# linux: Prosessit

## linux: Prosessit

- Jokainen komento käynnistää vähintään yhden **prosessin**: `ps` tulostaa prosessit
- Kokeile `ps -a` ja `ps -x`
- Jokaiselle prosessilla **tunniste** `PID`
- Normaalisti komento käynnistää prosessin **aktiiviseksi**
- `komento &` käynnistää prosessin **taustalle**
- **Esimerkki:** `emacs nimi &` siirtää editorin **taustalle** ja **linux** komentotulkissa **voi** edelleen antaa muita komentoja
- `kill PID` lopettaa prosessin `PID`, jos siihen on oikeudet
- `top` listaa tietoa käynnissä olevista prosessit prosessorin käytön mukaan
- Ctrl + c lopettaa aktiivisen prosessin kuten vaikkapa `top` yllä

```
jetsu@dx5-flspa-02: ~
File  Edit  View  Search  Terminal  Help
jetsu@dx5-flspa-02:~$ ps
  PID TTY          TIME CMD
30430 pts/10   00:00:00 bash
30880 pts/10   00:00:00 ps
jetsu@dx5-flspa-02:~$ ps -a
  PID TTY          TIME CMD
 5367 pts/7    00:00:12 emacs
15104 pts/0    00:00:03 emacs
15130 pts/2    00:00:22 emacs
15151 pts/0    00:00:03 evince
18243 pts/7    00:00:02 emacs
19161 pts/7    00:06:30 emacs
19209 pts/7    00:07:13 evince
28379 pts/2    00:00:55 evince
30183 pts/1    00:00:12 emacs
30198 pts/1    00:00:09 evince
30727 pts/1    00:00:03 gimp
30737 pts/1    00:00:00 script-fu
30881 pts/10   00:00:00 ps
jetsu@dx5-flspa-02:~$ ps -ax | grep evince
14192 ?        Sl     0:25 evince Kuvamalli2.pdf
14735 ?        Sl     0:02 evince Kuvamalli1.pdf
15151 pts/0    Sl     0:03 evince listofwork.pdf
19209 pts/7    Sl     7:13 evince L1vars.pdf
28379 pts/2    Sl     0:55 evince bootfreq.eps
30198 pts/1    Sl     0:09 evince L7tila.pdf
30883 pts/10   S+     0:00 grep --color=auto evince
jetsu@dx5-flspa-02:~$
```

# String manipulointi (kuva: @LoadingArtist.com)

## python

`A.find('c')` etsii `c`:n paikan `A`:ssa

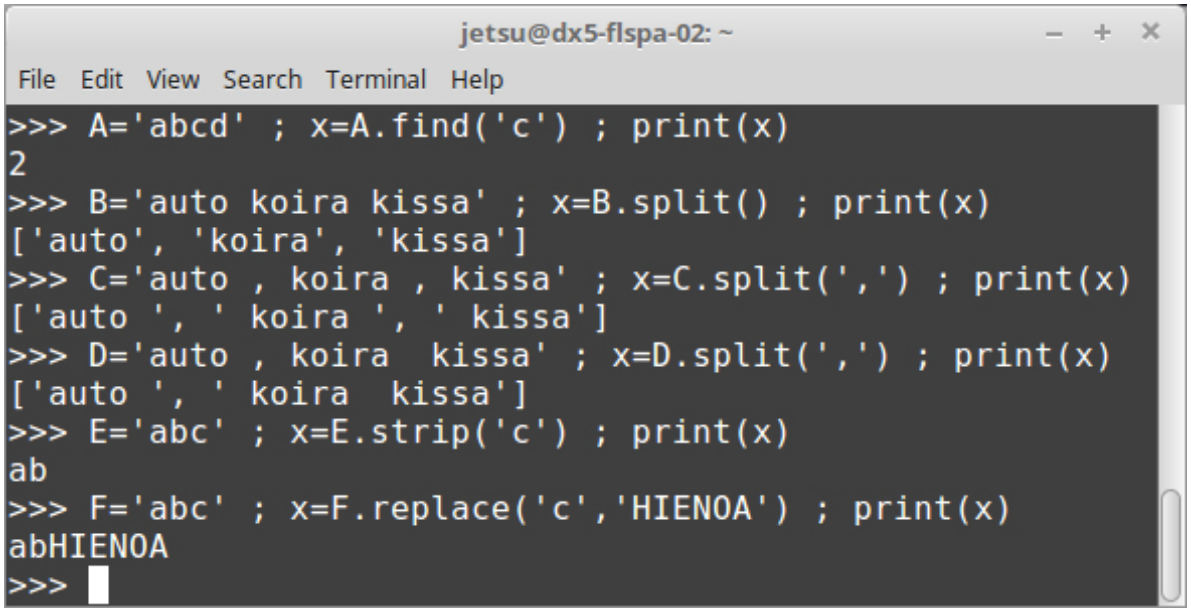
`B.split()` hajottaa `B`:n tyhjien mukaan

`C.split(',')` hajottaa `C`:n pilkkujen `,` mukaan

`D.split(',')` hajottaa `D`:n pilkkujen `,` mukaan
**Huomaa ero!**

`E.strip('c')` poistaa merkin `c`

`F.replace('c','HIENOA')` korvaa merkin `c` merkeillä `HIENOA`




- Näitä **ei tarvitse opetella** ulkoa. Esimerkiksi komennolla `grep .split *.py` löytyy malli kaikista ohjelmista `*.py` joissa esiintyy komento `.split`

# String manipulointi

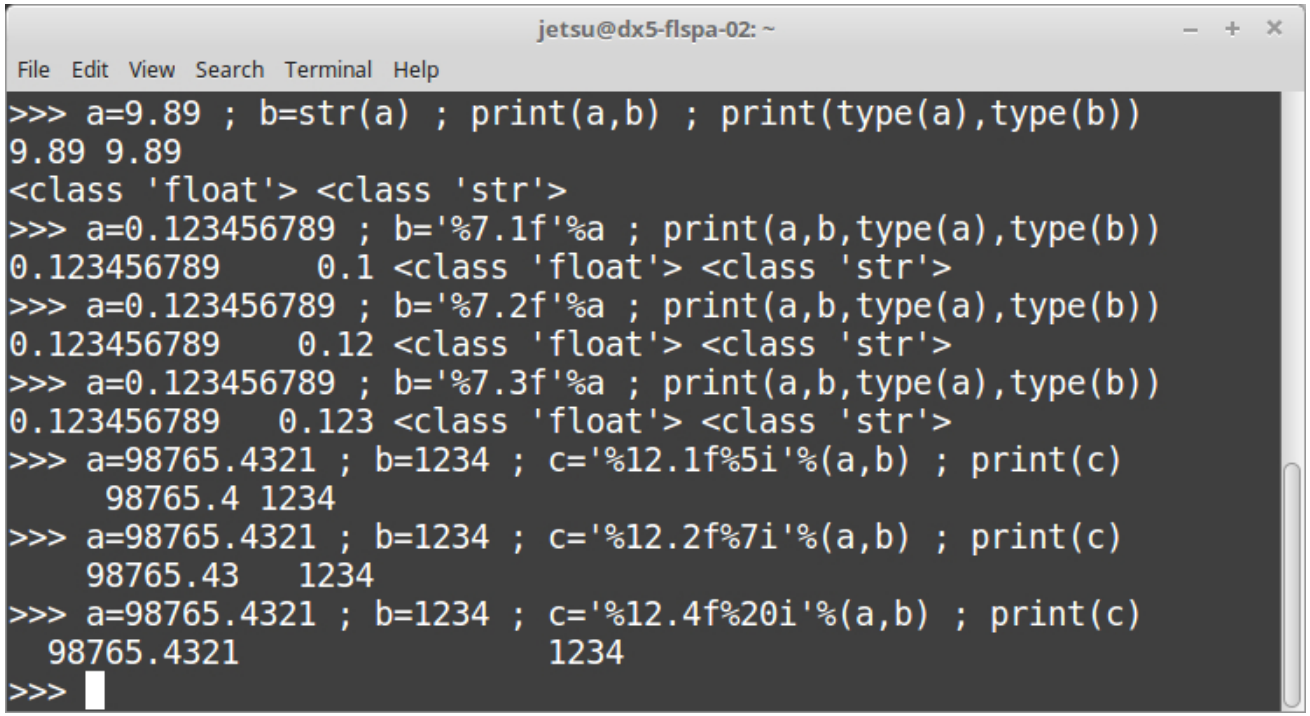
```
# Komenttirivi: Tama on python ohjelmani Stringmalli1.py
# – Joitakin esimerkkeja stringien manipuloinnista
# ____________________________________________________________
import os ; os.system('clear')          # Tyhjennetaan naytto
a="karhunkierros" ; b=a.find("karhu")   # Etsi teksti
print('b =',b)                          # Tulos ============
d ='AbCd' ; e=d.upper()                 # Isoiksi kirjaimiksi
print('d =',d,',e=',e)                  # Tulos ============
f ='EfGh' ; g=f.lower()                 # Pieniksi kirjaimiksi
print('f =',f,',g=',g)                  # Tulos ============
j='xku xnkkx' ; k=j.replace('x','a')    # Korvaa 'x' merkilla 'a'
print('j =',j,',k=',k)                  # Tulos ============
l='ABNyt OKABAB' ; m=l.strip('AB')      # Poista merkit 'AB'
print('l =',l,',m=',m)                  # Tulos ============
o='x;h;8;9' ; p=o.split(';')            # Jaa erottimella =';'
print('o =',o,',p=',p)                  # Tulos ============
print('p[0]=',p[0])                     # p:n 1. elementti
```

- **`python3 Stringmalli1.py`** tulostaa

```
b = 0
d = AbCd ,e= ABCD
f = EfGh ,g= efgh
j = xku xnkkx ,k= aku ankka
l = ABNyt OKABAB ,m= Nyt OK
o = x;h;8;9 ,p= ['x', 'h', '8', '9']
p[0]= x
```

# Formatointi

- Reaaliluvun `a` muunnos stringiksi `b`
- Reaaliluvun `a` muunnos stringiksi `b`, jossa `7.1f` = 7 merkkiä ja 1 desimaali `7.2f` = 7 merkkiä ja 2 desimaalia jne ...



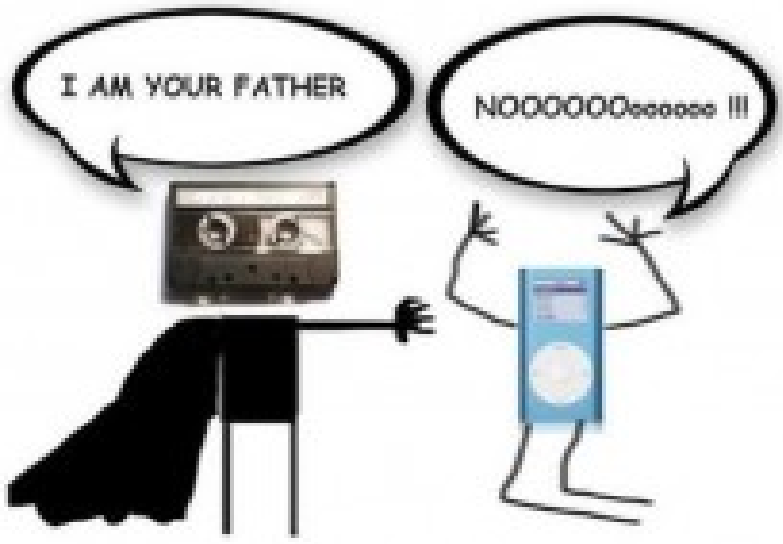

- Reaaliluvun `a` ja kokonaisluvun `b` muunnos stringiksi `c`
  `12.1f`= 12 merkkiä & 1 desimaali, `5i` = 5 merkkiä
  `12.4f`= 12 merkkiä & 4 desimaalia, `7i` = 7 merkkiä jne...
- **... ei tarvitse opetella** ulkoa. Esimerkiksi
  `grep .2f *.py` ⇒ Malli kaikista ohjelmista `*.py`, joissa esiintyy komento `.2f`

# Formatointi (kuva:@improgrammer.net)

```
# Komenttirivi: Tama on python ohjelmani Formatointimalli1.py
# — Joitakin string muunnoksen formatointi esimerkkeja
# ____________________________________________________________
import os ; os.system('clear')          # Tyhjennetaan naytto
# Reaaliluvun muunnos stringiksi ============================
a=1.458                                 # Luku
b=str(a)                                # Muunnos stringiksi
print('b=',b,',␣type(b)=',type(b))      # Tarkistus
# Reaaliluvun formatointi: leveys=8, desimaalit=5 ===========
c=7.89                                  # Luku
d='%8.5f' %c                            # Formatointi
print('d=',d)                           # Tarkistus
# Kokonaisluvun formatointi: leveys=6 =======================
e=71                                    # Luku
f='%6i' %e                              # Formatointi
print('f=',f)                           # Tarkistus
# Reaaliluvun ja kokonaisluvun yhdistetty formatointi =======
g=2.34 ; h=-912                         # Luvut
i='%4.1f%5i' %(g,h)                     # Formatointi
print('i=',i)                           # Tarkistus
# Reaaliluku, string ja kokonaisluku yhdistetty =============
j=9.84 ; k='tekstia' ; l=812            # Yhdistettavat
m='%6.0f%10s%7i' %(j,k,l)               # Formatointi
print('m=',m)                           # Tarkistus
# Reaaliluku, string, kokonaisluku ja '\&' yhdistetty ===
n=9.23 ; o=-87 ; p='\\\\'␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#␣Yhdistettavat
q='%8.3e & %3i %2s'␣%(n,o,p)␣␣␣␣␣␣␣␣␣␣␣␣␣#␣Formatointi
print('q=',q)␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#␣Tarkistus
```

- **`python3 Formatointimalli1.py`** tulostaa

```
b= 1.458 , type(b)= <class 'str'>
d=  7.89000
f=      71
i=  2.3 -912
m=      10    tekstia     812
q= 9.230e+00 & -87 \\
```



# Formatointi

```
# Komenttirivi: Tama on python ohjelmani Formatointimalli2.py
# – Joitakin print komennon formatointi esimerkkeja
# ____________________________________________________________
import os ; os.system('clear')          # Tyhjennetaan naytto
# Esimerkkeja formatoidusta print komennosta ================
a=0.123456789 ; b='abcde' ; c=123456    # float, string, integer
print("%6.2f"%(a))                      #  6 merkkia, 2 desimaali
print("%12.5f"%(a))                     # 12 merkkia, 5 desimaalia
print("%8s"%(b))                        # 12 merkkia
print("%30s"%(b))                       # 30 merkkia
print("%6i"%(c))                        #  6 merkkia
print("%15i"%(c))                       # 15 merkkia
print("%12.3f%10s%8i"%(a,b,c))          # Yhdistelma
print("%12.3f␣&␣%10s␣&␣%8i"%(a,b,c))    # Lisatty kaksi " & "merkkia
```

- **`python3 Formatointimalli2.py`** tulostaa

```
 0.12
     0.12346
   abcde
                          abcde
123456
         123456
       0.123     abcde  123456
       0.123 &       abcde &   123456
```

# Datan lukeminen ja kirjoittaminen kuva: www.cartoonstock.com

- **Formatointi:**
- Datan lukeminen, kirjoittaminen ja “näyttäminen”
- Tärkein on lopputulos: Tulos sama eri tavoilla
- Malliohjelmat: **“Yritys ja erehdys”** n. 30 tuntia
- **Luetaan** ensin **yksinkertaisempaa** tiedostoa `Mallidata1.dat` ohjelmilla
  `LueTiedosto1A.py` ( `readline`=rivi)
  `LueTiedosto1B.py` ( `readlines`=rivilista)
  `LueTiedosto1C.py` ( `numpy.loadtxt`)
- **Luetaan** sitten **monimutkaisempaa** tiedostoa `Mallidata2.dat` ohjelmilla
  `LueTiedosto2A.py` ( `readline`=rivi)
  `LueTiedosto1B.py` ( `readlines`=rivilista)
  `LueTiedosto1C.py` ( `numpy.loadtxt`)
- **Kirjoittaminen**
  `KirjoitaTiedosto1.py` ( `f.write`)
  `KirjoitaTiedosto2.py` ( `numpy.savetxt`)
- **TAVOITE:** Näistä **malleista** on teille **myöhemmin hyötyä** kun luette tai kirjoitatte tiedostoja

# Perusidea!

```
WHATEVER=open('...','r')
WHATEVER.readline()
WHATEVER.close()
```

```
IHANSAMA=open('...','w')
IHANSAMA.write(...)
IHANSAMA.close()
```

# Datan lukeminen: RIVI KERRALLAAN (readline)

Luettava tiedosto **Mallidata1.dat** muotoa

```
1  A
2  B
3  C
```

**python LueTiedosto1A.py** tulostaa

```
[1.0] ['A']
[1.0, 2.0] ['A', 'B']
[1.0, 2.0, 3.0] ['A', 'B', 'C']
```

- **readline** lukee **RIVI KERRALLAAN** kaksi kolumnia muuttujaan **rivi**
- Lukee stringinä $\Rightarrow$ **float(lukunyt)**

```
# Komenttirivi: Tama on ohjelmani LueTiedosto1A.py
import os ; os.system('clear')          # Tyhjennetaan naytto
luku=[]   ; sana=[]                     # Luo kaksi tyhjaa muuttujaa
file = open('Mallidata1.dat', 'r')      # Avaa tiedosto Mallidata1.dat
rivi = file.readline()                  # LUETAAN RIVI KERRALLAAN
while (len(rivi) > 0):                  # Ei tyhja rivi
    osat=rivi.split()                   # Tyhjien mukaan osiksi
    lukunyt=osat[0]                     # 1. elementti
    sananyt=osat[1]                     # 2. elementti
    luku.append(float(lukunyt))         # Lisaa arvo muuttujaan luku
    sana.append(sananyt)                # Lisaa arvo muuttujaan sana
    print(luku,sana)                    # Nayta nykyiset arvot
    rivi = file.readline()              # Lue seuraava rivi
file.close()                            # Sulje Mallidata1.dat
```

# Datan lukeminen: KAIKKI RIVIT KERRALLA (readlines)

Luettava tiedosto **Mallidata1.dat** muotoa

```
1  A
2  B
3  C
```

**python LueTiedosto1B.py** tulostaa

```
[1.0] ['A']
[1.0, 2.0] ['A', 'B']
[1.0, 2.0, 3.0] ['A', 'B', 'C']
```

- **readlines** lukee **KAIKKIEN RIVIEN KAIKKI** kaksi kolumnia **KERRALLA** muuttujaan **data**
- Lukee stringinä $\Rightarrow$ **float(lukunyt)**

```
# Komenttirivi: Tama on ohjelmani LueTiedosto1B.py
# ____________________________________________________________
import os ; os.system('clear')      # Tyhjennetaan naytto
f = open('Mallidata1.dat', 'r')     # Avaa Mallidata1.dat
data = f.readlines()                # LUETAAN RIVILISTANA
f.close()                           # Sulje Mallidata1.dat
luku=[]    ; sana=[]                # Luo kaksi tyhjaa muuttujaa
for rivi in data:                   # Looppi muuttujan data riveille
    osat=rivi.split()               # Tyhjien mukaan osiksi
    lukunyt=osat[0]                 # 1. elementti
    sananyt=osat[1]                 # 2. elementti
    luku.append(float(lukunyt))     # Lisaa arvo muuttujaan luku
    sana.append(sananyt)            # Lisaa arvo muuttujaan sana
    print(luku,sana)                # Nayta nykyiset arvot
```

# Datan lukeminen: numpy.loadtxt ...

Luettava tiedosto **Mallidata1.dat** muotoa

```
1  A
2  B
3  C
```

**python LueTiedosto1C.py** tulostaa

```
[ 1.  2.  3.] ['A' 'B' 'C']
[ 0.54030231 -0.41614684 -0.9899925 ]
```

- **np.loadtxt** lukee **1. kolumnin**
- **np.genfromtxt** lukee **2. kolumnin**
- **np.loadtxt** lukee suoraan vektoriksi, koska **np.cos(luku))** onnistuu

```
# Kommenttirivi: tama on python ohjelmani LueTiedosto1C.py
# ____________________________________________________________
import os ; os.system('clear')                    # Tyhjennetaan naytto
import numpy as np                                # numpy importoitu
file='Mallidata1.dat'                             # Tiedoston nimi
luku=np.loadtxt(file,dtype='float',usecols=(0,)) # Lue 1. kolumni
sana=np.genfromtxt(file,dtype='str',usecols=(1,)) # Lue 2. kolumni
print(luku,sana)                                  # Nayta muuttujien arvot
print(np.cos(luku)) # Muuttuja luku=vektori. Siita voi laskea esim. cosinin
```

- Kokeilin myös komentoa

  **sana=np.loadtxt(file,dtype='str',usecols=(1,))**

  **print(sana)** tulostaa **["b'A'b'B'b'C'"]**

# Vaikeamman datan lukeminen RIVI KERRALLAAN (readline)

- Luettava tiedosto **Mallidata2.dat** muotoa

Mallidata2.dat

```
HJD         Year M  D  H  M
2454395.96 2007 10 22 10 59
2454399.48 2007 10 25 23 35
2454403.01 2007 10 29 12 11
```

**python LueTiedosto2A.py** tulostaa

```
[]
[]
[2454395.96]
[2454395.96, 2454399.48]
[2454395.96, 2454399.48, 2454403.01]
date = [2454395.96, 2454399.48, 2454403.01]
type(date)= <class 'list'>
```

Lue **readline** avulla **riviltä 3. alkaen 1. kolumni**

```
# Komenttirivi: Tama on ohjelmani LueTiedosto2A.py
import os ; os.system('clear')          # Tyhjennetaan naytto
date=[]                                 # Tyhja muuttuja
file = open('Mallidata2.dat', 'r')      # Avaa tiedosto Mallidata2.dat
rivi = file.readline()                  # LUETAAN RIVI KERRALLAAN
while (len(rivi) > 0):                  # Ei tyhja rivi
    osat=rivi.split()                   # Tyhjien mukaan osiksi
    osa=osat[0]                         # 1 alkio
    q=osa.find("245")                   # Etsi "245" tekstia
    if (q>-1):                          # Jos loytyy ...,
# String muutetaan desimaaliluvuksi kaskylla "float"
        date.append(float(osa))         # Lisaa arvo muuttujaan date
    print(date)                         # Nayta nykyinen date
    rivi = file.readline()              # Lue seuraava rivi
file.close()                            # Sulje Mallidata2.dat
print("date = ",date)                   # Tarkistus
print("type(date)=",type(date))         # Tarkistus
```

# Vaikeamman datan lukeminen KAIKKI RIVIT KERRALLA (readlines)

Luettava tiedosto **Mallidata2.dat** muotoa

```
Mallidata2.dat
HJD         Year M  D  H  M
2454395.96 2007 10 22 10 59
2454399.48 2007 10 25 23 35
2454403.01 2007 10 29 12 11
```

**python LueTiedosto2B.py** tulostaa

```
[]
[]
[2454395.96]
[2454395.96, 2454399.48]
[2454395.96, 2454399.48, 2454403.01]
date = [2454395.96, 2454399.48, 2454403.01]
type(date)= <class 'list'>
```

- **readlines** avulla **riviltä 3. alkaen 1. kolumni**

```
# Komenttirivi: Tama on python ohjelmani LueTiedosto2B.py
# ____________________________________________________________
import os ; os.system('clear')        # Tyhjennetaan naytto
f = open('Mallidata2.dat', 'r')       # Avaa Mallidata2.dat
data = f.readlines()                  # LUETAAN RIVILISTANA
f.close()                             # Sulje Mallidata2.dat
date = []                             # Luo tyhja lista
for rivi in data:                     # Looppi datan riveille
    osat=rivi.split()                 # Tyhjien mukaan osiksi
    osa=osat[0]                       # 1 alkio
    q=osa.find("245")                 # Etsi "245" tekstia
    if (q>-1):                        # Jos loytyy ...,
# String muutetaan desimaaliluvuksi kaskylla "float"
        date.append(float(osa))       # Lisaa arvo muuttujaan date
    print(date)                       # Nayta nykyinen date
print("date = ",date)                 # Tarkistus
print("type(date)=",type(date))       # Tarkistus
```

# Vaikeamman datan lukeminen HELPOSTI (numpy.loadtxt)

Luettava tiedosto **Mallidata2.dat** muotoa

```
Mallidata2.dat
HJD         Year M  D  H  M
2454395.96 2007 10 22 10 59
2454399.48 2007 10 25 23 35
2454403.01 2007 10 29 12 11
```

- **skiprows=2** ohittaa **kaksi ensimmäistä riviä**
- **loadtxt** lukee **riviltä 3. alkaen 1. kolumnin** luvut
- **loadtxt** default: **dtype='float'** joten sitä ei tarvitse mainita

**python LueTiedosto2C.py** tulostaa

```
date = [ 2454395.96  2454399.48  2454403.01]
type(date)= <class 'numpy.ndarray'>
[ 0.00415457  0.36557712 -0.69084715]
```

- **date** on valmiiksi vektori, josta voi ottaa vaikkapa cosinin **np.cos(date)**

```
# Kommenttirivi: tama on ohjelmani LueTiedosto2C.py
# ------------------------------------------------------------------
import os ; os.system('clear')                    # Tyhjennetaan naytto
import numpy as np                                # numpy importoitu
file='Mallidata2.dat'                             # Tiedoston nimi
# Komennolla "skiprows=2" jatetaan lukematta 2 ensimmaista rivia
date=np.loadtxt(file,skiprows=2,usecols=(0,)) # Lue 1. kolumni
print("date = ",date)                             # Tarkistus
print("type(date)= ",type(date))                  # Tarkistus
print(np.cos(date))                               # Tarkistus
```

# Datan kirjoittaminen RIVI KERRALLAAN (write) (kuva: @quotesgram.com)

- **KirjoitaTiedosto1.py** ohjelma kirjoittaa kaksi tiedostoa **KirjoitaData1a.dat** ja **KirjoitaData1b.dat**

```
# Komenttirivi: Tama on ohjelmani KirjoitaTiedosto1.py
# — Kirjoitetaan tiedostoon rivi kerrallaan
# ____________________________________________________
import os ; os.system('clear')              # Tyhjennetaan naytto
import numpy                                # Importoidaan Numpy
x=numpy.arange(3.) ; print("x=",x)          # Luodaan 1 vektori
y=numpy.sin(x)     ; print("y=",y)          # Luodaan 2 vektori
# ____________________________________________________
# — Kirjoitetaan vain lukuja
f=open('KirjoitaData1a.dat','w')            # Avaa tiedoston
for i in range(len(x)):                     # for—looppi
    f.write("%6.1f%8.4f\n" %(x[i],y[i]))    # Kirjoittaa rivin
f.close()                                   # Sulkee tiedoston
# ____________________________________________________
# — Kirjoitetaan lukuja ja stringeja esim: Latex kayttoon
f=open('KirjoitaData1b.dat','w')            # Avaa tiedoston
for i in range(len(x)):                     # for—looppi
    f.write("%6.1f%3s%8.4f%4s\n"%(x[i],'&',y[i],'\\\\'))
f.close()                                   # Sulkee tiedoston
```

**python KirjoitaTiedosto1.py** tulostaa

```
x= [ 0.  1.  2.]
y= [ 0.          0.84147098  0.90929743]
```

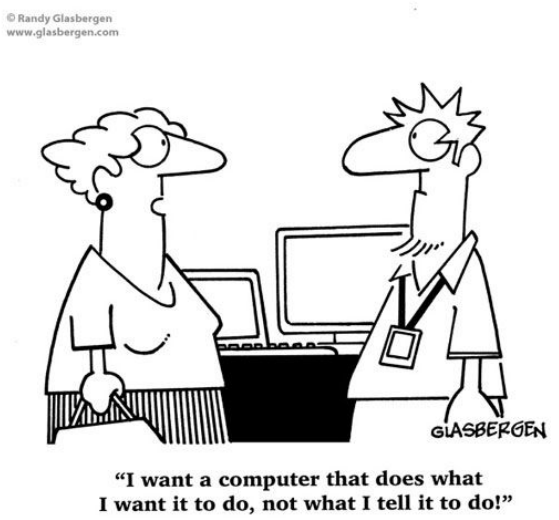
- Alempi tapa **käytännöllinen laskuharjoituksissa**

- **KirjoitaData1a.dat** on muotoa

```
0.0   0.0000
1.0   0.8415
2.0   0.9093
```

- **KirjoitaData1b.dat** on muotoa

```
0.0   &   0.0000   \\
1.0   &   0.8415   \\
2.0   &   0.9093   \\
```



# Numeerisen datan kirjoittaminen HELPOSTI (numpy.savetxt)

- **KirjoitaTiedosto2.py** ohjelma kirjoittaa kaksi tiedostoa **KirjoitaData2a.dat** ja **KirjoitaData2b.dat**

```
# Komenttirivi: Tama on ohjelmani KirjoitaTiedosto2.py
# -- Eras nopea tapa kirjoittaa kerralla kaikki data
# ____________________________________________________________
import os ; os.system('clear')          # Tyhjennetaan naytto
import numpy as np                      # Importoidaan Numpy
x=np.arange(3.) ; print("x=",x)         # Luodaan 1 vektori
y=np.sin(x)     ; print("y=",y)         # Luodaan 2 vektori
# ____________________________________________________________
np.savetxt('KirjoitaData2a.dat',(x,y),fmt='%10.3f')
np.savetxt('KirjoitaData2b.dat',np.transpose((x,y,)),fmt='%10.3f')
```

**python KirjoitaTiedosto2.py** tulostaa

```
x= [ 0.  1.  2.]
y= [ 0.          0.84147098  0.90929743]
```

- Tämä yhden rivi komento **np.savetxt** on **käytännöllinen** kun tahdotaan kirjoittaa lukuja nopeasti ja helposti tiedostoihin

- **KirjoitaData2a.dat** on muotoa

```
     0.000     1.000     2.000
     0.000     0.841     0.909
```

- **KirjoitaData2b.dat** on muotoa

```
     0.000     0.000
     1.000     0.841
     2.000     0.909
```

# **python**: Kirjoitusvinkki

- Kokeillaan **ENSIN** kunnes `print(...` tuottaa toivotun formaatin
- **SITTEN** sama formaatti `f.write(...` ja rivin vaihto `\n`
- **ETU 1:** Ei tarvitse katsoa/avata/sulkea `'KirjoitusVinkki.tex'`
- **ETU 2:** Ohjelman kaatumisesta ennen sopivan formaatin löytymistä ei ole suurta haittaa/lisätyötä

b

```
# KirjoitusVinkki.py
# - ENSIN kokeilu "print(..."
# - SITTEN kun onnistuu, "f.write(..." ja "\n"
# ____________________________________________________________
import os ; os.system('clear')                  # Tyhjennetaan naytto
import numpy                                    # Importoidaan Numpy
x=numpy.arange(3.) ; y=numpy.sin(x)
f=open('KirjoitusVinkki.tex','w')
for i in range(len(x)):
    print("%6.1f%3s%8.4f%4s"%(x[i],'&',y[i],'\\\\'))
    #f.write("%6.1f%3s%8.4f%4s\n"%(x[i],'&',y[i],'\\\\'))
f.close()
```