# Programmeermethoden NA

# Week 10: NumPy, iPython, Python module showcase

Kristian Rietveld

http://liacs.leidenuniv.nl/~rietveldkfd/courses/prna2016/


Universiteit Leiden
The Netherlands

# `np.choice`

- `np.choice` is niet beschikbaar in oude NumPy versies ...
- Het volgende kan ook:
  - Genereer een tijdelijke matrix bestaande uit random getallen 0 t/m 99.
  - Zet alle getallen groter dan het gegeven percentage op 1. Kleiner gelijk op 0. Denk aan maskers, die we vorige week zagen.
  - Maak nu een nieuwe boolean matrix en initialiseer deze met de zojuist gemaakte matrix met enen en nullen.

# Korte herhaling OOP

Wat was nu ook alweer het idee van OOP?

- We willen zelf een object-type ontwerpen.
- In een object kunnen we data opslaan.
- Via methoden op het object kunnen we het object manipuleren.

Voorbeeld: een breuk

- We slaan teller en noemer op in het object.
- Methoden: optellen, afdrukken, vereenvoudigen, enz.

# Korte herhaling OOP (2)

```
# Maak breuk met teller=1, noemer=3
b1 = Breuk(1, 3)
b2 = Breuk(1, 4)
b1.telop(b2)
b1.drukaf()
```

# Korte herhaling OOP (3)

```
class Breuk(object):
    def __init__(self, teller, noemer):
        self.teller = teller
        self.noemer = noemer

    def geefTeller(self):
        return self.teller

    def geefNoemer(self):
        return self.noemer

    def telop(self, breuk2):
        self.teller = \
            self.teller * breuk2.geefNoemer() + \
            breuk2.geefTeller() * self.noemer
        self.noemer *= breuk2.geefNoemer()

    def drukaf(self):
        print "{}/{}".format(self.teller, self.noemer)
```

# Korte herhaling OOP (4)

Methoden in een klasse hebben altijd `self` als eerste argument.

```
class Breuk(object):
    def telop(self, breuk2):
        ...
```

Bij het aanroepen van een methode wordt het object voor de punt doorgegeven als de parameter `self` in de methode.

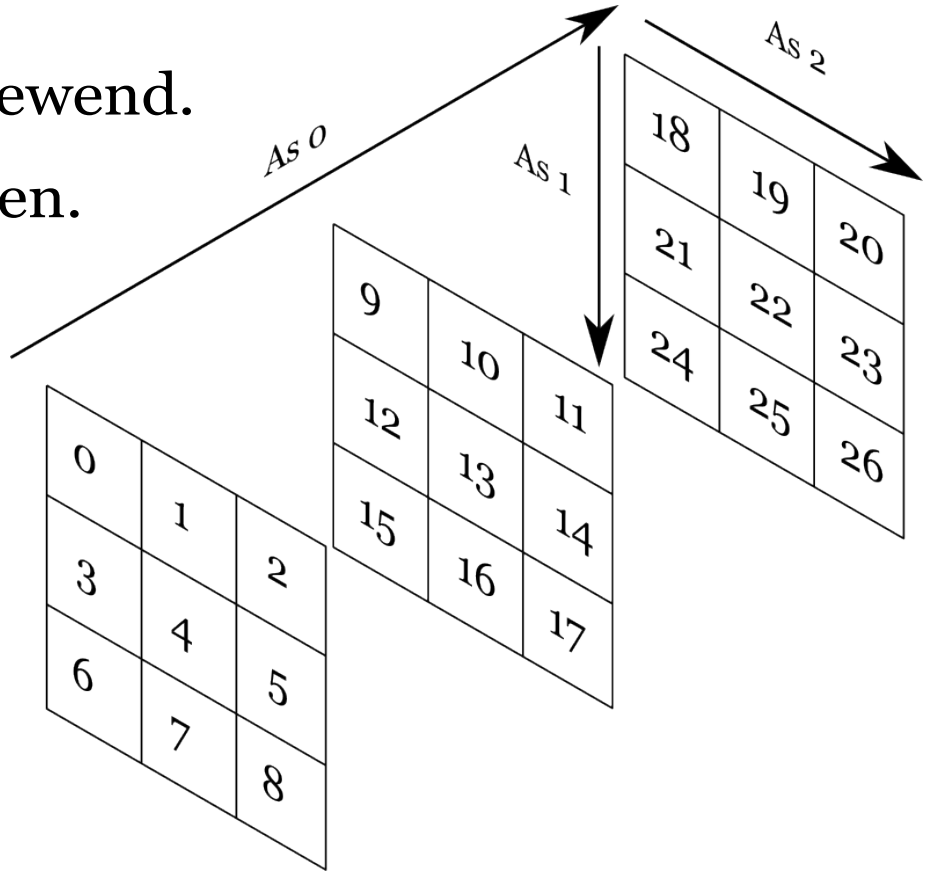
```
b1.telop(b2)
```

In de aanroep wijst `self` naar `b1`, en `b` naar `b2`.

# 3-dimensionale arrays

Dan nu verder met NumPy:

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



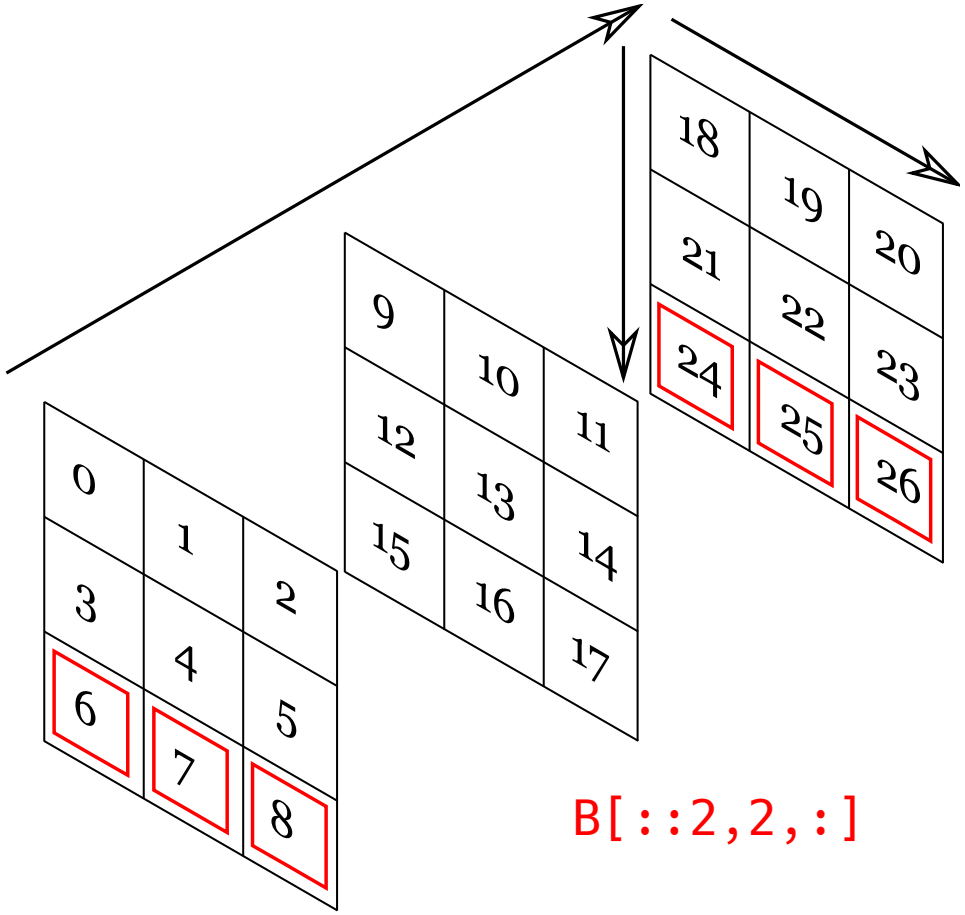
# 3-dimensionale arrays (2)

- 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 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)



# Slicing (3)

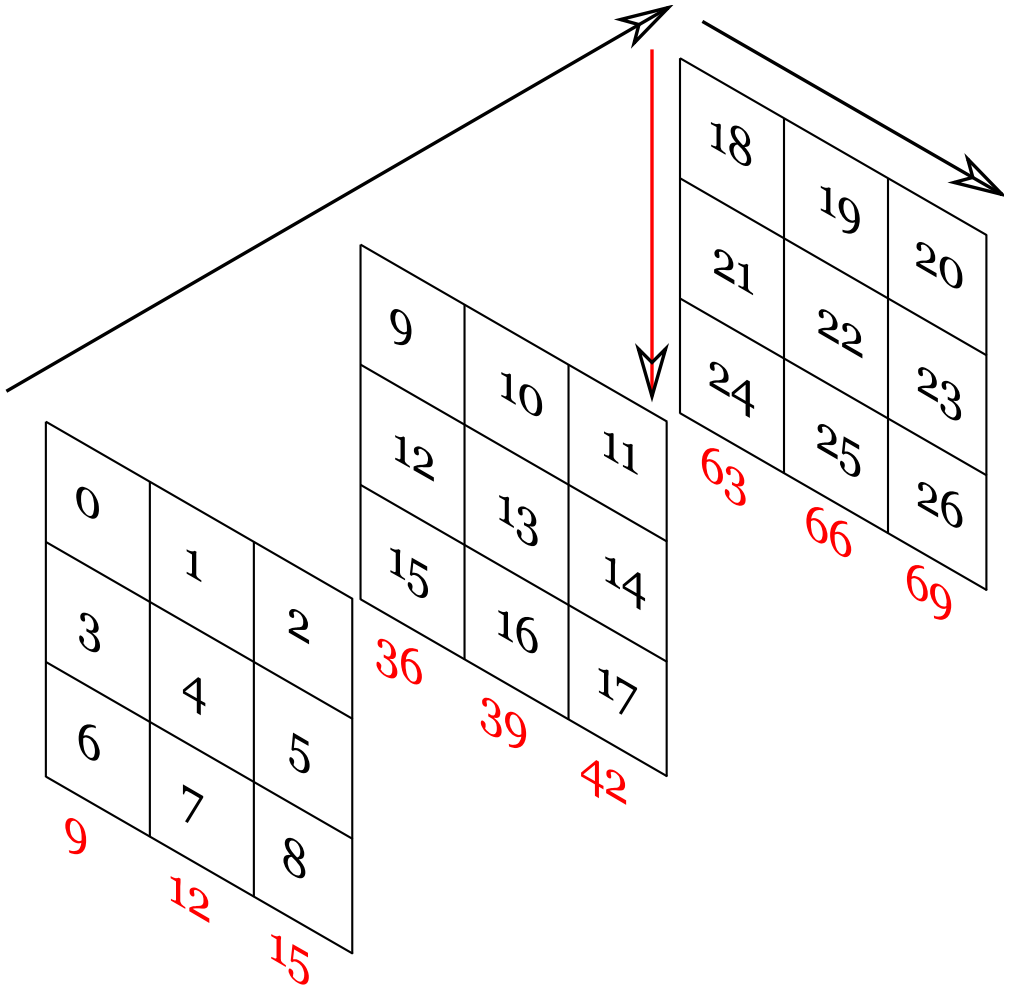
```
# Uit alle vlakken, selecteer rij/kolom 0, 2, ...
>> B[:,::2,::2]
array([[[ 0,  2],
        [ 6,  8]],

       [[ 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 (zelfde 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)
```

# iPython features

- Je kan makkelijk voorgaande resultaten hergebruiken.
- Je kan ook ls, cat, cd, etc. gebruiken.
- Tab completion (!)
- Pylab mode.
- Notebook modus.

# iPython

Hoe verkrijgen?

- *Windows*: Enthought Canopy
- *Mac*: niet standaard, je zult een Python-distributie moeten installeren of via MacPorts.
- *Ubuntu*:

```
apt-get install ipython ipython-qtconsole ipython
```

(Notebook demo)

# 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 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.
- Kan direct CSV inlezen, begrijpt headers.
- Tegenhanger van "R".

(Korte demo)

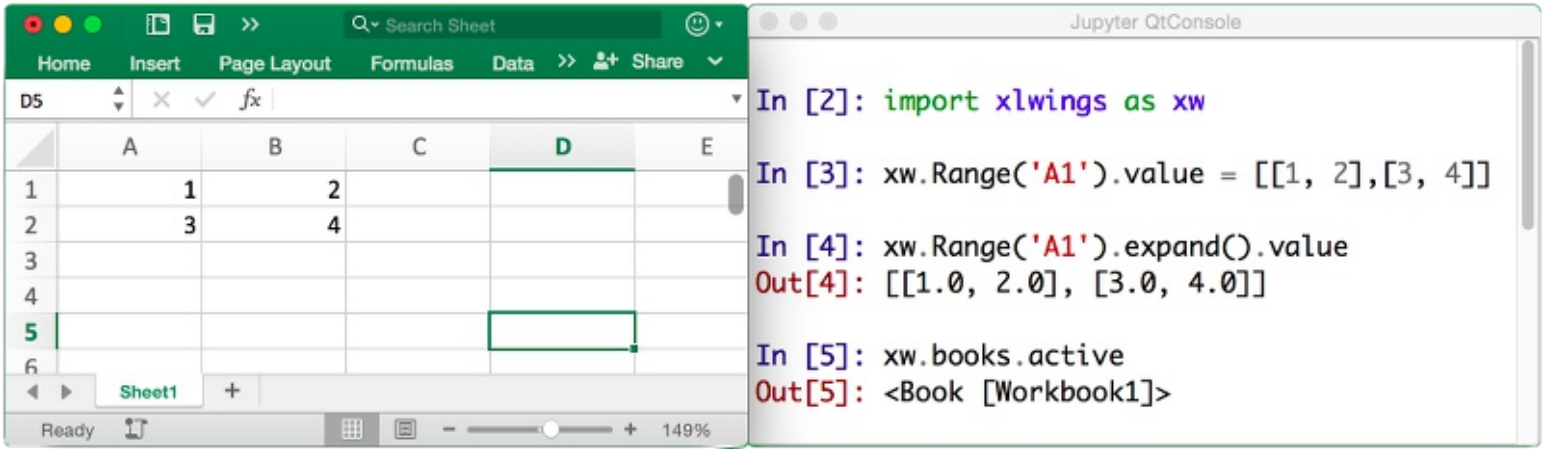
# Excel

Verschillende modules om te werken met Excel files:

- xlrd
- xlsxwriter
- xlutils

Of Excel spreadsheets manipuleren vanuit Python!

- xlwings



# Interactieve programma's

Een aantal studenten vroeg zich af:

- 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?

- Simpel 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:

- Generators
- List comprehensions
- Lambda functies
- Exceptions

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()])

som = sum(i for i in graaf_tel())
```

# Tot slot

- Werkcollege: derde programmeeropgave.
- Vragenuren: op verzoek nu donderdag en vrijdag.
- Oefententamen komt deze week online, volgende week bespreken.