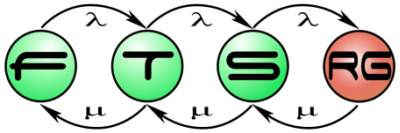
Szkriptelés és Python alapok

Horányi Gergő, Micskei Zoltán, Szatmári Zoltán, Tóth Dániel, Honfi Dávid



Tartalom

Motiváció: szkriptelés

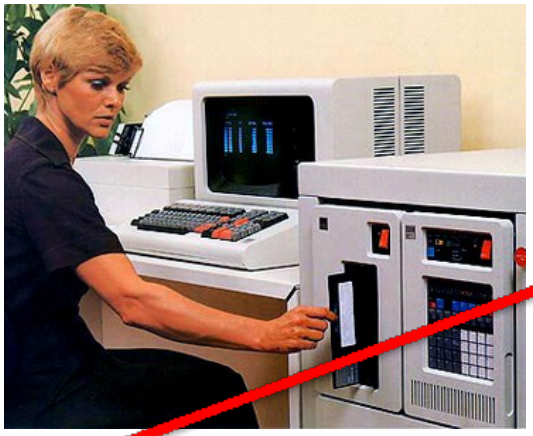
Linux alapok

Python alapok (március 5., szombat)

Windows PowerShell (március 7., hétfő)

Parancssoros felületek

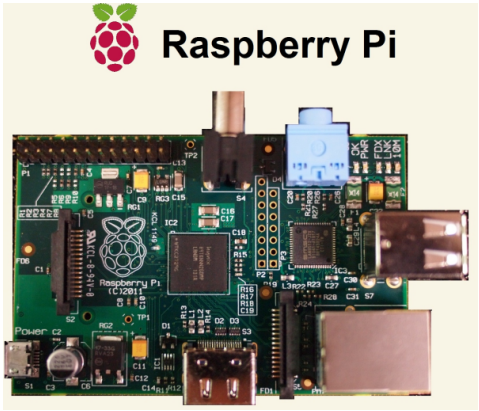
CLI: elavult



forrás: http://www-03.ibm.com/ibm/history

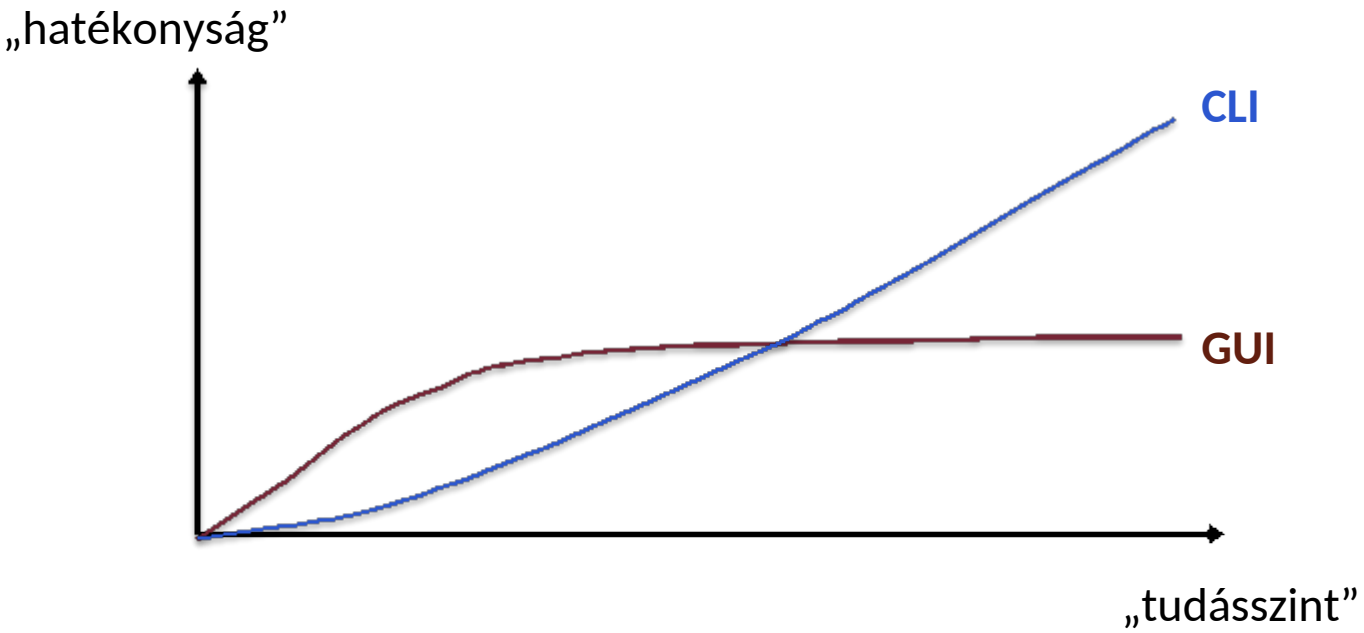
GUI: modern

CLI manapság:



Parancssoros és grafikus felületek

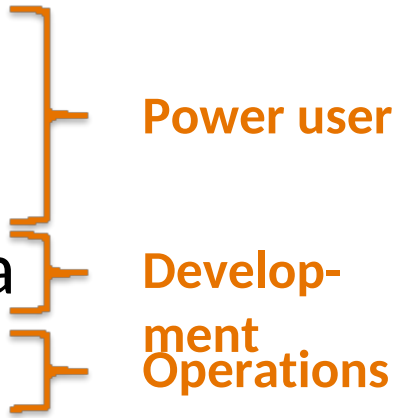
	GUI	CLI
Tanulhatóság	Könnyű	Nehezebb
Automatizálható	Nehezen	Könnyen
Hasznos	Kezdő / alkalmi felhasználó	Szakértő / gyakori használat



Szkriptelés motivációja: automatizálás

Gyakran ismétlődő feladatok

- Fájlok csoportos átnevezése
- MP3 csoportos átkódolás
- Több fejlesztési projekt együttes fordítása
- Felhasználók csoportos felvétele
- …



Java/C# programot is írhatunk rá

- **DE**: biztos ez a leghatékonyabb eszköz?

Motiváció: szkript nyelvek

Nem szükséges speciális fejlesztői környezet

Legtöbb gépen elérhető a futtatókörnyezet

Gyors és hatékony eszköz

Szkript nyelvek jellegzetességei

Interpreter futtatja

Akár soronként is értelmezhető

Minden futási időben értékelődik ki

Sok esetben típustalan

(De nem mindig!)

Tartalom

Motiváció: szkriptelés

Linux alapok

Python alapok (március 5., szombat)

Windows PowerShell (március 7., hétfő)

Linux alapok (ismétlés)

Fontos alapparancsok:

- cat: file tartalom kiírása konzolra
- grep: keresés fájlban reguláris kifejezéssel
- ls: könyvtárak kilistázása („dir”)
- cp: fájlmásolás
- rm: fájltörlés
- chmod: fájl jogosultságának állítása
- … *(lásd még: gyakorlaton)*

Sokféle shell és szkript környezet

- sh, csh, bash…

Bash shell (alapvető funkciók)

Automatikus kiegészítés: **TAB billentyű**

Parancs előzmények tárolása

- **Fel** és **Le** gombokkal navigálás
- **CTRL+R** kombinációval keresés
- history parancs

Terminál gyors bezárása: CTRL+D

Átirányítások

Standard fájlleírók (I/O), minden programnak

- 0 – stdin
- 1 – stdout
- 2 – stderr

Átirányítás

- cat fájlnév #fájl → stdout
- cat fájlnév 2>&1 #stderr → stdout
- cat fájlnév > másikfájl #fájl → stdout → másikfájl
- cat fájlnév >> másikfájl #fájl → stdout → másikfájl (append)
- cat fájlnév 2> másikfájl #fájl → stdout, és stderr → másikfájl
- cat fájlnév &> másikfájl #minden a fájlba ömlesztve

Csővezeték (pipe)

Alkalmazások összekötése (jele: | karakter)

```
cat input.txt | grep 'TODO'
#cat stdout-ját a grep stdin-jába
```

Láncolhatóak az alkalmazások… DE…

- Formázatlan bináris adatátadás történik
- Gyors, de strukturált adatot nem kezel
- Strukturált adat: sorok és mezőkre bontás, feldolgozni (Erre használható : cut, awk, sed…)
- Egyszerű adatszerkezeteknél még elmegy…

DEMO Linux és Bash alapok

Bash alapfunkciók

- cat, grep, ls

Alapvető shell funkciók

I/O átirányítások

Fájlok másolása Windows és Linux között

Tartalom

Motiváció: szkriptelés

Linux alapok

Python alapok

Windows PowerShell (következő óra)

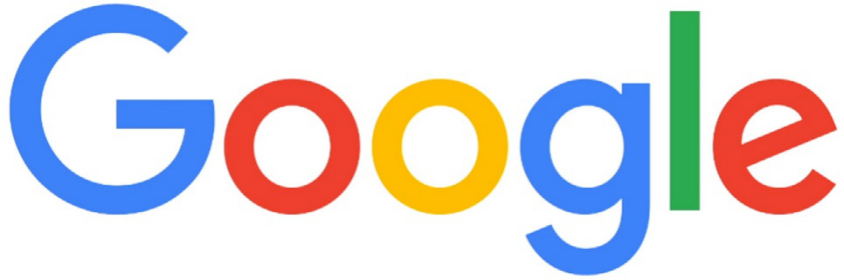
Miért éppen Python?

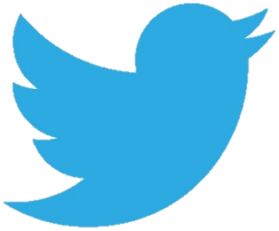
Számos elterjedt szkript nyelv létezik:

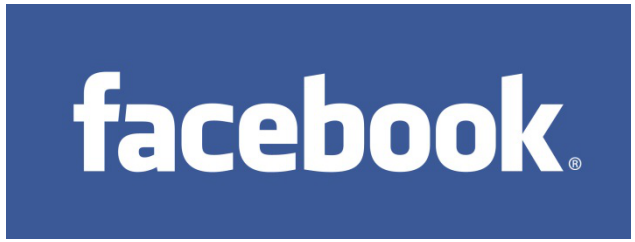
Python

- Hasonlít a már tanult nyelvekhez (C, Java, C#, …)
- Nagyon elterjedt, aktívan fejlesztik
- Jól dokumentált, rengeteg kiegészítéssel

Ki használ Pythont?







stb...

A Python nyelv

1991-ben jelent meg az első verzió

- Jelenleg a 3.4-es verziót használjuk (van 3.5 is)

Általános célú, magas szintű

Több paradigmát is támogat:

- Objektum-orientált
- Imperatív
- Funkcionális

Nem csak szkriptelésre használható

Python filozófia

„Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Readability counts.”

The Zen of Python (PEP20) részlet

Hello world példa

Indítsuk el a Python interpretert

$ python3

FIGYELEM: nem python, hanem python3

- A python az a 2.x verzió!

```
pi@raspberrypi ~/test $ python3
Python 3.2.3 (default, Mar  1 2013, 11:53:50)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Írjunk ki valamit:

>>> print("Hello world!")

Hello world szkript

Kedvenc editorba írjuk be (nano, mcedit, vi, emacs…)

```
#!/usr/bin/env python3
# this is a comment
print( "Hello world" )
```

Első sor: „shebang”

- Egy hint, jelzi, hogy ez milyen fájl is valójában

Adjunk neki futtatási jogot:

```
chmod +x hello.py
```

Futtassuk:

DEMO Python

Python alapfunkciók áttekintése

Hello World példa

Változókezelés

A szokott típusok elérhetőek

- Számok
- Sztringek
- Listák, …

Szkriptnyelv → automatikus típusválasztás

- DE: van típusellenőrzés

Változókonvertáló függvények léteznek

- pl.: `int("6")`
  `str(15)`

Változókezelés

```
irf = "Intelligens rendszerfelügyelet" #Értékadások
year = 2016
course = irf + " " + str(year)

x = y = z = 0                    # többszörös értékadás
a, b = 2, 3

> print (Course)        # Nem definiált változó, hibaüzenet!
                        # (kis-, nagybetű számít!)
> print(course);
    Intelligens rendszerfelügyelet 2016
```

Sztringek kezelése

```
q1 = "Bring us a shrubbery!"
q2 = 'Brave Sir Robin ran away'    # meg lehet így is adni
q3 = '''What is the air-speed velocity
 of an unladen swallow?'''         # triple ': lehet többsoros

print( q1[2] )                     # eredmény: i
```

Sztringek részeinek visszaadása (slicing):

```
    s[start:stop]                  # azon s[k], ahol start <= k < stop

print( q1[6:8] )                   # eredmény: us
print( q1[11:-1] )                 # eredmény: shrubbery
print( q1[:4] )                    # eredmény: Brin
```

Listák

A lista is egy sorozat (sequence):

```
fruits = ["apple", "pear"]

fruits.append("peach")
len(fruits)                    # 3

fruits[1]                              # "pear"

"pear" in fruits# True
```

DEMO Változókezelés

Változók, értékadások

Szövegek kezelése

Listák kezelése

Vezérlési szerkezetek: elágazás

Pythonban zárójelezés helyett blokkok behúzása van!

```
number = -1
if number < 0:
    print("Negative number")
elif number < 3:
    print("Small number")
else:
    print("Big number")
```

Szóköz **VAGY** TAB karakterekkel, de csak az egyikkel

Akár parancssori értelmezőben is használhatjuk

Vezérlési szerkezetek az interpreterben

… jelzi, hogy összetett utasításban vagyunk

Ide külön be kell írni a szóközöket nekünk

Végén egy üres sor jelzi, hogy lezártuk ezt a szerkezetet

```
>>> number = 3
>>> if number < 3:
...     print("Small number")
... elif number < 0:
...     print("Negative number")
... else:
...     print("Big number")
...
Big number
>>>
>>>
```

Ciklusok

For ciklus **sorozaton** iterál végig (~C# foreach)

```
for x in [1, 2, "alma"]:
    print(x)

for i in range(0, 5):
    print(i)
```

while ciklus:

```
# Fibonacci
a, b = 0, 1
while b < 100:
    print(b, end=',')
    a, b = b, a+b
```

Modulok

Előre elkészített segédmodulokat használhatunk

- CSV kezelés (csv)
- Külső parancsok hívása (subprocess)
- Operációs rendszer adatai (os)

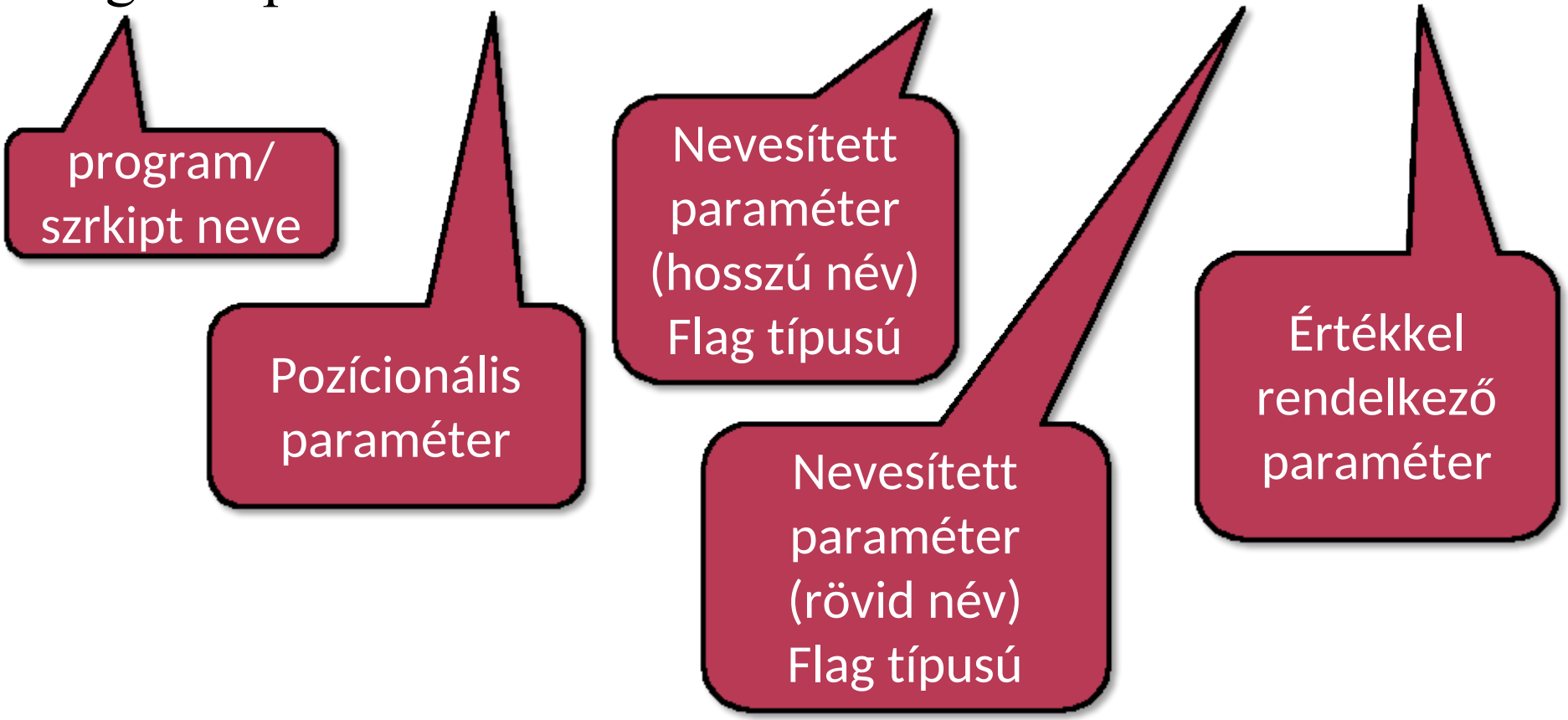
Használatuk:

- import modulename

Parancssori paraméterek

Hogyan használunk egy parancssori programot?

```
wget http://mit.bme.hu/ --verbose -d -t 1
```



Argparse modul

Paraméterek kezelése Pythonban

- sys.argv listában megkapjuk
- lehetne kézzel kezelni, de

argparse: paraméterkezelő modul

- nevesített paraméterek (rövid és hosszú névvel)
- flag-ek
- pozícionális paraméterek
- opcionális paraméterek
- tömbparaméterek

Argparse példa

```
parser = argparse.ArgumentParser();
parser.add_argument("name",
    help="The name to be greeted.",
    type=str)
parser.add_argument("-q", "--quantity",
    help="Amount of greetings.",
    type=int, default=1)
args = parser.parse_args();

args.name
```

Így férünk hozzá a paraméter értékéhez

A szükséges ellenőrzéseket elvégzi helyettünk

Még [-h]elpet is generál

Visszatérési érték

Minden parancsnak van visszatérési értéke

- Következtethetünk belőle a lefutás eredményére
- Ha minden rendben, akkor 0
- Hibás esetekben különböző hibakódok visszaadása

Pythonban: `sys.exit(return_value)`

Főleg paraméterek ellenőrzésénél fontos

DEMO Parancssori paraméterek

ParameterHandlingArgParse.py

- Paraméterek definiálása
- Nevesített paraméterek használata
- Paraméterhibák kezelése

Visszatérési érték

Sztring darabolás

String objektum *partition* vagy *split* metódusával

```
passwd="root:*:0:0:/bin/sh"
first, sep, remainders = passwd.partition(":")
all = passwd.split(":")
print(first)
print(remainders)
print(all)
```

```
> root
> *:0:0:/bin/sh
> ['root', '*', '0', '0', '/bin/sh']
```

Külső parancsok hívása

subprocess.call()

- Parancsok hívása (stdin és stdout használata nélkül)

subprocess.check_output()

- Parancsok hívása az stdin és stdout felhasználásával
- Ha szükséges a parancs kimenetének feldolgozása

Kommentek

Hagyományos és sorvégi kommentek

- # karakter használatával

Fejkommentek (docstring)

- Függvény, osztály, modul elején
- 3-3 idézőjel (") használatával

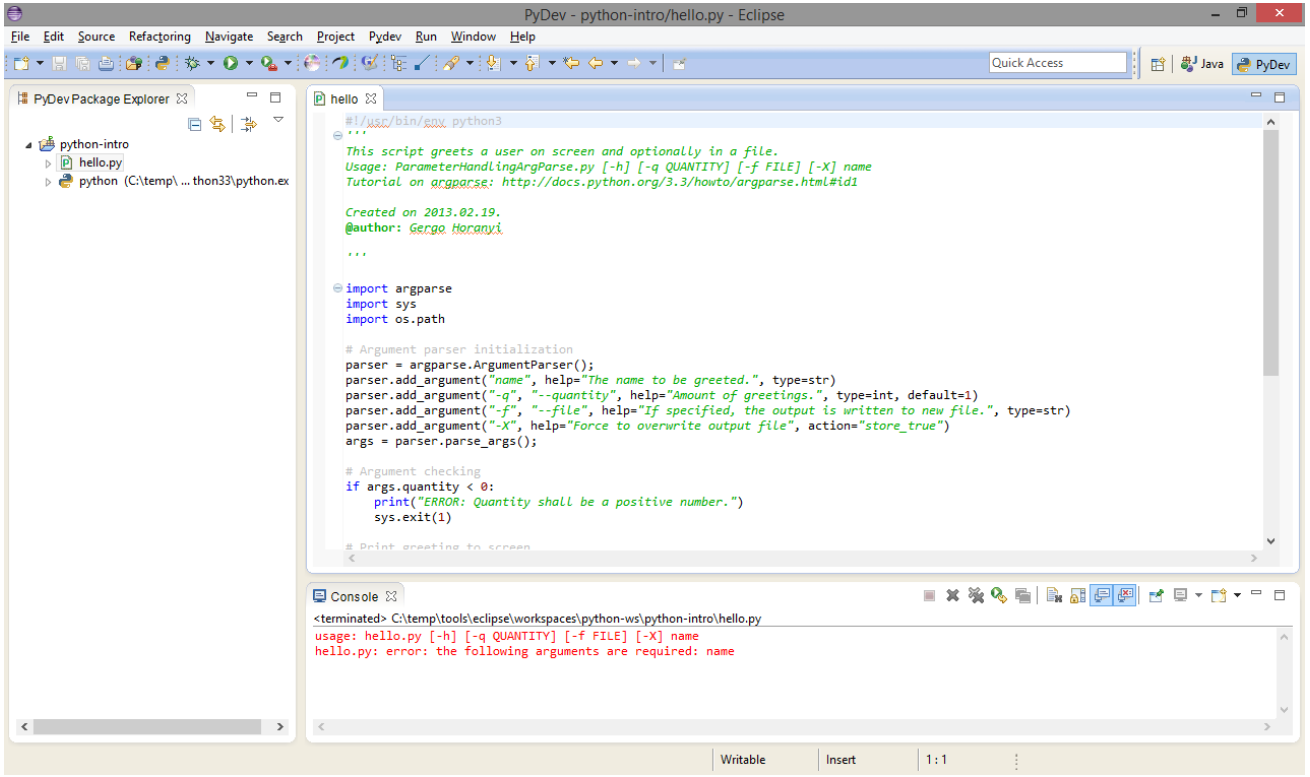
```
def sum(a, b):
    """Return the sum of a and b"""
```

Miben fejlesszünk?

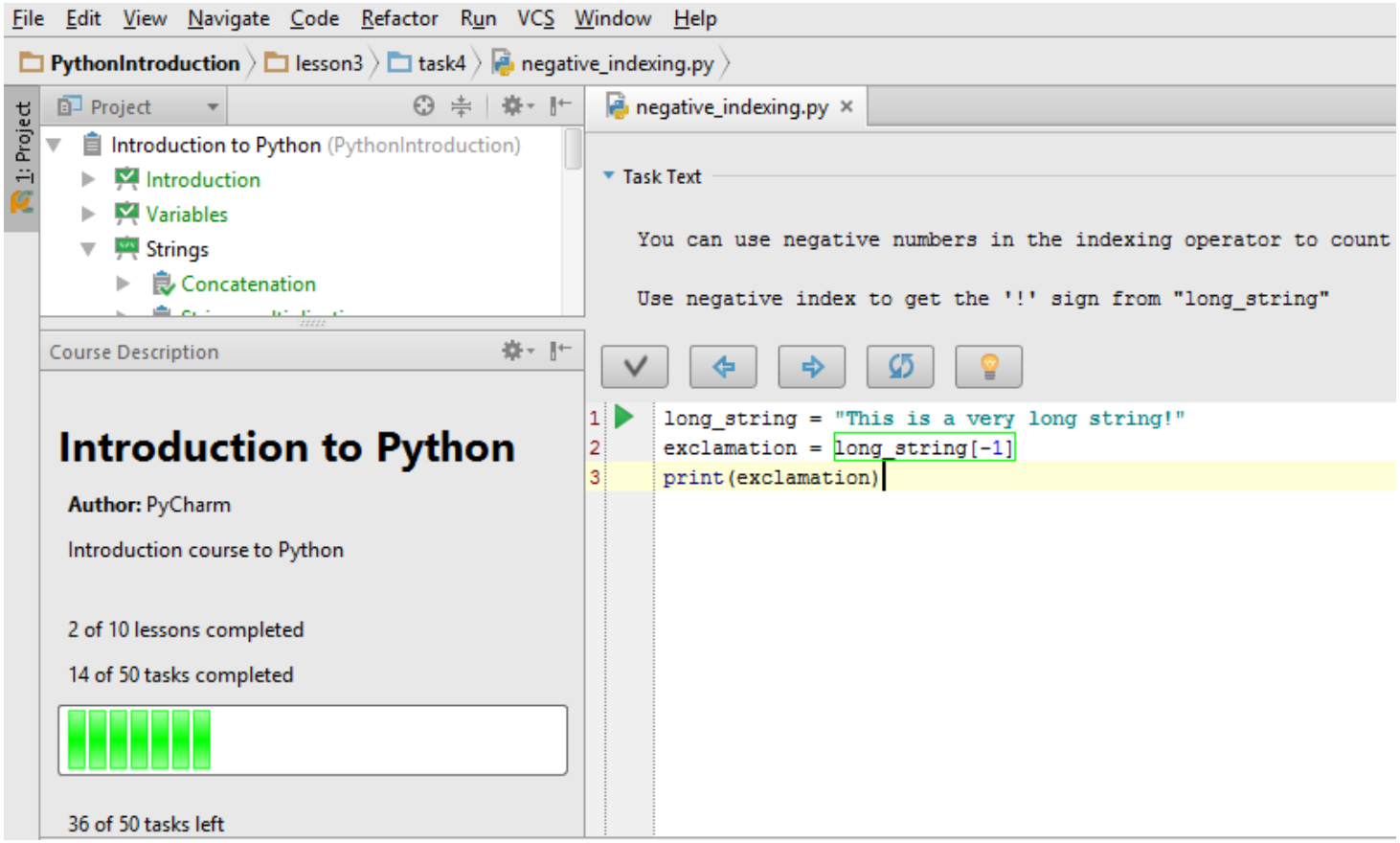
Parancssori fejlesztőeszköz (mcedit, nano, …)

- bármilyen szövegszerkesztő

Integrált fejlesztőkörnyezet (IDE): PyDev



PyCharm Community Edition



Teljes értékű IDE

Python oktatóanyag (task-based)

Online Python Tutor

Start visualizing Python, Java, and JavaScript code now!

For example, here is a visualization showing a Python program that recursively finds the sum of a linked list.
Click the "Forward" button to see what happens as the computer executes each line of code.

Python 2.7

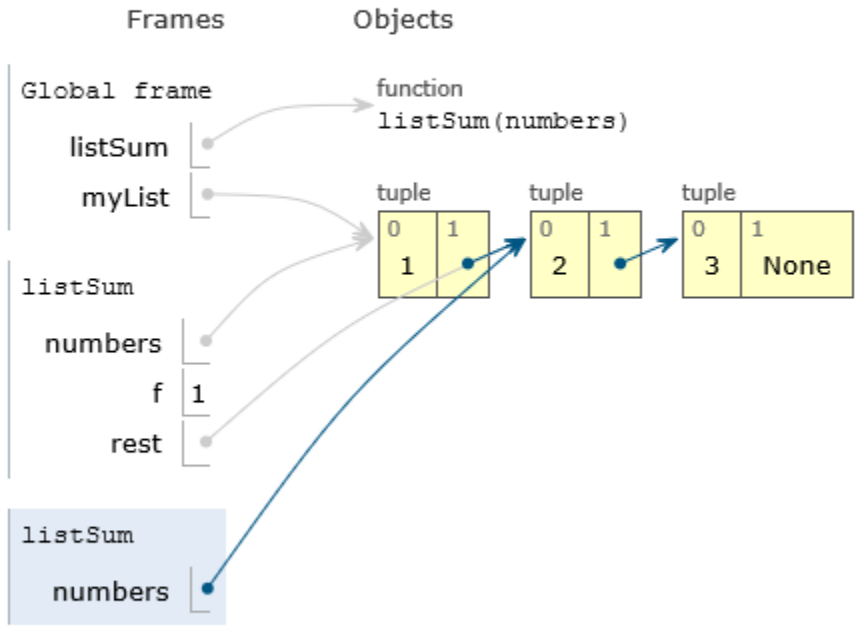
```
1  def listSum(numbers):
2    if not numbers:
3      return 0
4    else:
5      (f, rest) = numbers
6      return f + listSum(rest)
7
8  myList = (1, (2, (3, None)))
9  total = listSum(myList)
```

Edit code

< Back Step 9 of 22 Forward >

line that has just executed
next line to execute

Visualized using Online Python Tutor by Philip Guo



http://pythontutor.com/

Python Style Guide (PEP8)

Hogyan írjunk szép és olvasható kódot?

- Use 4-space indentation, and no tabs.
- Wrap lines so that they don't exceed 79 characters.
- Use blank lines to separate functions and classes, and larger blocks of code inside functions.
- When possible, put comments on a line of their own.
- Use spaces around operators and after commas, but not directly inside bracketing constructs: a = f(1, 2) + g(3, 4).
- Name your classes and functions consistently;
- Don't use non-ASCII characters in identifiers
- …

Ami kimaradt

függvények (def)

osztályok, saját modulok

további adatstruktúrák (dictionary, set, ...)

fájlok olvasása és írása (open, write)

hibakezelés (try/except)

további beépített modulok:

- json, math, random, urllib, datetime, xml, …

...

Feladat

Készítsünk egy olyan szkriptet, ami

- paraméterként kap egy könyvtárnevet
- kiírja, hogy hány alkönyvtár van benne
- kiírja, hogy melyik kiterjesztésből van a legtöbb a könyvtárban lévő fájloknál

További információ

A Unix operációs rendszer:
http://www.hit.bme.hu/~szandi/unix/index.html

man bash, man sed, man cut, man sort, man grep…

Official Python tutorial:
http://docs.python.org/3.4/tutorial/

Google Python class:
https://developers.google.com/edu/python/