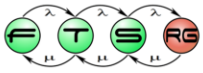
Intelligens rendszerfelügyelet (VIMIA370)

Szkriptelés alapok

Tóth Dániel, Szatmári Zoltán,
Horányi Gergő



Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Utolsó módosítás: 2013. február 26.

Tartalom

- **Műveletek automatizálása, szkriptelés**
  - Eltérések az általános programozási nyelvekhez képest
- Linux alapok
- Python alapok
- Windows PowerShell (következő óra)

Motiváció

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

Motiváció 2.

- Nem szükséges speciális fejlesztői környezet
- A legtöbb számítógépen könnyen elérhető futtatókörnyezet hozzá
- Gyors és hatékony eszköz
- Sok online segédanyag, példa elérhető

Szkriptelés

- Általában a szkript nyelvek jellegzetességei
  - Interpreter futtatja
  - Akár soronként is értelmezhető
  - Minden futási időben értékelődik ki
  - Sok esetben típustalan (de a Python ez alól kivétel)

Tartalom

- Műveletek automatizálása, szkriptelés
  - Eltérések az általános programozási nyelvekhez képest
- **Linux alapok**
- Python alapok
- Windows PowerShell (következő óra)

Linux alapok

- A Bash szkriptnyelven lehet alapvetően programozni
- Csak az alapokat nézzük meg, aztán Python
- 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)*

Linux alapok

- Alapvető Bash funkciók
  - Automatikus kiegészítés
    - TAB billentyű
  - Parancs előzmények tárolása
    - `Fel` és `Le` gombokkal navigálás
    - `CTRL+R` billentyű kombinációval keresés
    - `history` parancs
  - Terminál gyors bezárása
    - `CTRL+D` billentyűkombináció

Átirányítások

- Standard 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 2> másikfájl` #fájl→stdout, stderr→másikfájl
  - `cat fájlnév &> másikfájl` #minden a fájlba ömlesztve

Csővezetékek

- `cat fájl | grep 'x'`
  `#cat stdout-ját a grep stdin-jába`
- Láncolhatóak az alkalmazások… DE…
  - Formázatlan bináris adatátadás történik
  - Gyors, de strukturált adatot nem kezel
  - Strukturált adatot szöveges formába kell alakítani (valamilyen módon), majd a fogadó oldalon sorokra, majd azon belül mezőkre bontva feldolgozni
  - Erre használható programok: cut, awk, sed (tokenizálás, reguláris kifejezések stb.)
  - Erre jó a bash is, pl. a `pipecmd | while read VAR` vagy `for VAR in $(pipecmd)` konstrukciókkal
  - Egyszerű adatszerkezeteknél még elmegy…
    - Az emberek itt szokták értékelni a Powershellt ☺

DEMO Linux és Bash alapok

- Bash alapfunkciók
  - `cat, grep, ls`
- Alapvető shell funkciók
- I/O átirányítások
- Fájlok másolása Windows és Linux között

Tartalom

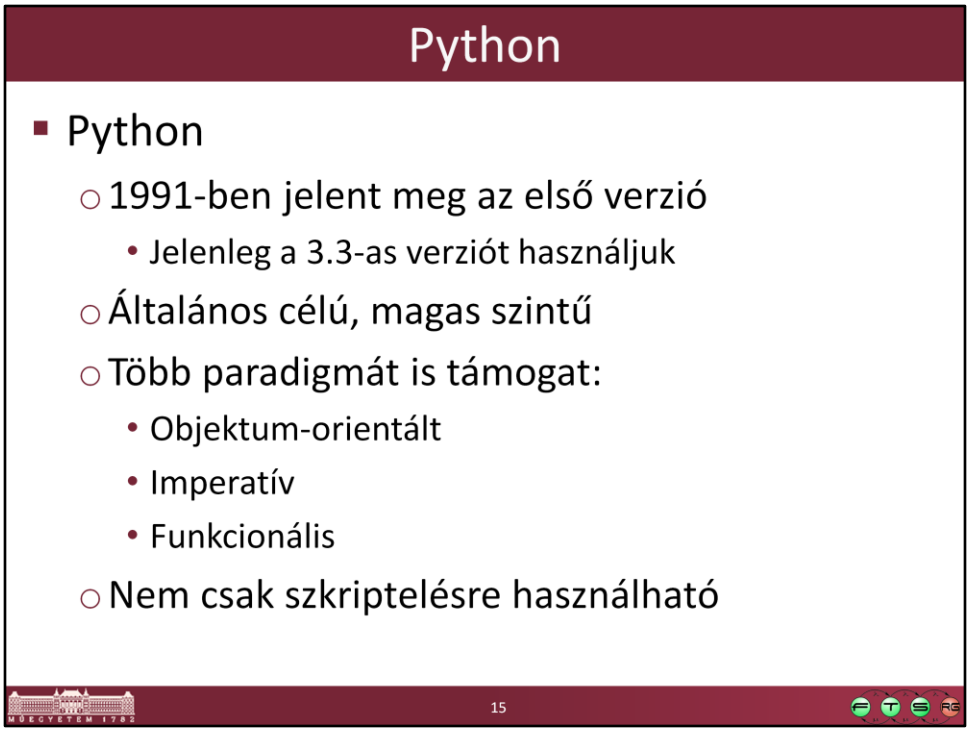
- Műveletek automatizálása, szkriptelés
  - Eltérések az általános programozási nyelvekhez képest
- Linux alapok
- **Python alapok**
- Windows PowerShell (következő óra)

Miért éppen Python?

- Számos elterjedt szkript nyelv létezik
  - Bash
  - Perl
  - Python
  - Ruby
- Python
  - Hasonlít a már tanult nyelvekhez (C, Java, C#, …)
  - Nagyon elterjedt, aktívan fejlesztik
  - Jól dokumentált, rengeteg kiegészítéssel

Érdemes megnézni: https://us.pycon.org/2013/

Google: https://developers.google.com/edu/python/
http://code.google.com/p/google-api-python-client/
Dropbox: https://tech.dropbox.com/2012/12/welcome-guido/
Amazon: http://aws.amazon.com/python/
Twitter: https://github.com/bear/python-twitter
Facebook: https://developers.facebook.com/blog/post/301/
Microsoft: http://pytools.codeplex.com/
HP: http://hp.jobs/sunnyvale-ca/software-designer-javapython-engineer/33914192/job/
Spotify: https://ep2013.europython.eu/conference/talks/spotify-and-python-love-first-sight



Figyeljünk arra, hogy a Python hivatalos honlapja hajlamos a 2.x verziójú dokumentációkat feldobni. Mindig válasszuk ki a 3.3-as verziót!

Python

„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

The full proposal: http://www.python.org/dev/peps/pep-0020/

Hello world példa

- Kedvenc editorba írjuk be (joe, mcedit, vi, emacs, kwrite…)

```
#!/usr/bin/env python3
#ez egy komment
print( "Hello world" )
```

- Az első sor kommentje a „shebang”. Egy hint, a `file` nevű programnak jelzi, hogy ez milyen fájl is valójában.
- Adjunk neki futtatási jogot:

```
chmod +x helloworld.py
```

- Futtassuk (a ./ azért kell, mert az aktuális könyvtár nincs a path-ban)

```
./helloworld.py
```

DEMO Python

- Python alapfunkciók áttekintése
- Hello World példa


```
#!/usr/bin/env python3
print( "Hello world" )
```

Változókezelés

- A szokott típusok elérhetőek
  - Számok
  - Sztringek
  - Listák, …
- Szkriptnyelv → automatikus típusválasztás
  - ennek ellenére erősen típusos nyelv
- Változókonvertáló függvények léteznek
  - pl.: `int("6")`

Változókezelés

```
#!/usr/bin/env python3
versionName = "Mountain Lion" #Értékadások
major = 10
minor = 6 + 2
versionNumber= str(major) + "." + str(minor)

> print("Mac OS X", versionName, versionNumber);
  Mac OS X Mountaion Lion 10.8
> 4 + 6
  10
```


A Python 2.x-es verziójában még működött a print nem függvény változata: print "hello", ez 3.x-nél már csak függvényként működik: print("hello")

Változókezelés

```
#!/usr/bin/env python3
print (versionname)          # Nem definiált változó, hibaüzenet!
                             # (kis-, nagybetű számít!)

print( 1/2 )                 # nem lesz csonkolva, 0.5-ír ki!

x = y = z = 0
a, b = 2, 3

quote = "Bring us a shrubbery!"     # Hozz nekünk egy rekettyést!
print( quote[2] )            # i
print( quote[11:-1] )        # shrubbery
print( quote[:8] )           # Bring us
print( quote[:] )            # Bring us a shrubbery! (másolat)
```

Tömbök

- Vannak tömbök!
  - hasonlóan kezelhetjük, mint a sztringeket tettük

```
fruits = ["apple", "pear"]
fruits.append("peach")
len(fruits)
fruits[0:2] = ["grape", "plum"]

for x in fruits:
     print("This is a fruit", x, sep=": ")
```

DEMO Változókezelés

- Változók, értékadások
- Szövegek kezelése
- Tömbök kezelése


```
#!/usr/bin/env python3
versionName = "Mountain Lion" #Értékadások
major = 10
minor = 6 + 2
versionNumber= str(major) + "." + str(minor)
print("Mac OS X", versionName, versionNumber);
4 + 6
Print(versionName)

print( 1/2 )
x = y = z = 0
a, b = 2, 3

quote = "Bring us a shrubbery!"
print( quote[2] )
print( quote[11:-1] )
print( quote[:8] )
print( quote[:] )
```

Elágazás

- **Pythonban zárójelezés helyett blokkok behúzása (indentation) van!**

```
number = 2
if number < 3:
    print("Small number")
elif number < 0:
    print("Negative number")
else:
    print("Big number")
```

- Szóköz **VAGY** TAB karakterekkel, de csak az egyikkel
- Akár parancssori értelmezőben használhatjuk
- Ne felejtsük le a kettőspontot a végéről
- Logikai és/vagy: and/or

Az intendálást TAB-bal a legegyszerűbb, ilyenkor a SHIFT+TAB-bal az intendálás szintjét lehet csökkenteni.

(A Python Style Guide javaslata szerint azonban érdemesebb átállítani a szövegszerkesztőkent, hogy TAB helyett szóközt rakjon, így biztosítva, hogy máshol is ugyanúgy jelenjen meg a kód. A hivatalos ajánlás 4 darab szóköz használata.)

Ciklusok

- Létezik *foreach* ciklus:

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

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

- És *while* is:

```
# Fibonacci
a, b = 0, 1
while b < 1000:
    print(b, end=',')
    a, b = b, a+b
```


Érdekesség: a for ciklusnak is lehet else ága!

Példa: http://docs.python.org/3.3/tutorial/controlflow.html

```
for n in range(2, 10):
    for x in range(2, n):
        if n % x == 0:
            print(n, 'equals', x, '*', n//x)
            break
    else:
        # loop fell through without finding a factor
        print(n, 'is a prime number')
```

Modulok

- Előre elkészített segédmodulokat használhatunk
  - CSV kezelés (`csv`)
  - Operációs rendszer hívásai (`os`)
  - Reguláris kifejezések kezelése (`re`)
- Használatuk:
  - `import modulename`

Bővebben lásd: Python Tutorial. Chapter 6. Modules, URL: http://docs.python.org/3.3/tutorial/modules.html

Parancssori paraméterek

- Egy listában megkapjuk → azt csinálunk vele amit akarunk
  - persze ez nehézkes lenne, tehát: **`argparse`**
- **`argparse`**
  - nevesített paraméterek (rövid és hosszú névvel)
  - flag-ek
  - pozícionális paraméterek
  - opcionális paraméterek
  - tömbparaméterek

Documentation: http://docs.python.org/dev/library/argparse.html
Tutorial: http://docs.python.org/dev/howto/argparse.html#id1

Argparse

- Példakód:

```
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();
```

- 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

```
import sys
if args.quantity < 0:
  print("ERROR: Quantity shall be a positive number.")
  sys.exit(1)
```

DEMO Parancssori paraméterek

- ParameterHandlingArgParse.py
  - Paraméterek definiálása
  - Nevesített paraméterek használata
  - Paraméterhibák kezelése
- Visszatérési érték

ParameterHandlingArgParse.py
ParameterHandlingArgParse.py IRF
ParameterHandlingArgParse.py IRF -q 5
ParameterHandlingArgParse.py IRF -q 5 -file tmp
cat tmp
ParameterHandlingArgParse.py IRF -q 5 -file tmp
ParameterHandlingArgParse.py IRF -q 5 -file tmp -X

String 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

- `os.system()`
  - Parancsok hívása az `stdin` és `stdout` használata nélkül
- `os.popen()`
  - Parancsok hívása az `stdin` és `stdout` felhasználásával
  - Ha szükséges a parancs kimenetének feldolgozása

Reguláris kifejezések

- Sok helyen használhatjuk őket
  - Pl. sed, awk, grep
  - (Perl, Java, C#...)
  - Egyszerű string manipulációt nagyon megkönnyíti
- Példa kinek a nevét írtuk rosszul

	A	B	C
1	**Személy**	**Kedvenc étel**	**mennyiség**
2	Don Mascarpone	Tiramisu torta	3 szelet
3	Vito Mascarpone	Bolognai spagetti	2 tányér
4	Kicsi Angelo	Gelato fagylalt	5 gombóc
5	Nagy Luzio	Gelato fagylalt	2 gombóc
6	Federico mortellini	mogyoró	nagy zsák

Reguláris kifejezések

- Megoldás:
  - Exportáljuk CSV-be a táblázatot, így fog kinézni:

```
"Személy","Kedvenc étel","mennyiség"
"Don Mascarpone","Tiramisu torta","3 szelet"
"Vito Mascarpone","Bolognai spagetti","2 tányér"
"Kicsi Angelo","Gelato fagylalt","5 gombóc"
"Nagy Luzio","Gelato fagylalt","2 gombóc"
"Federico mortellini","mogyoró","nagy zsák"
```

Reguláris kifejezések

- Egy lehetséges megoldás:

```
import csv
import re
for l in csv.reader(open("csvdemo.csv")):
  if re.match("[A-Z][a-z]* [A-Z][a-z]*",
                l[0]) == None:
     print(l[0])
```

- Eredmény:

```
"Személy"
"Federico mortellini"
```



```
for l in csv.reader(open("csvdemo.csv")):
  if re.match("[A-Z][a-z]* [A-Z][a-z]*", l[0]) == None:
    print(l, l[0])
```

DEMO Reguláris kifejezések

- CSV kezelés
- Reguláris kifejezés kezelés

Reguláris kifejezések Pythonon kívül

- SED == Stream EDitor
  - Alapvetően az stdinről olvasott szöveg-streamen végez programozható átalakításokat, és az eredményt az stdoutra írja.
  - Egyszerre valósítja meg többek között a *cut*, a *grep*, a *tr*, a *head* és a *tail* parancsot.
  - Write-only programozás
    - Példa: Hanoi tornyai

```
s~^xx*$~:n:3:2:1:&:~;tB;d;:B;/^:$/d;h
s~^:.:\(.\):.:\(.\):*:.*~\2 --> \1~;x
/^:y:.:.:.:*:.*/b0;/^:n:.:.:.:x:.*/b1
s~:n:\(.\):\(.\):\(.:x*\)x:\(.*\)~:n:\2:\1:\3:y:\1:\2
:\3x:\4~
bB;:1;x;p;x;s~^:n:.:.:.:x:\(.*\)~:\1~;bB;:0;x;p;x
s~^:y:\(.\):\(.\):\(.\):x\(x*:*\)~:n:\1:\3:\2:\4~ bB
```

DEMO SED

- Hanoi tornyai megoldása SED segítségével
- Kutya – macska karakterlánc csere

cat kutya.txt | sed "s/kutya/macska/g"

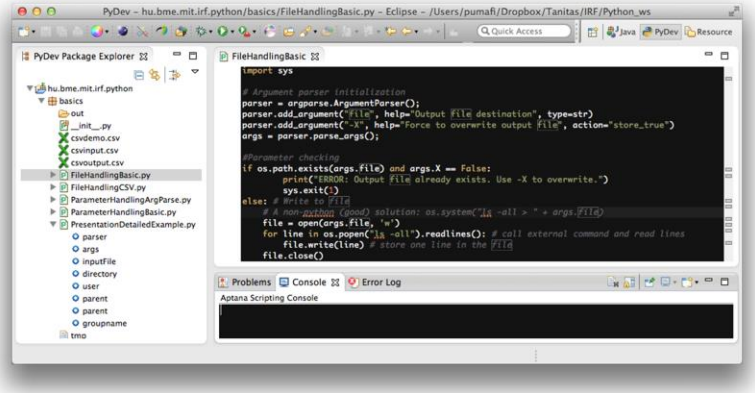
Kommentek

- Hagyományos és sorvégi kommentek
  - # karakter használatával
- Fejkommentek (docstring)
  - Függvény, osztály, modul elején
  - 3-3 idézőjel (") használatával

```
def function(a, b):
    """function(a, b) -> list"""
```

Miben fejlesszünk?

- Parancssori fejlesztőeszköz (`mcedit, nano, …`)
  - bármilyen szövegszerkesztő
- Integrált fejlesztőkörnyezet (IDE): PyDev



Tanácsok, hibakeresés

- Legyen komment a szkript elején
  - Ki írta, mire való, hogy kell paraméterezni
- A bemenő paramétereket ellenőrizzük
  - Mindent vizsgáljunk meg!
- A szkript NE töröljön vagy írjon felül olyan fájlokat, amire nem kértük
  - ☠
  - Ideiglenes fájlokhoz használjuk az mktemp, `tempfile-t`
- Tartsuk be a Python Style Guide-ot (PEP8)

PEP8: http://www.python.org/dev/peps/pep-0008/

Feladatmegoldás

Készítsen egy Python szkriptet, ami fogad egy felhasználó és hozzárendelt könyvtár listát CSV formátumban, létrehozza a felhasználókat és a könyvtárakat és beállítja a jogosultságokat úgy, hogy minden felhasználó be tudjon lépni, olvasni és írni is tudjon a hozzárendelt összes könyvtárban, de ne tudjon belépni egyéb könyvtárakba, amikhez nem volt hozzárendelve. Egy felhasználó több könyvtárhoz és is lehet rendelve és egy könyvtárhoz is több felhasználó lehet rendelve. Posix ACL-eket nem használhat, viszont szükség esetén létrehozhat új csoportokat. Ha a rendszeren meglévő felhasználót talál, azt ne módosítsa, hagyja ki teljesen! Feltételezhet angol locale beállítást. A bemenetet a következő formátumban kapja meg:

```
konyvtar1:usernev1
konyvtar1:usernev2
konyvtar2:usernev2
```

Megoldás felépítés

- **Fejkomment**
- Paraméterek ellenőrzése
- Bemenetből a felhasználók és könyvtárak kigyűjtése
- Még nem létező felhasználók létrehozása
- Még nem létező könyvtárak létrehozása
- Csoportok létrehozása az egyes könyvtárakhoz
- Jogok beállítása

Feladatmegoldás

```
#!/usr/bin/env python3
"""
This script receives a CSV file with directories and associated users and an
   output folder.
Created on 2013.02.19.
@author: Gergo Horanyi
"""
import argparse
import os.path
import sys
import csv

# Initialize argparser
parser = argparse.ArgumentParser()
parser.add_argument("input", help="The path to the input CSV file containing
   the directories and users to be created.")
parser.add_argument("output", help="The parent directory where the new
   directories shall be created.")
```

Megoldás felépítés

- Fejkomment
- **Paraméterek ellenőrzése**
- Bemenetből a felhasználók és könyvtárak kigyűjtése
- Még nem létező felhasználók létrehozása
- Még nem létező könyvtárak létrehozása
- Csoportok létrehozása az egyes könyvtárakhoz
- Jogok beállítása

Feladatmegoldás

```
# Parameter checking
args = parser.parse_args()
if not os.path.exists(args.input):
    print("ERROR: The given input file does not exist.")
    sys.exit(1)
elif not os.path.isdir(args.output):
    print("ERROR: The give output directory
                                    does not exist or not a directory.")
    sys.exit(2)
elif os.popen("id -nu").read() != "root":
    print("ERROR: The script shall be started as root (with sudo).")
    sys.exit(3)

# Everything seems to be all right
```

Megoldás felépítés

- Fejkomment
- Paraméterek ellenőrzése
- **Bemenetből a felhasználók és könyvtárak kigyűjtése**
- **Még nem létező felhasználók létrehozása**
- Még nem létező könyvtárak létrehozása
- Csoportok létrehozása az egyes könyvtárakhoz
- Jogok beállítása

Feladatmegoldás

```
inputFile = open(args.input)

# Iterate over the rows of the CSV file
for row in csv.reader(inputFile).readLines():
        directory = row[0]
        user = row[1]
        if int(os.popen("grep "+user+" /etc/passwd | grep -c ':'").read()) == 0:
                # Check whether the user exists
                print("New user shall be added:", user)
                os.system("useradd " + user)
        if not args.output.endswith("/"):
                # Check whether the output parameter has an "/" at the end
                parent = args.output + "/"
        else:
                parent = args.output
```

Megoldás felépítés

- Fejkomment
- Paraméterek ellenőrzése
- Bemenetből a felhasználók és könyvtárak kigyűjtése
- Még nem létező felhasználók létrehozása
- **Még nem létező könyvtárak létrehozása**
- **Csoportok létrehozása az egyes könyvtárakhoz**
- **Jogok beállítása**

Feladatmegoldás

```
# Create directory if not exists
if not os.path.isdir(parent + directory):
       os.makedirs(parent + directory)
groupname = "irf_example_group_" + directory

# Create group for directory
os.system("groupadd " + groupname)
# Set the group of the directory
os.system("chgrp -R " + groupname + " " + parent + directory)
# Revoke all permissions from the directory
os.system("chmod a-rwx " + parent + directory)
# Add all permissions for the directory
os.system("chmod g+rwx " + parent + directory)
# Add the user to the group.
os.system("usermod -a -G " + groupname + " " + user)
sys.exit(0)
```

További info

- LinuxConfig: „Bash scripting Tutorial”, http://www.linuxconfig.org/Bash_scripting_Tutorial
- 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/3.3/tutorial/
- Google Python class: https://developers.google.com/edu/python/
- PyCon 2013 konferencia https://us.pycon.org/2013/

http://www.linuxconfig.org/Bash_scripting_Tutorial
http://www.hit.bme.hu/~szandi/unix/index.html
http://docs.python.org/3.3/tutorial/
https://developers.google.com/edu/python/
https://us.pycon.org/2013/