = f.readlines()
3  >>> print lines
4  ['Elso sor.\n', 'Masodik sor.\n', 'Harmadik sor.\n']
```

Az egész file-t beolvassa, s a file sorait egy listában adja vissza.
A '\n' most is ott van a sorok végén.

```
14  >>> f = open('/tmp/szoveg.txt', 'r')
15  >>> text = f.read()
16  >>> text
17  'Elso sor.\nMasodik sor.\nHarmadik sor.\n'
```

Az egész file-t beolvassa, s a file tartalmát egy sztringben adja vissza.

Megjegyzés: nagy file-ok esetén ez a két módszer igencsak memóriaigényes lehet.

írás fájlba

```
Az egyik modszer.
A masik modszer.
~
```

```
1  >>> f = open('/tmp/teszt.txt', 'w')
2  >>> f.write('Az egyik modszer.\n')
3  >>> print >>f, 'A masik modszer.'
4  >>> f.close()
5  >>>
6  >>> import sys
7  >>>
8  >>> print >>sys.stderr, 'Jaj, reaktorszivargas lepett fel!'
9  Jaj, reaktorszivargas lepett fel!
```

A standard hibakimenetre is ezzel a 2. módszerrel tudunk írni.

Lásd még: Q függelék (olvasás bináris fájlból).

Megjegyzés: nukleáris létesítményt ne Python-ban programozzunk.
De ne is Java-ban! :)

Régebbi módszer:

```
 5  def main():
 6      f = open(INPUT, 'r')
 7      #
 8      # Munka a file tartalmával.
 9      # DE! Ha kivétel lép fel,
10      # akkor a file nem lesz
11      # rendesen bezárva!
12      #
13      f.close()
```

Modern módszer:

```
 6  def main():
 7      with open(INPUT, 'r') as f:
 8          # Munka a file tartalmával.
 9          # Még ha valami kivétel is
10          # lép fel, a file rendesen be
11          # lesz zárva. A "with" blokk
12          # ezt garantálja.
13          print f.read()
```

Itt nem is kell explicit módon meghívni az f.close() függvényt.

Példa: másolat készítése egy szöveges állományról

Python 2.7-től

```
 7  def main():
 8      with open(INPUT, 'r') as f1, open(OUTPUT, 'w') as to:
 9          for line in f1:
10              to.write(line)
```

Feladat: írjuk át ezt a példát a régebbi módszert használva.

Feladat

A `string1.py` file-ból távolítsuk el a megjegyzéseket. Az egyszerűség kedvéért csak azokat a sorokat töröljük, amelyek `#` jellel kezdődnek. A kimenetet egy `string1_clean.py` nevű file-ba írjuk ki.

Link: http://www.inf.unideb.hu/~szathml/pmwiki/index.php?n=Py.20121006d