Bevezetés a Python programozási nyelvbe

Szathmáry László
Debreceni Egyetem
Informatikai Kar

6. Gyakorlat
- globális változók
- fájlkezelés

(utolsó módosítás: 2019. febr. 5.)

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

```
>>> f = open("szoveg.txt", "r")
>>> for line in f:
...     print(line, end="")
...
...
Elso sor.
Masodik sor.
Harmadik sor.
>>>
>>> f.close()
```

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

szoveg.txt

a beolvasott soroknak része a '\n'

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.

Kérdés: nagy file-ok esetén melyik módszert érdemes használni?

```
>>> f = open("szoveg.txt", "r")
>>> lines = f.read().splitlines()
>>> f.close()
>>>
>>> lines
['Elso sor.', 'Masodik sor.', 'Harmadik sor.']
```

Külön sorokra való egyszerű szétbontás.

```
>>> html
'<html>\n<body>\n...\n</body>\n</html>'
>>> print(html)
<html>
<body>
...
</body>
</html>
>>> html.splitlines()
['<html>', '<body>', '...', '</body>', '</html>']
>>> html.split("\n")
['<html>', '<body>', '...', '</body>', '</html>']
```

írás fájlba

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

```
>>> f = open("teszt.txt", "w")
>>> f.write("Az egyik modszer.\n")
18
>>> print("A masik modszer.", file=f)
>>> f.close()
>>>
>>> import sys
>>>
>>> print("Jaj, reaktorszivargas lepett fel!", file=sys.stderr)
Jaj, reaktorszivargas lepett fel!
>>>
```

írás a standard hibakimenetre

Lásd még: N függelék (olvasás bináris fájlból).

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: https://arato.inf.unideb.hu/szathmary.laszlo/pmwiki/index.php?n=Py3.20121006d

Feladatok #1

házi feladat

1. [20121006d] megjegyzések eltávolítása
2. [20120818h] 100 db 50-jegyű szám (PE #13) [**B** változat]
3. [20130218c] Karakterszámláló
4. [20120818g] Öt egymást követő számjegy legnagyobb szorzata (PE #8)
5. [20130211a] Anagramma
6. [20130919b] a-z; megfordítva: z-a (Jaj! Gondolkodós feladat!)
7. [20130902e] Zárójelek
8. [20130902b] Hamming-távolság

Feladatok #2

házi feladat

1. [20120815g] PI vers (*list comprehension*-nel)
2. [20120818i] számjegyek összege (PE #16)
3. [20130902c] mondat extra szóközök nélkül

Jegyzet

```
>>> f = open("szimbolum_tabla.txt", "r")
>>> f.read()
Traceback (most recent call last):
  File "<input>", line 1, in <module>
    f.read()
  File "/usr/lib/python3.5/codecs.py", line 321, in decode
    (result, consumed) = self._buffer_decode(data, self.errors, final)
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xdb in position 1: invalid continuation byte
>>>
================================================================================
[12:02:22] /tmp $ file -i szimbolum_tabla.txt
szimbolum_tabla.txt: text/plain; charset=iso-8859-1
================================================================================
[12:02:29] /tmp $ bpython
bpython version 0.16 on top of Python 3.5.2 /usr/bin/python
>>>
>>> f = open("szimbolum_tabla.txt", "r", encoding="iso-8859-1")
>>> f.read()
'FÛBETON BURKOLAT\nKILOMÉTERKÕ\nTERELÕNYÍL (1 ÁGÚ)\nTERELÕNYÍL (2 ÁGÚ)\nTERELÕNYÍL (3 ÁGÚ)\n'
>>>
```

Az alapértelmezett karakterkódolás platformfüggő (a locale beállítástól függ). Itt a példában „utf-8” kódolással próbáltuk megnyitni a file-t, de mint kiderült, a file „iso-8859-1” kódolású. Az open() fv.-nek van egy *encoding* nevű opcionális paramétere, ahol pontosan meg tudjuk mondani, hogy a file milyen karakterkódolású. Mint látható, ekkor már rendesen sikerült beolvasni a tartalmát.