



Szkriptnyelvek

Szathmáry László
Debreceni Egyetem
Informatikai Kar

Függelékek

(utolsó módosítás: 2025. jan. 17.)

2024-2025, 2. félév



A) Függelék

A Python telepítése

Telepítés Linux alá

A mai Linux disztribúciók alapból tartalmazzák a Python interpretert (ilyen a gyakorlaton használt Linux Mint is). Az interaktív shellt parancssorból a „python3” paranccsal tudjuk elindítani:

```
[14:50:57] ~ $ python3
Python 3.6.4 (default, Jan  5 2018, 02:35:40)
[GCC 7.2.1 20171224] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Megjegyzés: egyes (régebbi) Linux disztribúciókon a „python” parancs hatására a Python 2 interpreter indul el. A Python shell indításakor szemmel mindig ellenőrizzük le, hogy melyik verzió indult el. A legjobb megoldás, ha a „python3” parancsot használjuk.

Telepítés Windows alá

- Látogassuk meg a <https://www.python.org/downloads/windows/> helyet s töltsük le a telepítőt (Windows x86-64 executable installer), majd telepítsük a C: meghajtó gyökérkönyvtárába (C:\Python312).
- A telepítéskor jelöljük be, hogy a telepítő adja hozzá a PATH környezeti változóhoz a szükséges könyvtárakat.
- A parancssoros shellből a `quit()` segítségével tudunk kilépni, vagy a Ctrl+Z (majd Enter) billentyűkombinációval.
- A telepítő hozzárendelte a .py kiterjesztésű fájlokhoz a Python interpretet, vagyis egy Python szkriptet ezután úgy is el tudunk indítani, hogy duplán rákattintunk.

Ajánlott videó a telepítés menetéről: <https://www.youtube.com/watch?v=1HUcq3SPxJ4>

B) Függelék

Interaktív shellek

- Az alapértelmezett shellt a „`python3`” paranccsal tudjuk elindítani.
- Az IPython a hagyományos python shell lehetőségeit terjeszti ki: szintaxis kiemelés, TAB-bal történő kiegészítés, stb.
Telepítése: `pip3 install ipython --user -U`
Oktatóvideó: <http://www.youtube.com/watch?v=2G5YTIheCbw>.
- 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: `pip3 install bpython --user -U`

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]
- Visual Studio Code
(<https://www.youtube.com/playlist?list=PL6ubK4uhF2NUTRadeOkTIIIfGdLW20ffi9>)
- Sublime Text
- Spyder (the Scientific PYthon Development EnviRonment)
- PyCharm IDE (a Community Edition ingyenes)

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

E) Függetlenség

Néhány meglepetés („easter eggs”)

Próbáljuk ki a következőket:

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

a Python filozófiáját fogja kiírni



Ajánlott videó:

- [Néhány rejtett poén \(easter egg\) Python-ban és Java-ban](#)

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')
```

Harmadik lehetőség:

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

Régi módszer:

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

új módszer,
inkább ezeket
használjuk

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

<http://mkaz.com/solog/python-string-format>

```
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

Beépített függvények

<https://docs.python.org/3/library/functions.html>

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

Ajánlott videók:

- [A Python nyelv beépített függvényei \(1.rész\)](#)
- [A Python nyelv beépített függvényei \(2.rész\)](#)

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

M) Függetlenség

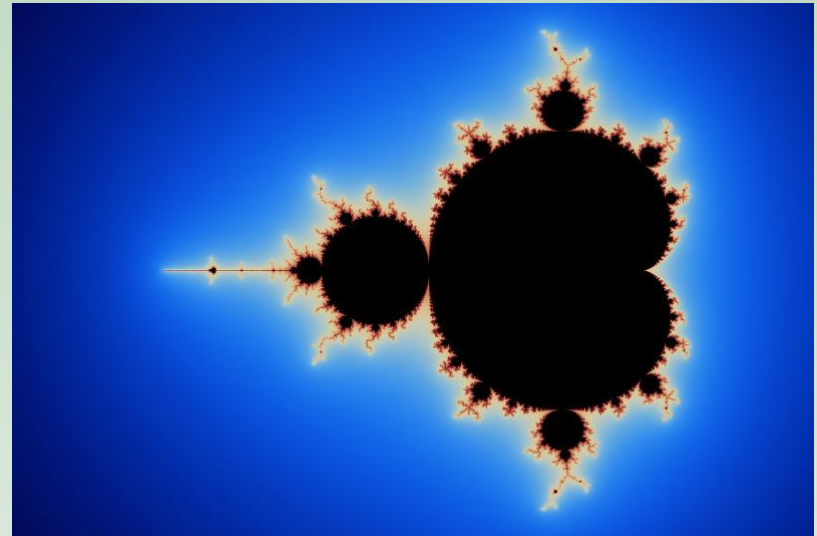
Obfuszkált Python

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)
        ) )

```

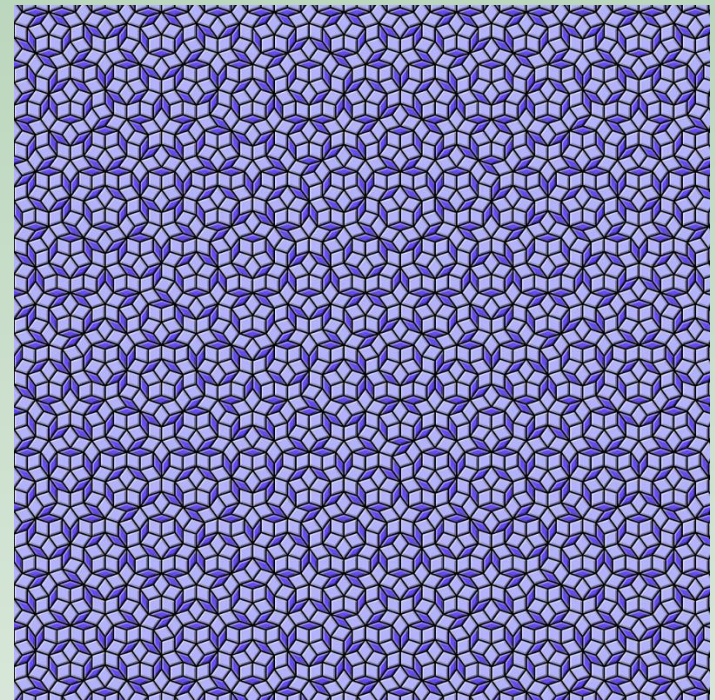


<http://preshing.com/20110926/high-resolution-mandelbrot-in-obfuscated-python>

```

-
                                =\
                                ""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=O.Ima geSurfac
                                e(1,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>

N) 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

Q) Függelék

A Python nyelv születése

Guido 1996-ban a következőket írta a Python nyelv születéséről:

„Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).”

(forrás: [Wikipedia](#))