



Bevezetés a Python programozási nyelvbe

Szathmáry László
Debreceni Egyetem
Informatikai Kar

Függelékek

Utolsó módosítás: 2012. nov. 9.

2012-2013, 1. félév



A) Függelék

A Python telepítése

Telepítés Linux alá

A mai Linux disztribúciók alapból tartalmazzák a Python interpretert (ilyen a gyakorlaton használt Ubuntu GNU/Linux is). Az interaktív shellt parancssorból a „python” paranccsal tudjuk elindítani:

```
$ python
Python 2.7.3 (default, Apr 20 2012, 22:39:59)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Megjegyzés: elképzelhető, hogy a Python 3 interpreter is fent van a gépünkön. Ezt a „python3” paranccsal tudjuk leellenőrizni. A két interpreter egymás mellett is fent lehet, nem akadnak össze.

Telepítés Windows alá

- Látogassuk meg a <http://www.python.org/download/> helyet s töltsük le a „Windows Installer” verziót, majd telepítsük a C: meghajtó gyökérkönyvtárába (C:\Python27).
- A Start menüből el tudunk indítani egy egyszerű grafikus shellt (IDLE a neve), vagy elindíthatjuk a parancssoros shellt is. A parancssoros shellből az `exit()` segítségével tudunk kilépni.
- A telepítő hozzárendelte a .py kiterjesztésű fájlokhoz a Python interpretet, vagyis egy Python szkriptet ezután úgy is el tudunk indítani, hogy duplán rákattintunk.
- A C:\Python27 könyvtárat érdemes beletenni a PATH-ba, így parancssorból bárhonnán meg tudjuk hívni a Python shellt.

B) Függelék

Interaktív shellek

- Az alapértelmezett shellt a „**python**” paranccsal tudjuk elindítani.
- Az IPython a hagyományos python shell lehetőségeit terjeszti ki: szintaxis kiemelés, TAB-bal történő kiegészítés, stb.

Telepítése: `sudo apt-get install ipython .`

Oktatóvideó: <http://www.youtube.com/watch?v=2G5YTlheCbw>.

- A bpython egy másik népszerű kiterjesztése az alap shellnek. Szintén tud szintaxis kiemelést, illetve gépelés közben javaslatokat tesz a kód kiegészítésére. A javaslatok között TAB-bal tudunk váltani.

Telepítése: `sudo apt-get install bpython .`

C) Függelék

Szövegszerkesztők, integrált fejlesztői környezetek (IDE-k)

Szövegszerkesztők

A Python interpreteren és egy szövegszerkesztőn kívül tulajdonképpen nincs is másra szükségünk...

- GEdit (Linux), Notepad++ (Windows) [kezdő]
- Sublime Text 2 [középhaladó]
(<http://pythonadventures.wordpress.com/2012/08/09/sublime-text-2/>)
- Vim extrákkal: Turning Vim into a modern Python IDE [haladó]
(<http://sontek.net/blog/detail/turning-vim-into-a-modern-python-ide>)

A szövegszerkesztőt / IDE-t úgy állítsuk be, hogy a TAB leütésére 4 db szóközt szúrjon be.

Integrált fejlesztői környezetek (IDE-k)

Ha valamilyen IDE-t szeretnénk használni, akkor számos lehetőség közül választhatunk:

- Eclipse + PyDev
(telepítés: <http://pythonadventures.wordpress.com/2011/05/02/eclipse-with-pydev/>)
- Spyder (<http://code.google.com/p/spyderlib/>)
- Ninja-IDE (<http://ninja-ide.org/>)
- Vannak fizetős IDE-k is: PyCharm ([link](#)), WingIDE ([link](#)), stb.

Egy részletesebb listát itt találhatunk:

<http://pythonadventures.wordpress.com/2011/04/04/what-ide-to-use-for-python/>

D) Függelék

Ékezetek használata a forráskódban

```
1  #!/usr/bin/env python
2
3
4  def main():
5      print "ékezetes szöveg"
6
7  if __name__ == "__main__":
8      main()
```



Alapesetben csak ASCII karaktereket használhatunk.

File "ekezet.py", line 5

SyntaxError: Non-ASCII character '\xc3' in file ekezet.py on line 5, but no encoding declared;
see <http://www.python.org/peps/pep-0263.html> for details

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3
4
5  def main():
6      print "ékezetes szöveg"
7
8  if __name__ == "__main__":
9      main()
```



A forrás kódolását egy speciális kommentben kell megadni.

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4
5  def main():
6      print 'árvíztűrő tükörfúrógép'
7
8  #####
9
10 if __name__ == "__main__":
11     main()
```

```
1  #!/usr/bin/env python
2  # encoding: utf-8
3
4
5  def main():
6      print 'árvíztűrő tükörfúrógép'
7
8  #####
9
10 if __name__ == "__main__":
11     main()
```

egyszerűsített
jelölés

Lényeg: ez a speciális komment
az 1. vagy a 2. sorban szerepeljen.

Gyakori hiba Unicode sztringeknél

```
7 def main():  
8     s = u'Voil\u00E0'  
9     print str(s)
```



UnicodeEncodeError: 'ascii' codec can't encode character u'\xe0'
in position 4: ordinal not in range(128)

Trükk (Python 2.x esetén):

```
3 import sys  
4 reload(sys)  
5 sys.setdefaultencoding("utf-8")  
6  
7  
8 def main():  
9     s = u'Voil\u00E0'  
10    print str(s)
```

E) Függelék

Néhány meglepetés („easter eggs”)

Próbáljuk ki a következőket:

```
>>> import antigravity
...
>>> import this
...
>>> import __hello__
...
>>> from __future__ import braces
...
```

a Python filozófiáját
fogja kiírni

F) Függelék

Stílus (PEP8)

Figyeljünk oda a forráskódjaink stílusára is. Ha később elővesszük a programunkat, szeretnénk benne könnyen eligazodni. Illetve lehet, hogy a projektünket valaki más fogja folytatni, gondoljunk őrre is.

A Python forráskódok stílusbeli ajánlásait a [PEP8](#) nevű dokumentumban gyűjtötték össze. Ezeket betartva könnyen olvasható programokat tudunk írni, amikre „öröm lesz ránézni”. Néhány szempont:

- A TAB használatát kerüljük, helyette 4 szóközt használjunk.
- A sorok ne legyenek hosszabbak 79 karakternél.
- A függvényeket és osztályokat, illetve a függvényeken belüli nagyobb blokkokat üres sorokkal válasszuk el egymástól.
- Használjunk docstring-eket.
- Az operátorok köré és a vesszők után tegyünk ki egy szóközt. Az aktuális és formális paraméterlistán viszont a nevesített paraméterek esetén az '=' jel köré nem kell szóköz.
- Az osztályok neve `IlyenLegyen`. A függvények és változók neve `pedig_ilyen`. Az osztályokon belül a függvények első paraméterének neve `self` legyen.
- Ha a kódunkat nemzetközi környezetben fogják használni, akkor ne használjunk semmiféle különleges karaktert, maradjunk a sima ASCII kódolásnál.

G) Függelék

Operátorok

Összehasonlítások összefűzése:

```
4 >>> x = 10
5 >>> 0 < x < 20
6 True
```

Ternáris operátor:

```
8 >>> x = -5
9 >>> print 'negativ' if x < 0 else 'pozitiv'
10 negativ
11 >>> x = 3
12 >>> print 'negativ' if x < 0 else 'pozitiv'
13 pozitiv
```



```
15 >>> x = -5
16 >>> if x < 0: print 'negativ'
17 ... else: print 'pozitiv'
18 ...
19 negativ
```

not:

```
21 >>> li = [1, 2, 3]
22 >>> 2 in li
23 True
24 >>> 5 not in li
25 True
26 >>> 2 not in li
27 False
28 >>> not (5 in li)
29 True
```

ugyanaz

H) Függelék

Sztringek formázása

Első lehetőség:

```
"the {0} is {1}".format('sky', 'blue')
```

Második lehetőség (Python 2.7+ -től):

```
"the {} is {}".format('sky', 'blue')
```

új módszer,
inkább ezeket
használjuk

Harmadik lehetőség:

```
"the {what} is {color}".format(what='sky', color='blue')
```

Régi módszer:

```
"the %s is %s" % ('sky', 'blue')
```

régi módszer
(még támogatott, de inkább
kerüljük a használatát)

opcionális kettőspont után:
format specifier

```
4 >>> pi = 3.14159
5 >>> print 'pi erteke: {0:.2f}'.format(pi)
6 pi erteke: 3.14
7 >>> print 'pi erteke: %.2f' % pi
8 pi erteke: 3.14
```

régi formázási módszer,
helyette a `format()` -ot használjuk

További példák:

http://knowledgestockpile.blogspot.com/2011/01/string-formatting-in-python_09.html

```
1 >>> for x in range(1, 10+1):
2 ...     print '{0:2d} {1:3d} {2:4d}'.format(x, x**2, x**3)
3 ...
4 1      1      1
5 2      4      8
6 3      9     27
7 4     16     64
8 5     25    125
9 6     36    216
10 7     49    343
11 8     64    512
12 9     81    729
13 10    100   1000
```

```
1 >>> 'Laci'.center(20)
2 '          Laci          '
3 >>> 'Laci'.ljust(20)
4 'Laci                    '
5 >>> 'Laci'.rjust(20)
6 '                    Laci'
7 >>>
8 >>> '12'.zfill(5)
9 '00012'
```

← adott hosszon balra igazít,
a maradék helyet szóközzel
tölti ki

J) Függelék

Írás a standard kimenetre

```
1 >>> a = range(5)
2 >>> a
3 [0, 1, 2, 3, 4]
4 >>> for e in a:
5 ...     print e
6 ...
7 0
8 1
9 2
10 3
11 4
12 >>> for e in a:
13 ...     print e,
14 ...
15 0 1 2 3 4
16 >>>
17 >>> import sys
18 >>>
19 >>> for e in a:
20 ...     sys.stdout.write(e)
21 ...
22 Traceback (most recent call last):
23   File "<stdin>", line 2, in <module>
24 TypeError: expected a character buffer object
25 >>>
26 >>> for e in a:
27 ...     sys.stdout.write(str(e))
28 ...
29 01234>>>
```

1

(„\n”)

2

(szóköz)

3

(„full control”)

K) Függelék

Szekvencia bejárása fordított sorrendben

```
1  >>> li
2  [1, 3, 4, 6, 8, 9]
3  >>> li[::-1]
4  [9, 8, 6, 4, 3, 1]
5  >>> for e in li[::-1]:
6  ...     print e,
7  ...
8  9 8 6 4 3 1
9  >>>
10 >>> reversed(li)
11 <listreverseiterator object at 0x240bd10>
12 >>> li
13 [1, 3, 4, 6, 8, 9]
14 >>> for e in reversed(li):
15 ...     print e,
16 ...
17 9 8 6 4 3 1
```

új listát ad vissza

Nem ad vissza új listát.
Generátor, vagyis az
elemeket egyenként
adja vissza.

Ciklusban használatos.

Ha nagyon nagy méretű tömbökkel dolgozunk, akkor inkább
a `reversed()` beépített fv.-t használjuk.

L) Függelék

Random számok



Átkerült a gyakorlat anyagába...

M) Függelék

Beépített függvények

		Built-in Functions		
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	apply()
delattr()	help()	next()	setattr()	buffer()
dict()	hex()	object()	slice()	coerce()
dir()	id()	oct()	sorted()	intern()

<http://docs.python.org/library/functions.html>

Ezek a függvények bármikor elérhetőek, nem kell a használatukhoz külön modult importálni.

N) Függetlenség

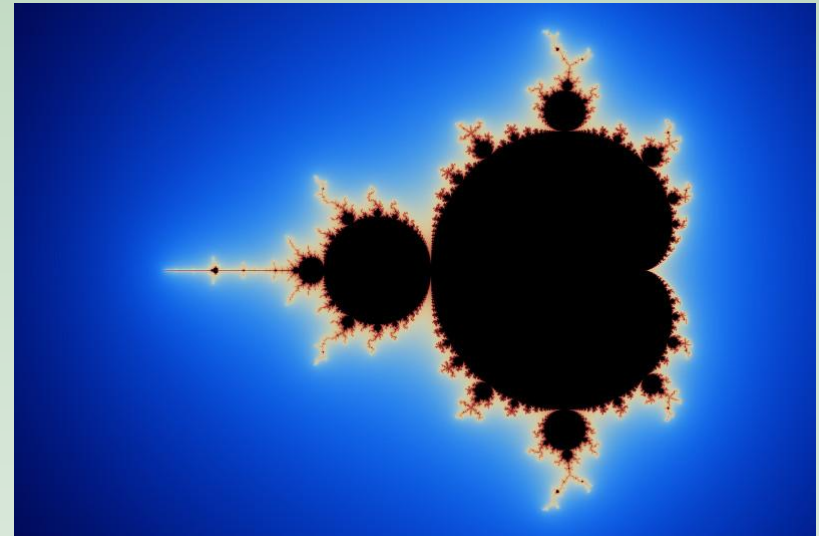
Obfuszkált Python

Ha egy Perl-es ismerősünk azzal jön, hogy „a Python azért nem jó, mert csak olvasható kódot lehet benne írni” :), akkor bátran mutassuk meg neki a következő kódokat:

```

-                                     = (
                                     255,
                                     lambda
                                     V, B, c
                                     :c and Y(V*V+B,B, c
                                     -1) if (abs(V)<6) else
(                                     2+c-4*abs(V)**-0.4)/i
) ;v, x=1500,1000;C=range(v*x
);import struct;P=struct.pack;M,\
j ='<QIIHHHH',open('M.bmp','wb').write
for X in j('BM'+P(M,v*x*3+26,26,12,v,x,1,24))or C:
i ,Y=_;j(P('BBB',*(lambda T:(T*80+T**9
*i-950*T **99,T*70-880*T**18+701*
T **9 ,T*i**(1-T**45*2)))(sum(
[ Y(0,(A%3/3.+X%v+(X/v+
A/3/3.-x/2)/1j)*2.5
/x -2.7,i)**2 for \
A in C
[:9]])
/9)
) )

```

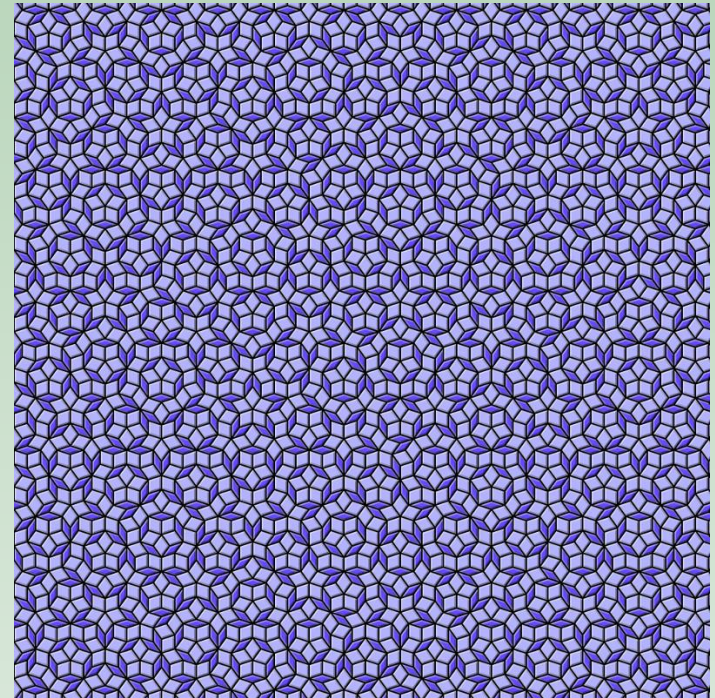


<http://preshing.com/20110926/high-resolution-mandelbrot-in-obfuscated-python>

```

-
    =\
    ""if!
    1:"e,V=100
    0,(0j-1)**-.2;
    v,S=.5/ V.real,
    [(0,0,4      *e,4*e*
    V)];w=1      -v"def!
    E(T,A,      B,C):P
    ,Q,R=B*w+      A*v,B*w+C
    *v,A*w+B*v;retur      n[(1,Q,C,A),(1,P
    ,Q,B),(0,Q,P,A)]*T+[(0,C      ,R,B),(1,R,C,A)]*(1-T)"f
or!i!in!_[:11]:S      =sum([E      (*x)for      !x!in!S],[])"imp
ort!cair      o!as!O;      s=0.Ima      geSurfac
e(l,e,e)      ;c=O.Con text(s);      M,L,G=c.
move_to      ,c.line_to,c.s      et_sour
ce_rgb      a"def!z(f,a)      :f(-a.
imag,a.      real-e-e)"for!T,A,B,C!in[i      !for!i!
in!S!if!i[""";exec(reduce(lambda x,i:x.replace(chr
(i),"\\n "[34-i:]),      range(      35),_+""0)]:z(M,A
);z(L,B);z      (L,C);      c.close_pa
th() "G      (.4,.3      ,1);c.
paint(      );G(.7      ,.7,1)
;c.fil      l() "fo      r!i!in
!range      (9):"!      g=1-i/
8;d=i/      4*g;G(d,d,d,      1-g*.8
)"!def      !y(f,a):z(f,a+(1+2j)*(      1j**(i
/2.)) *g)"!for!T,A,B,C!in!S:y(M,C);y(L,A);y(M
,A);y(L,B)"!c.st      roke() "s.write_t
o_png('pen      rose.png')
""""
))

```



<http://preshing.com/20110822/penrose-tiling-in-obfuscated-python>

O) Függelék

Keleti bölcsesség

"I hear and I forget. I see and I remember. I do and I understand."

Confucius



Vagyis: gyakorolni, gyakorolni és gyakorolni...

P) Függelék

A Python egy újabb eszköz



A Pythonra úgy tekintünk, mint egy újabb eszközre az eszköztárunkban.

Az adott feladathoz a legmegfelelőbb eszközt használjuk.

Q) Függelék

Olvasás bináris fájlból

```
12 def mp3():
13     f = open('Unstoppable.mp3', 'rb')    # megnyitás bináris módban
14     print f.tell()                      # 0, a file elején vagyunk
15     f.seek(-128, 2)                     # lépünk vissza 128 pozíciót a file végétől
16     print f.tell()                      # 3411286 (akt. pozíció a file elejétől)
17     tag_data = f.read(128)              # olvassunk be 128 byte-ot
18     f.close()
```

Az `f.read()` -nek opcionálisan meg lehet adni, hogy hány byte-ot olvasson be.

Az `f.seek()` két paramétert vár:

1. a file-kurzor hány pozíciót mozduljon el
2. az elmozdulás mihez képest történjen:
 - 0: abszolút pozícióba lépjen a file elejétől
 - 1: relatíve mozduljon el az aktuális pozíciótól
 - 2: relatíve mozduljon el a file végétől

R) Függelék

Haladó rendezés

Átkerült a gyakorlat anyagába...

