Programmeermethoden NA

Week 10: NumPy, Python module showcase

Kristian Rietveld

http://liacs.leidenuniv.nl/~rietveldkfd/courses/prna/

Universiteit Leiden

Inhoud vandaag

De laatste zaken om te bespreken:

- NumPy
  - Een aantal laatste NumPy operatoren.
  - Elementen selecteren met maskers.
  - 3-dimensionale NumPy arrays.
- Iteratietechnieken.
- Python module showcase & hoe meer leren over Python?

Reductieoperatoren langs een as

- Tot nu toe reductieoperatoren op volledige array. Uitkomst: een enkel element.
- Je kunt een reductie ook langs een bepaalde as van de array laten plaatsvinden.
- Geef als parameter mee: `axis=1` met `1` het nummer van de as (geteld vanaf 0).

Reductie langs een as

```
>>> A = np.tile( [1,2,3], (3,1))
>>> A
array([[1, 2, 3],
       [1, 2, 3],
       [1, 2, 3]])
>>> A.sum(axis=0)
array([3, 6, 9])
>>> A.sum(axis=1)
array([6, 6, 6])
```

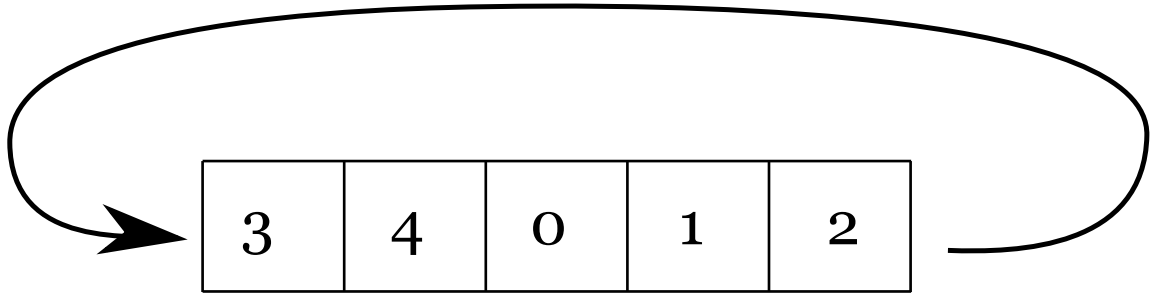
`cumsum` operator

Met `cumsum` kun je cumulatief sommeren. Het aantal dimensies neemt dan niet af, dus dit is geen reductie operator.

```
>>> A = np.arange(15).reshape(3, 5)
>>> A
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> A.cumsum()
array([  0,   1,   3,   6,  10,  15,
        21,  28,  36,  45,  55,  66,
        78, 91, 105])
>>> A.cumsum(axis=1)
array([[ 0,  1,  3,  6, 10],
       [ 5, 11, 18, 26, 35],
       [10, 21, 33, 46, 60]])
```

Rollen

Door middel van "rollen" kunnen we elementen in een matrix "n" plaatsen opschuiven.




```
np.roll(np.arange(5), 2)
```

Roteren en spiegelen

Tenslotte nog operaties om te roteren en te spiegelen.

```
>>> A = A.arange(0, 9).reshape( (3, 3) )
# Draai 90 graden tegen de klok in
>>> np.rot90(A)
array([[2, 5, 8],
       [1, 4, 7],
       [0, 3, 6]])
# Horizontaal spiegelen. Er is ook np.flipud()
>>> np.fliplr(A)
array([[2, 1, 0],
       [5, 4, 3],
       [8, 7, 6]])
```

All en any

Om te kijken of een array aan een bepaalde conditie (Boolean expessie) voldoet, kunnen we gebruik maken van `np.all()` en `np.any()`.

```
>>> A = np.arange(10, 19).reshape( (3, 3) )
# Zijn alle elementen >= 15?
>>> np.all(A >= 15)
False
# >= 10?
>>> np.all(A >= 10)
True
# Is er tenminste een element gelijk aan 14?
>>> np.any(A == 14)
True
# En aan 4?
>>> np.any(A == 4)
False
# Tenminste een element kleiner dan 10?
>>> np.any(A < 10)
False
```

Maskers

In de vorige slide gebeuren er eigenlijk twee dingen:

- Er wordt een Boolean array gemaakt: een masker.
- Er wordt gekeken of ten minste een, of alle, Boolean waarden True zijn.

```
>>> A = np.arange(10, 19).reshape((3, 3))
>>> A >= 15
array([[False, False, False],
       [False, False,  True],
       [ True,  True,  True]], dtype=bool)
# Tel aantal keer true in de Boolean array
>>> np.sum(A >= 15)
4
```

Selectie van elementen

We kunnen maskers ook gebruiken om elementen te selecteren!

```
>>> mask = A >= 15
# Druk elementen >= 15 af. (Let op: 1-d view)
>>> print A[mask]
[15 16 17 18]
# Tel 100 op bij elementen >= 15
>>> A[mask] += 100
>>> print A
[[ 10  11  12]
 [ 13  14 115]
 [116 117 118]]
```

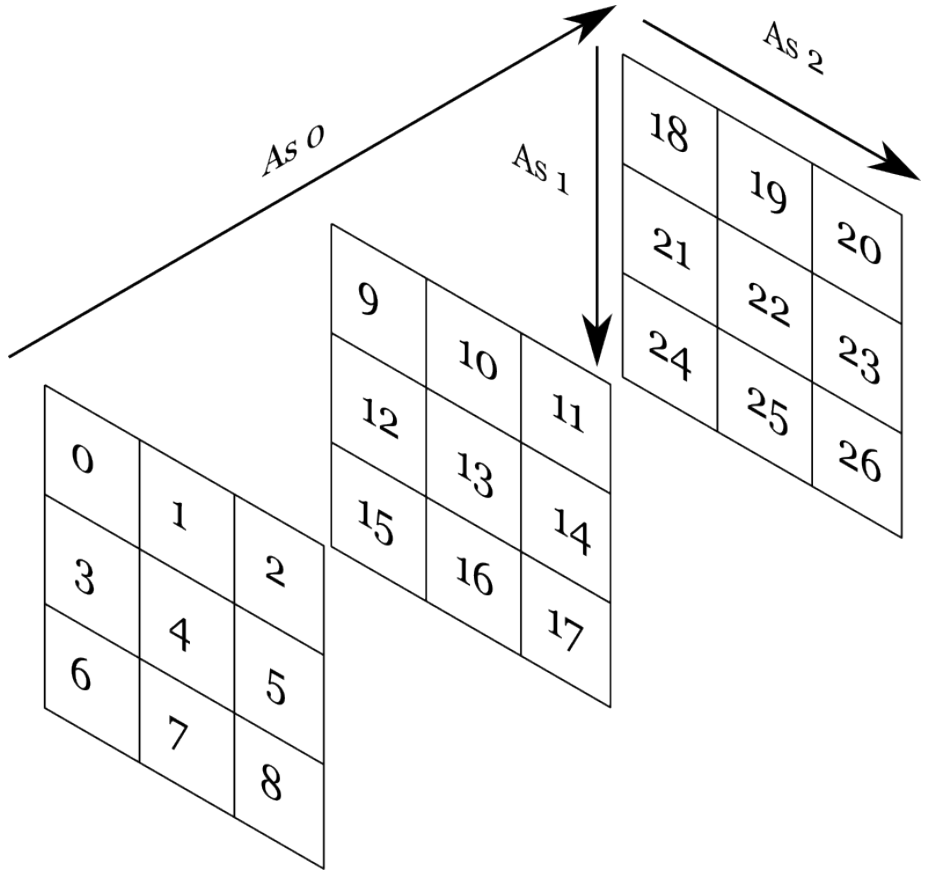
Selectie van elementen (2)

Slice zowel rij als kolom gecombineerd:

```
>>> A = np.zeros( (5, 5) )
>>> m = np.zeros(A.shape, dtype=np.bool8)
>>> m[2,:] = True
>>> m[:,2] = True
>>> A[m] = 999
>>> A
array([[   0.,    0.,  999.,    0.,    0.],
       [   0.,    0.,  999.,    0.,    0.],
       [ 999.,  999.,  999.,  999.,  999.],
       [   0.,    0.,  999.,    0.,    0.],
       [   0.,    0.,  999.,    0.,    0.]])
>>> A[m]
array([ 999.,  999.,  999.,  999.,  999.,  999.,  999.,  999.,  999.])
```

3-dimensionale arrays

- Een NumPy array met 3 dimensies is helemaal geen probleem.
- Initialisatie zoals je bent gewend.
- Vorm-tuple bevat 3 waarden.



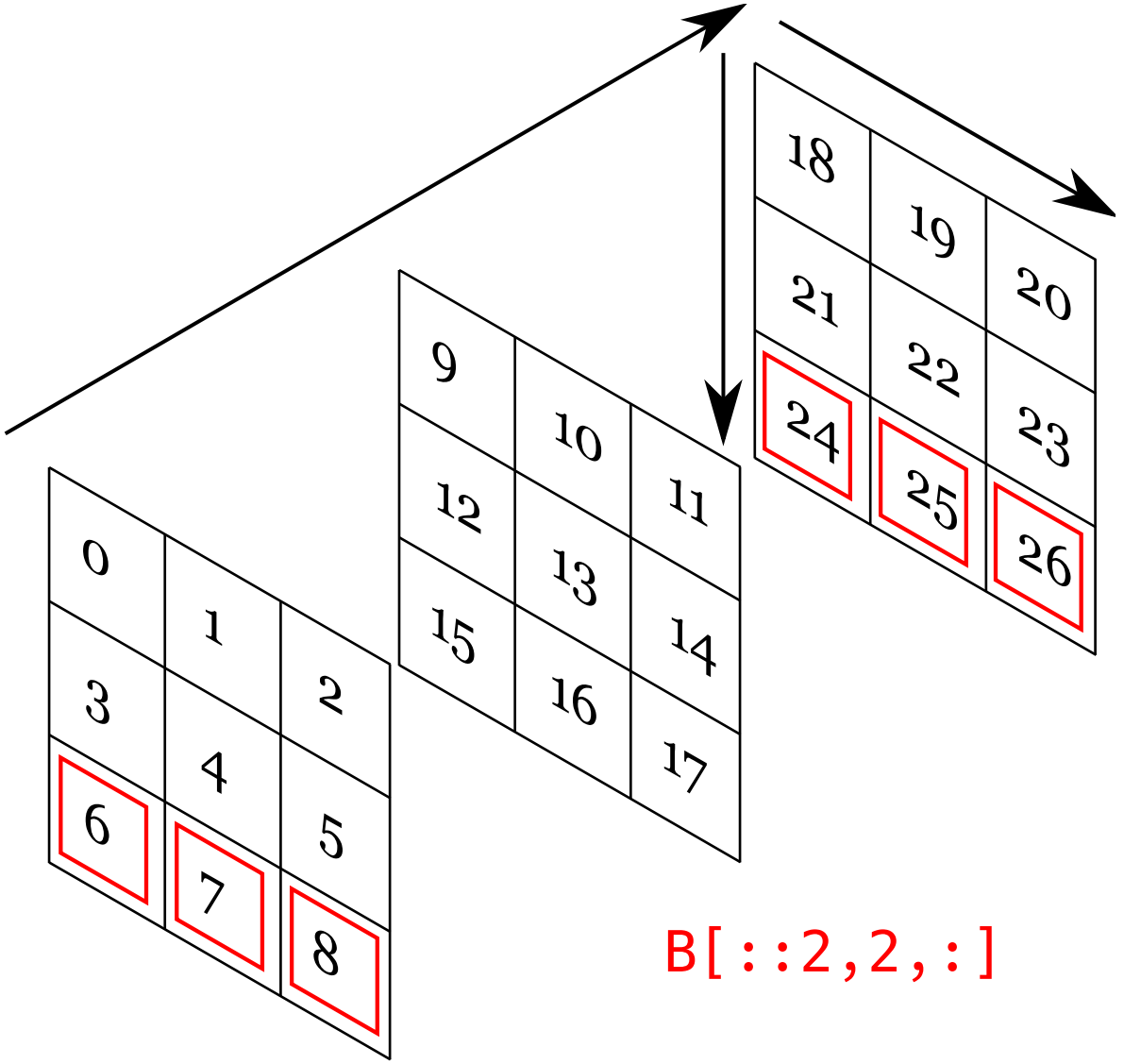
3-dimensionale arrays (2)

- Drie dimensies, heeft dat nu wel zin?
- Voorbeeld:
  - 2 vlakken: 1 voor x-coordinaten, 1 voor y-coordinaten.
  - Per vlak: N tijdstappen langs de rij-as.
  - Per vlak: M verschillende objecten (vogels?) langs de kolom-as.
  - (2, N, M)

Slicing in 3-d

```
>>> B = np.arange(27).reshape( (3,3,3) )
# Kies alleen "voorste" vlak; B[0] is equivalent.
>>> B[0,:,:]
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
# Kies uit het voorste vlak de derde kolom.
>>> B[0,:,2]
array([2, 5, 8])
# Uit vlakken 0, 2, ... kies de derde rij.
>>> B[::2,2,:]
array([[ 6,  7,  8],
       [24, 25, 26]])
```

Slicing (2)




```
B[::2,2,:]
```

Slicing (3)

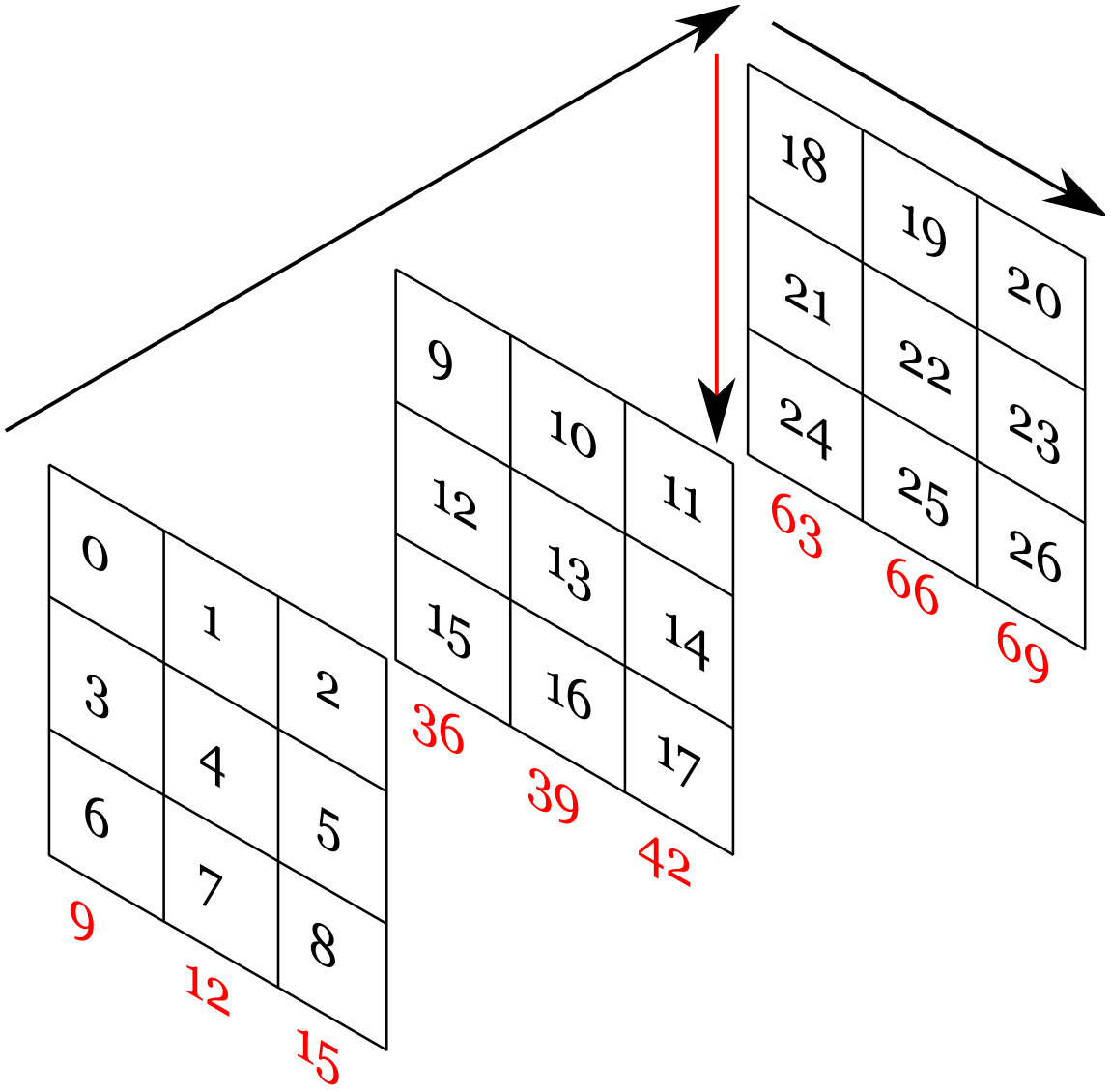
```
# Uit alle vlakken, selecteer rij/kolom 0, 2, ...
>> B[:,::2,::2]
array([[[ 0,  2],
        [ 6,  8]],

       [[ 9, 11],
        [15, 17]],

       [[18, 20],
        [24, 26]]])
```

Reductie in 3-d

Reductie-operatoren werken ook op 3-dimensionale arrays, er kan weer een as worden opgegeven.

```
>>> B = np.arange(27).reshape( (3,3,3) )
# Sommeer elke kolom (dus langs de rij-as)
>>> B.sum(axis=1)
array([[ 9, 12, 15],
       [36, 39, 42],
       [63, 66, 69]])
```

Reductie in 3-d (2)



Iteratietechnieken

Stel we willen een lijst aflopen en hebben in de loop body zowel een index als lijst-element nodig.

```
for i in range(len(lijst)):
    print i, "-", lijst[i]

i = 0
for l in lijst:
    print i, "-", l
    i += 1

# Nog mooier
for i, l in enumerate(lijst):
    print i, "-", l
```

Iteratietechnieken (2)

Itereren over een lijst van tuples:

```
lijst = [(1, 'a'), (2, 'b'), (3, 'c')]
for getal, letter in lijst:
    print getal, ",", letter
```

Als je twee aparte lijsten hebt (dezelfde lengte) kun je deze samenvoegen met `zip`:

```
getallen = [1, 2, 3]
letters = ['a', 'b', 'c']
for g, l in zip(getallen, letters):
    print g, ",", l

horz = range(10, 20, 2)
vert = range(13, 23, 2)
for x, y in zip(horz, vert):
    print "({}, {})".format(x, y)
```

Iteratietechnieken (3)

```
lijst = [4, 13, 2, 8, 11, 5]
for l in reversed(lijst):
    print l,
# Geeft: 5 11 8 2 13 4
for l in sorted(lijst):
    print l,
# Geeft: 2 4 5 8 11 13
for l in reversed(sorted(lijst)):
    print l,
# Geeft: 13 11 8 5 4 2
```

Iteratietechnieken (4)

Hoe werken we eenvoudig met data in een dictionary?

```
voorraad = { "peren": 2, "appels": 8,
   "tomaten": 0, "witte bonen": 101 }

for k in sorted(voorraad.keys()):
    print k,

for v in voorraad.values():
    print v,

for k, v in voorraad.items():
    print "Er zijn {0} stuks {1}.".format(v, k)
```

Module showcase

De Python bibliotheek is al zeer uitgebreid.

Documentatie online: https://docs.python.org/2/library/index.html

De documentatie is meestal als volgt gestructureerd:

- Omschrijving inhoud en doel module.
- Omschrijving alle klassen en functies in de module. Uitleg werking en parameters functies.
- Aan het einde vind je vaak enkele voorbeelden.

Modules uit de standaardbiliotheek

- re - regular expressions
- datetime & calendar
- decimal & fraction
- zipfile & tarfile
- SQL DB toegang
- Internet modules: e-mail, HTTP, FTP, ...
- UNIX / Mac / Windows specifieke modules
- En nog veel meer ...

CSV module

- CSV: Comma Separated Values.
- Wordt ondersteund door elk spreadsheet-programma.
- Python module voor inlezen/wegschrijven.

```
import csv
f = open("data.csv", "r")
csvreader = csv.reader(f)
for row in csvreader:
    # elke row is een Python list
    print row
f.close()
```

SciPy

- Bouwt voort op NumPy: meer science & mathematics functionaliteiten.
- Constanten (natuurkundige/sterrenkundige).
- I/O: MATLAB matrices, IDL, wave files, sparse matrices.
- Lineaire algebra.
- Fourier Transforms.
- Integratie & differentiaal vergelijkingen.
- Etc…

Externe Packages

- Er zijn nog veel meer packages dan alleen de packages die standaard met Python worden meegeleverd.
- Deze slides zijn gegenereerd met een Python script!!
- Hoe externe packages installeren?
  - Linux: liefst via Linux distributie, anders "pip".
  - Mac: of MacPorts, of Python distributie, of "pip".
  - Windows: via Python distributie.
  - (Zie ook het dictaat voor links)

Packages zoeken

- PyPI: Python Package Index.
  - https://pypi.python.org/pypi
- Of Google …

Natuur & Sterrenkunde

- Astropy
  - Astronomische coordinaten
  - Model fitting
  - Convolution
  - Cosmologische modellen …
- Verschillende natuurkundige modules
  - ElectromagneticPython
  - gwpy - gravitational wave astrophysics
  - PyFeyn - Feynman diagrammen tekenen
  - SunPy - Solar Physics

Pandas

- Data manipulatie & analyse.
- Lijkt op meer op een "spreadsheet" vergeleken met NumPy.
  - (De docent gebruikt het voor het automatisch verwerken van cijfers)
- Kan direct CSV inlezen, begrijpt headers.
- Tegenhanger van "R".

(Korte demo)

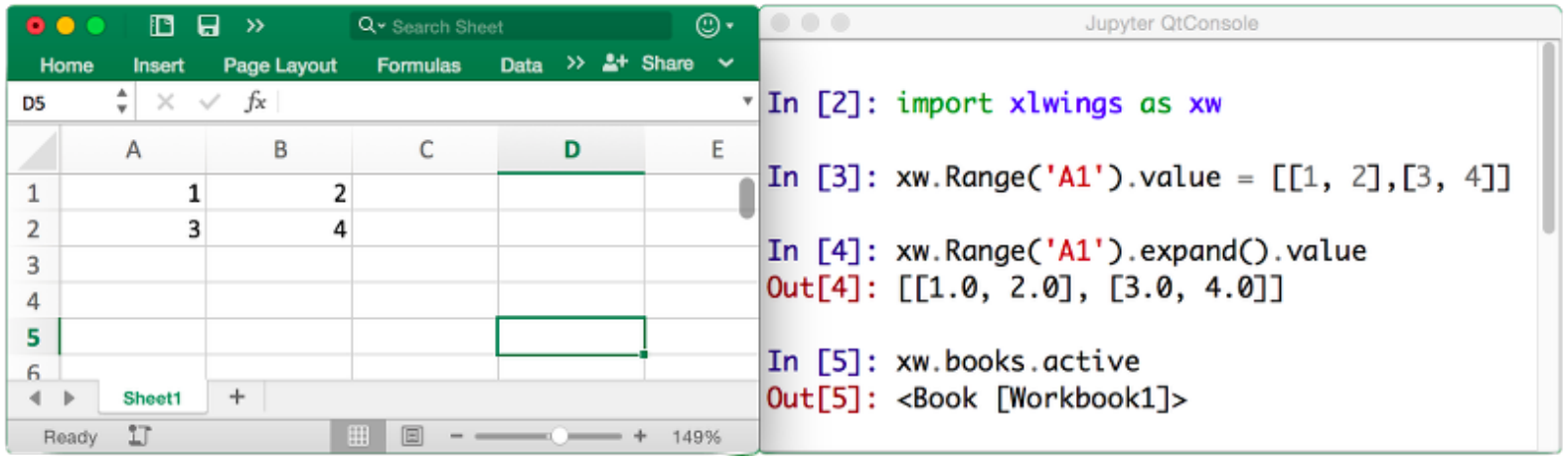
Excel

Verschillende modules om te werken met Excel files:

- xlrd
- xlsxwriter
- xlutils

Of Excel spreadsheets manipuleren vanuit Python!

- xlwings



Interactieve programma's

Vraag van vorig jaar:

- Hoe pijltjestoetsen gebruiken?
- Tegelijkertijd wachten op invoer en iets berekenen / tekenen.

In principe wil je in dit geval meer controle over het terminalvenster.

De standaard library om dit te doen heet 'ncurses' en daar is natuurlijk een Python-binding voor.

(Snake demo)

Grafische programma's

Maar wat als we echte grafische programma's willen schrijven?

- Eenvoudig tekenen: Turtle.
- Graphical User Interfaces: GUIs
  - "TkInter", ingebouwd in Python maar ziet er niet fantastisch uit.
  - GUIs zijn lastig, omdat elk systeem een eigen "smaak" grafische interface heeft.
- 3D? Dan OpenGL.

(Demos)

Meer Python

De taal Python heeft nog veel meer interessante functionaliteiten:

- Bijvoorbeeld hulpmiddelen om “functioneel” te programmeren:
  - Generators
  - List comprehensions
  - Lambda functies
  - map(), reduce(), filter() functies
- Exceptions & exception handling
- Iterators
- Function decorators

Deze geavanceerde functionaliteiten worden niet besproken in het dictaat, maar zijn wel terug te lezen in de Python Tutorial: https://docs.python.org/2/tutorial/

Generators

```
def graaf_tel():
    reeks = range(1, 7)
    for getal in reeks:
        yield getal

for i in graaf_tel():
    print i
```

List comprehensions

```
x = [0 for i in range(10)]
x = [i for i in A if i < 10]

strings = map(str, [i for i in graaf_tel()])
strings = map(str, graaf_tel())


som = sum(graaf_tel())
```

Lambda & filter

```
import random, operator

# 20 random getallen
R = random.sample(range(1000), 20)

# Bepaal alle getallen kleiner dan 100
klein = filter(lamba x: x < 100, R)

# Zelfde als “sum”, maar dan expliciet
som = reduce(operator.add, klein)
# Zo kunnen we ook een “product” maken
product = reduce(operator.mul, klein)
```

Tot slot

- Werkcollege: derde programmeeropgave.
- Volgende week bespreken we het tentamen van vorig jaar (16 december 2016).
  - Tentamen is te downloaden van de website.
  - Advies: maak zelf het tentamen van te voren om te oefenen.

Programmeermethoden NA Week 10

Universiteit Leiden