Introduktion til Programmering med Videnskabelige Anvendelser

Gerth Stølting Brodal

Datalogisk Institut
Aarhus Universitet

Workshop 4, Odense, 30. april 2019

Forelæseren

Navn	Gerth Stølting Brodal
Forskning	Algoritmer og Datastrukturer (Datalogi)
Undervisning	
2018-	Bachelor : Introduktion til Programmering med Videnskabelige Anvendelser
2004-18	Bachelor : Introduktion til Algoritmer og Datastrukturer
1999-17	Kandiatkurser i Computational Geometry, Algorithm Engineering, Advanced Data Structures, External Memory Algorithms and Data Structures
Python	~~Nybegynder (2018)~~ Elementært niveau (2019)

Kursusbeskrivelse – kursuskatalog.au.dk/da/course/82756/

Introduktion til Programmering med Videnskabelige Anvendelser

Kvalifikationsbeskrivelse

Efter kurset vil deltagerne have kendskab til principper og teknikker til systematisk konstruktion af programmer.

Ved afslutningen af kurset vil deltagerne være i stand til at:

- Anvende konstruktioner fra et almindeligt programmeringssprog,
- Udvikle **velstrukturerede** programmer og udføre **test** og **fejlfinding** af disse,
- Forklare grundlæggende programmerings begreber og enkle algoritmiske teknikker,
- Anvende standard **værktøjer til videnskabelige formål**.

Indhold

Kurset giver en introduktion til programmering med videnskabelige applikationer. Programmerings begreber og teknikker introduceres ved hjælp af programmeringssproget **Python**. Begreberne vil desuden blive **illustreret i andre programmeringssprog**. Kursusinholdet inkluderer følgende.

Grundlæggende programmeringsbegreber: Datatyper, operatorer, variable, kontrolstruktur, betingelser, løkker, funktioner, rekursion, scope, undtagelser. *Objekt orienteret programming*: Abstracte datatyper, klasser, nedarvning, indkapsling. *Grundlæggende algoritmiske teknikker*: Sortering, binær søgning, dynamisk programmering. *Systematisk udvikling af programmer*: Test og fejlfinding. Fil-baseret input/output, numerisk analyse, funktionel programmering. Scientific computing ved hjælp af standard pakker i Python.

ECTS 10

Timer Forelæsninger: 2 x 2 timer/uge
Øvelser: 1 x 3 timer/uge
Studiecafé

Undervisningssprog
Dansk

Forelæser
Gerth Stølting Brodal

Faglige forudsætninger
(Noget) Lineær algebra

Eksamen
2 timer multiple-choice
Hjælpemidler: Ingen
7-trinsskala

Forudsætninger for prøvedeltagelse
Aflevering og godkendelse af 10 obligatoriske opgaver og **1 projekt**

Note Karakteren afspejler en samlet vurdering af implementeringsprojekt og multiple-choice eksamen

Baggrund for kurset

- Gennemgribende studiereform ved Science and Technology @ AU, 2017
  - 4 kvarterer om året erstattets med 2 semestre (5 ECTS → 10 ECTS kurser)
- Tidligere har mange ikke-datalogi studerende været pålagt at følge datalogi kurset “Introduktion til Programmering (1 + 2)” i Java
  - blev udtrykt ønske fra andre uddannelser om et mere målrettet kursus
- Sammen med institut for matematik (Niels Lauritzen) defineret et nyt programmeringskursus målrettet til matematik studerende
  - Python
  - Noget projektarbejde / matematiske anvendelser
  - Dynamisk programmering (+ grundlæggende sortering og binær søgning)
  - Grundlæggende forståelse for ligheder / forskelle mellem Python og Java

Kurset i praksis (ikke faglige aspekter)

- Kurset kørte første gang i foråret 2018
  - 2 måneder før kursusstart sagde den planlagte kursusansvarlige op
  - Med kort varsel blev jeg kursusansvarlig – og kendte *intet* til Python
- Kursets primære deltagere (foråret 2019)
  - Matematik
  - Matematik-økonomi
  - Kemi
  - Sidefag i matematik (tyiske ikke Science)

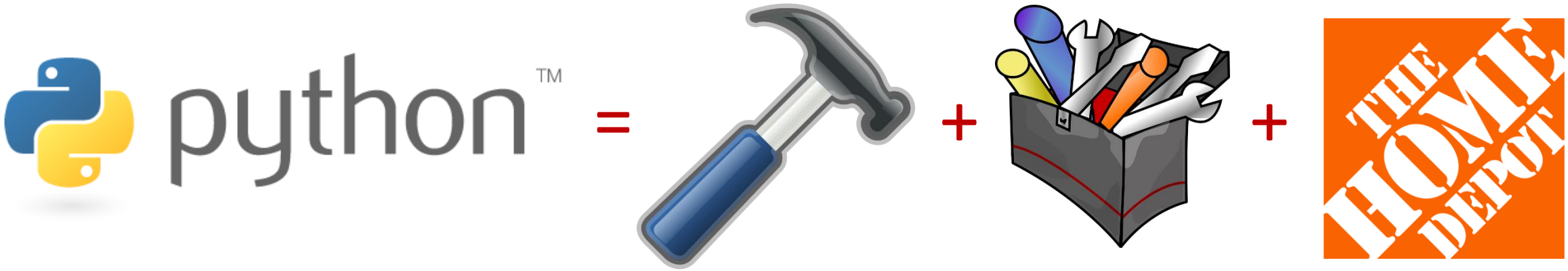
  At det ikke kun var matematikstuderende der fulgte kurset, blev først opdaget en uge inden kurset startede første gang
- Kursusindehold
  - En ting er den officielle kursusbeskrivelse ... noget andet er virkeligheden når forelæseren selv kan bestemme indhold og eksamen

Deltagere 2019			
Matematik	58	Historie	2
Matematik-økonomi	33	Idéhistorie	2
Kemi	18	Kognitionsvidenskab	1
Fysik	8	Kunsthistorie	1
Enkeltfag	7	Medicinalkemi	1
Musikvidenskab	7	Nano	2
Idræt	5	Nordisk sprog og litteratur	1
Mekanik	3	Spansk	1
Filosofi	2		

Mit personlige mål...

At de studerende ved kursusafslutningen kan

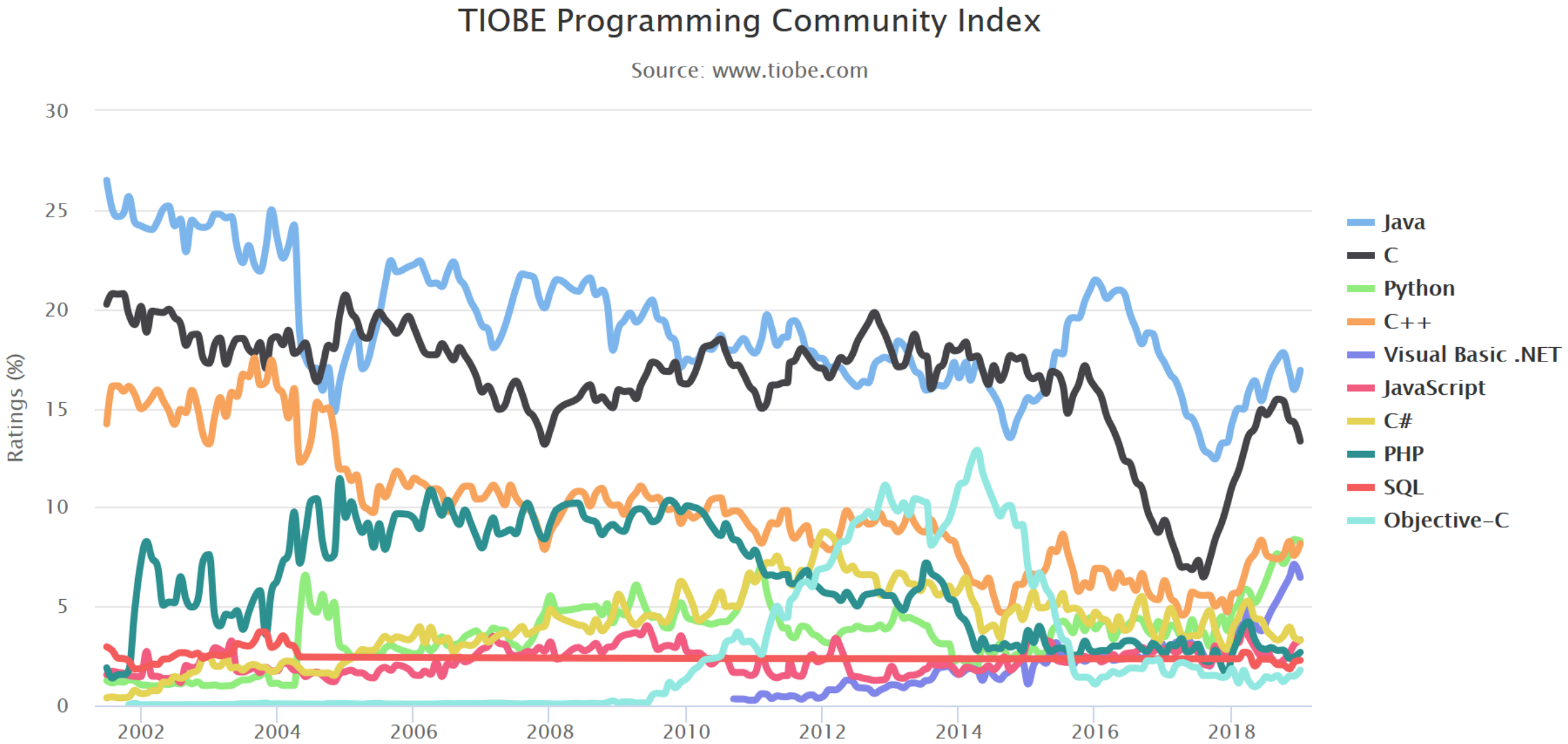
- mestre grundlæggende programmeringsbegreber
- kender og kan bruge mere avancered programmeringsfunktionaliteter (rekursive funktioner & data typer, OO, λ-udtryk, dekoratorer)
- Har basal kendskab til forskellige Python pakker
  - numpy, matplotlib, pandas, tkinter, scipy, Jupyter, doctest, ...
- være i stand til at navigere i Python økosystemet



Kursusevaluering 2018

”Den første forelæsning var meget ***skræmmende*** *og overvældende”*

Popularitet af programmeringssprog



Beregn $1 + 2 + \cdots + n = n \cdot (n + 1) / 2$

n	C (gcc 6.4)	C++, int (g++ 6.4)	C++, long (g++ 6.4)	Java (1.8)	Python (3.6.4)	PyPy (3.5.3)
10^7	0.13 sek*	0.18 sek*	0.15 sek	0.35 sek*	1.3 sek	0.35 sek
10^9	0.25 sek**	0.29 sek**	0.44 sek	0.86 sek**	191 sek	35 sek

Forkert svar (overløb)

* -2004260032 istedet for 50000005000000

** -243309312 istedet for 5000000000500000000

Have fun debugging!

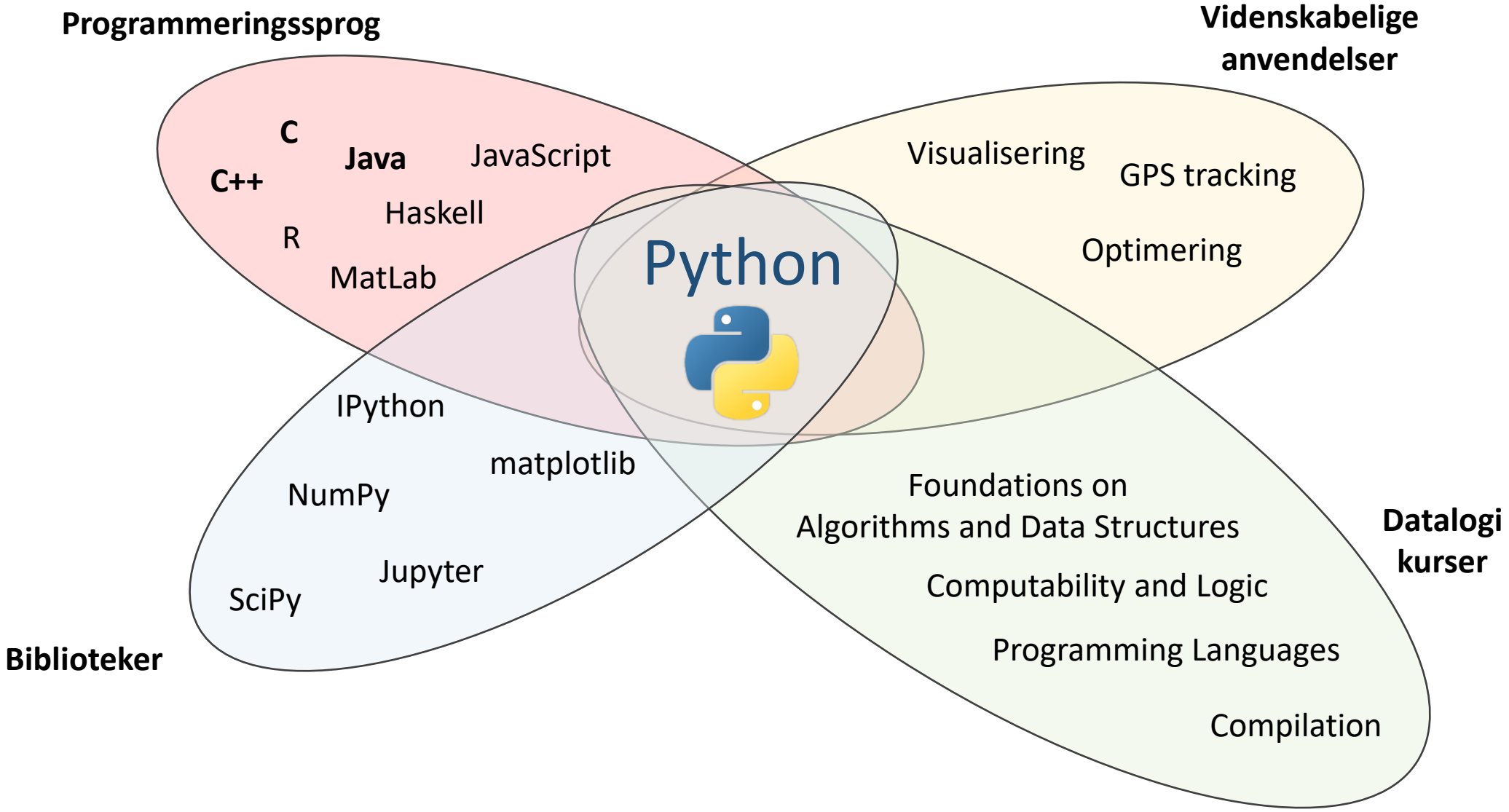
- Relativ hastighed

C ≈ C++ > Java >> Python

- C, C++, Java skal man være omhyggelig med at håndtere overløb –vælg repræsentation af heltal omhyggeligt med tilstrækkeligt antal bits (8, 16, 32, 64, 128)

Try Google: civilization gandhi overflow

Kursets indhold



Kursusindhold

Grundlæggende programmering
Advanceret / specifik Python
Biblioteker & anvendelser

1. Introduction to Python	10. Functions as objects	19. Linear programming
2. Python basics / if	11. Object oriented programming	20. Generators, iterators, with
3. Basic operations	12. Class hierarchies	21. Modules and packages
4. Lists / while / for	13. Exceptions and files	22. Working with text
5. Tuples / comprehensions	14. Doc, testing, debugging	23. Relational data
6. Dictionaries and sets	15. Decorators	24. Clustering
7. Functions	16. Dynamic programming	25. Graphical user interfaces (GUI)
8. Recursion	17. Visualization and optimization	26. Java vs Python
9. Recursion and Iteration	18. Multi-dimensional data	27. Final lecture

27 forelæsninger (2 x 45 min) + 14 øvelser (3 timer) + 3 timer studiecafé / uge + 10 afleveringer + PeerWise + MentiMeter + 1 projekt (1 måned, 25% af karakteren) + MCQ eksamen (75%, 2 timer)

https://blackboard.au.dk/webapps/blackboard/content/listContentEditable.jsp?content_id=_2092252_1&course_id=_124137_1

Binomial Coefficient
Dynamic programming using decorator

Slide fra forelæsning om dynamisk programmering

- Use a decorator (@memoize) that implements the functionality of remembering the results of previous function calls

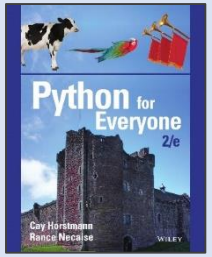
bionomial_decorator.py

```
def memoize(f):
    # answers[args] = f(*args)
    answers = {}
    def wrapper(*args):
        if args not in answers:
            answers[args] = f(*args)
        return answers[args]
    return wrapper
```

```
@memoize
def binomial(n, k):
    if k==0 or k==n:
        return 1
    else:
        return binomial(n-1, k) + binomial(n-1, k-1)
```

www.python-course.eu/python3_memoization.php

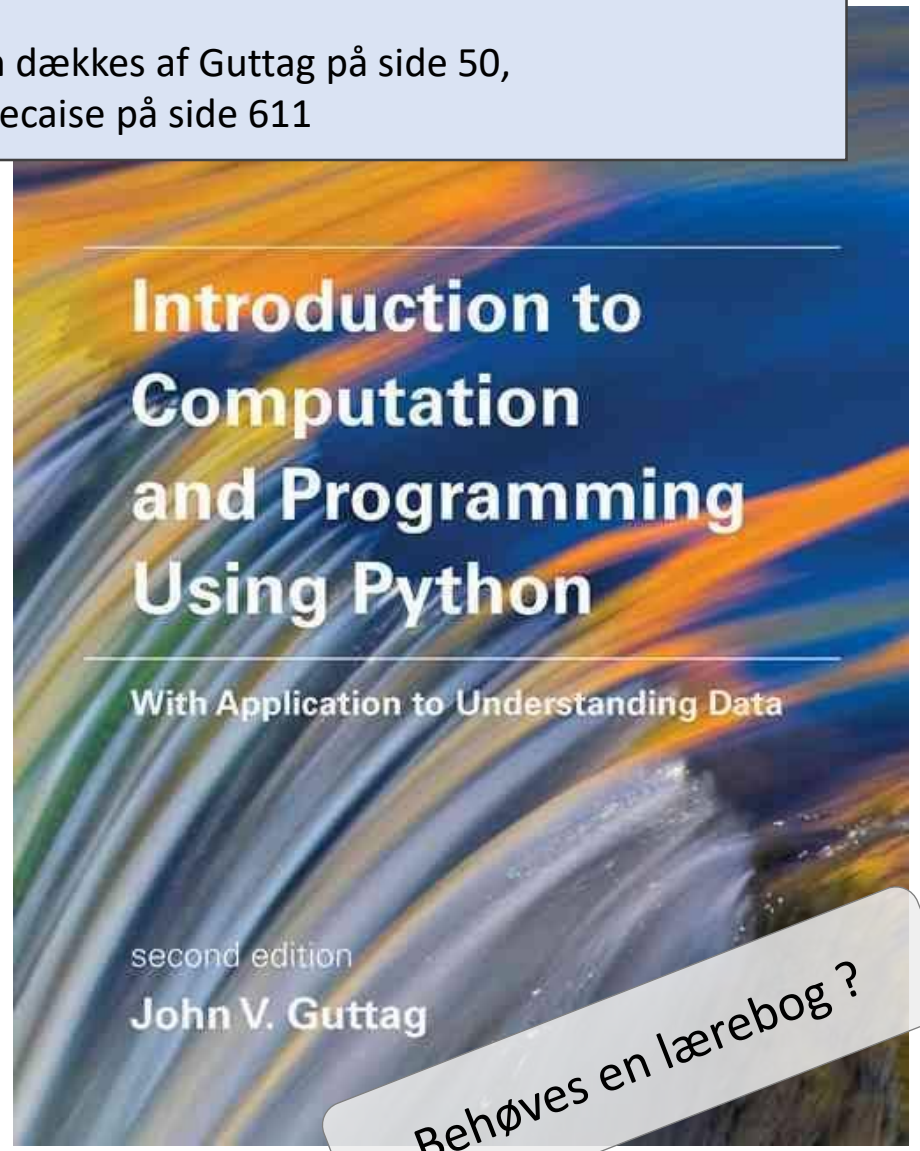
Lærebog



Sammenligning med en standard lærebog om *programmerings-sproget Python* af Cay Horstmann og Rance Necaise:

Emnet **rekursion** dækkes af Guttag på side 50, Horstmann og Necaise på side 611

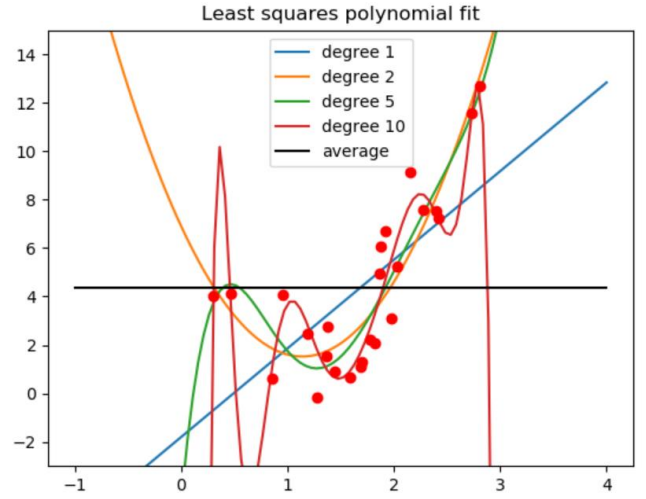
- Lærebogen følges en smule i tilfældig rækkefølge – og dækkes kun delvist – men giver en god introduction til de mest vigtige Python begreber på få sider med mange (måske for) matematiske orienterede eksempler
- Primær undervisningsmateriale er slides (opdateres i sidste øjeblik...)
- En central kompetence de studerende skal tilegne sig er at finde relevant information med Google (f.eks. Python biblioteker, stackoverflow.com, ...)

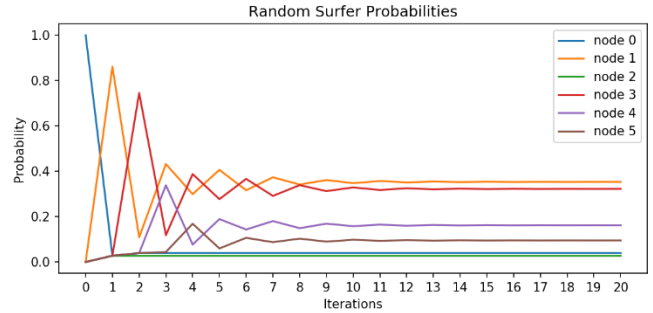


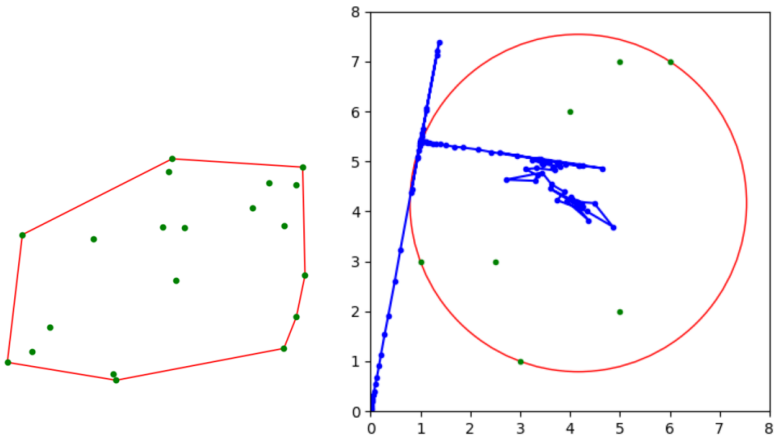
Eksempler

Matematiske og videnskabelige begreber i kursus

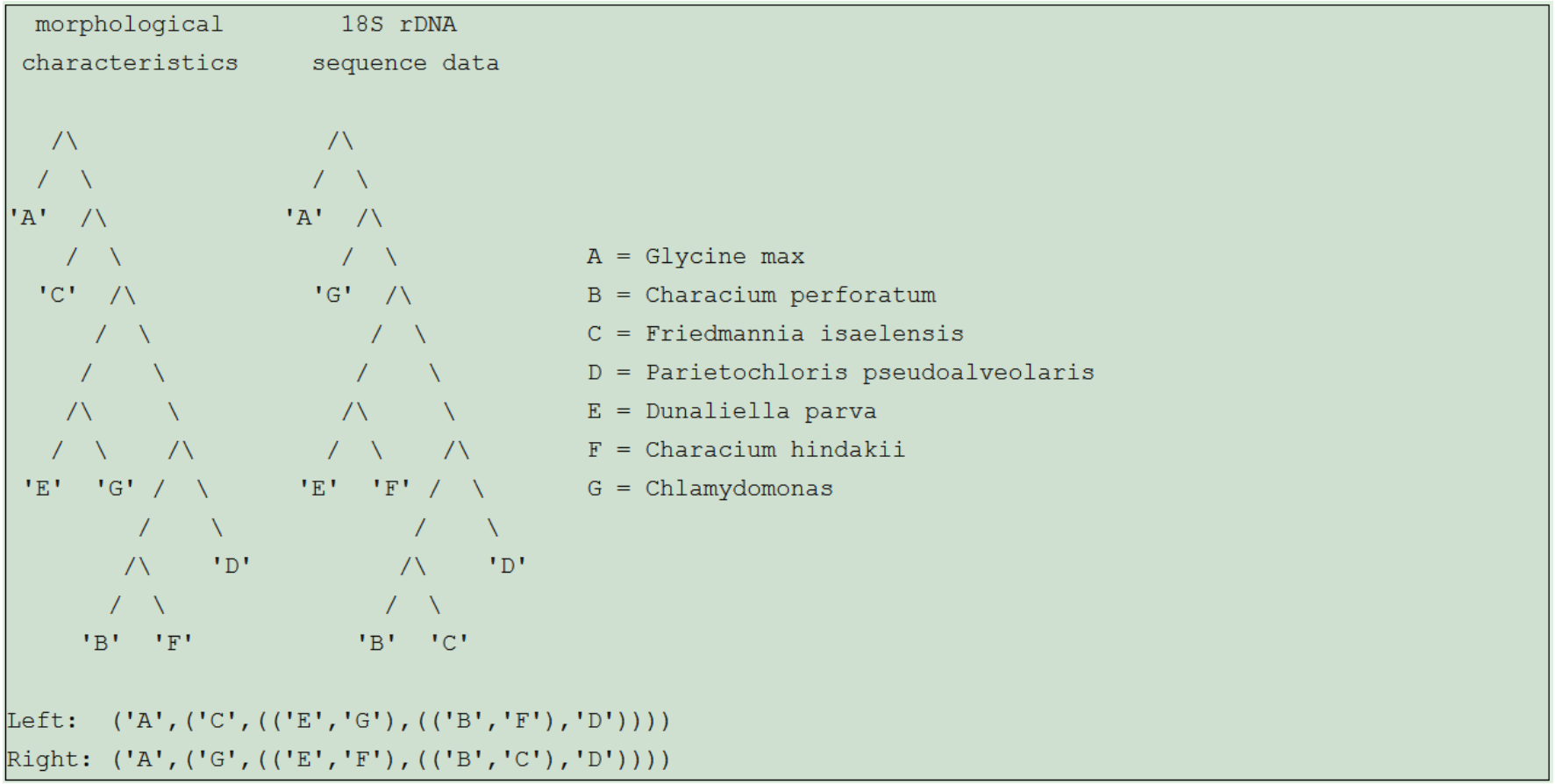
- Rekursion tilbagevendende tema (rekursive funktioner, rekursive datatyper, rekursive objekter, rekursive OO metodekald, aflevering om phylogenetiske træer)
- Dynamisk programmering (rekursion + dekorator) og rekursionsligninger
- Plot af data (matplotlib.pyplot)
- Matricer og multidimensional data (numpy)
- Mindste kradraters metode (numpy.polyfit)
- Lineær programmering (scipy.optimize.linprog)
- Maksimale strømninger (scipy.optimize.linprog)
- Eigenvektor, PageRank (numpy.linalg.eig)
- Minimum of funktioner (scipy.optimize.minimize)
  - minimum omsluttende cirkel, sammenligning med Matlab
- Jupyter notebooks







Afleveringsopgave 3 + 4 (efter ca. 1 måned)



- Rekursion & rekursive data typer
- Ikke trivielle sprog konstruktioner (tuples, lister, list comprehension)

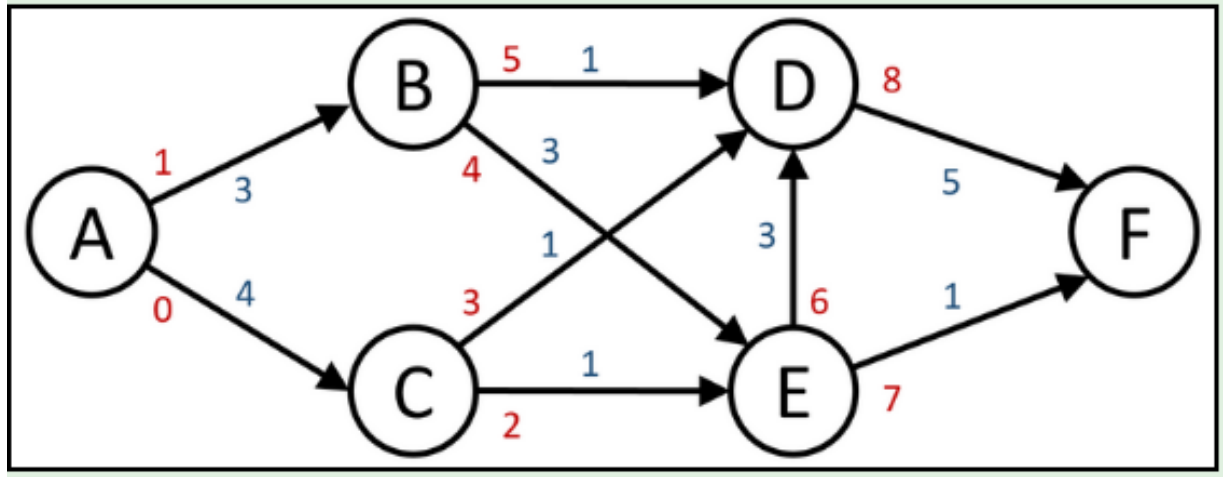
triplet_distance_compact.py - C:\Users\au121\Desktop\ipsa19\code\exercises\06_dictionaries\triplet_distance_compact.py (3.7.2)

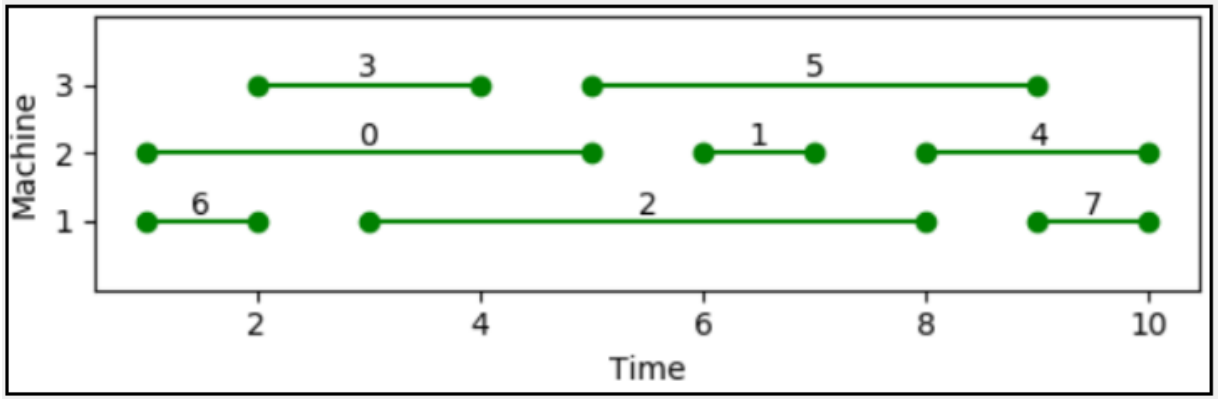
File Edit Format Run Options Window Help

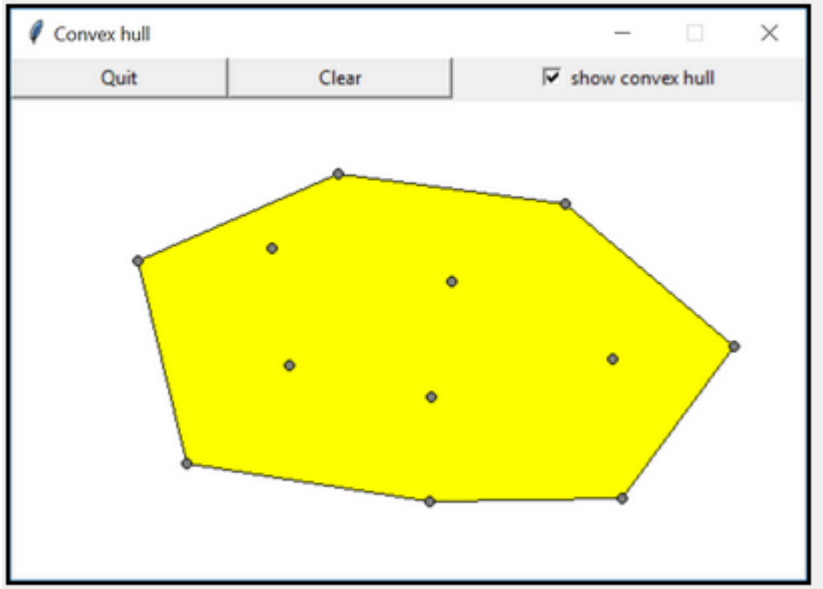
```
def triplet_distance(tree1, tree2):
    def compute(tree):
        if not isinstance(tree, tuple): return [tree], []
        (ll, lt), (rl, rt) = [compute(c) for c in tree]
        t = [(x,(y,z)) for a,b in [(ll,rl), (rl,ll)] for y in b for z in b if y < z for x in a]
        return ll + rl, lt + rt + t

    (L1, T1), (L2, T2) = compute(tree1), compute(tree2)
    n = len(L1)
    return n * (n-1) * (n-2) // 3 // 2 - len(set(T1) & set(T2))

T1 = ((('A', 'F'), 'B'), ('D', ('C', 'E')))
T2 = ((('D', 'A'), 'B'), 'F'), ('C', 'E'))
print("Triplet distance: ", triplet_distance(T1, T2))
```

Ln: 1 Col: 35

Andre afleveringer



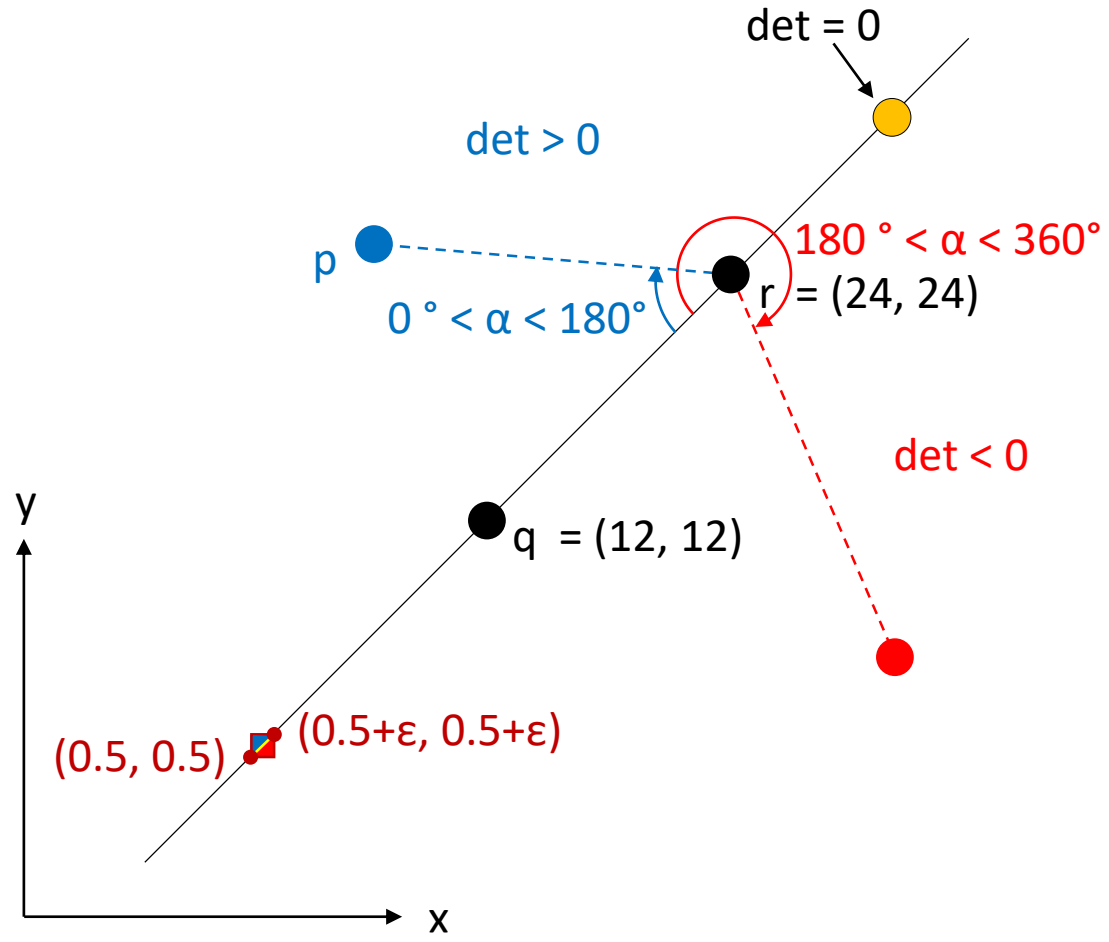




Geometric orientation test

Purpose of example

- illustrate tuples
- list comprehension
- matplotlib.pyplot
- floats are strange 

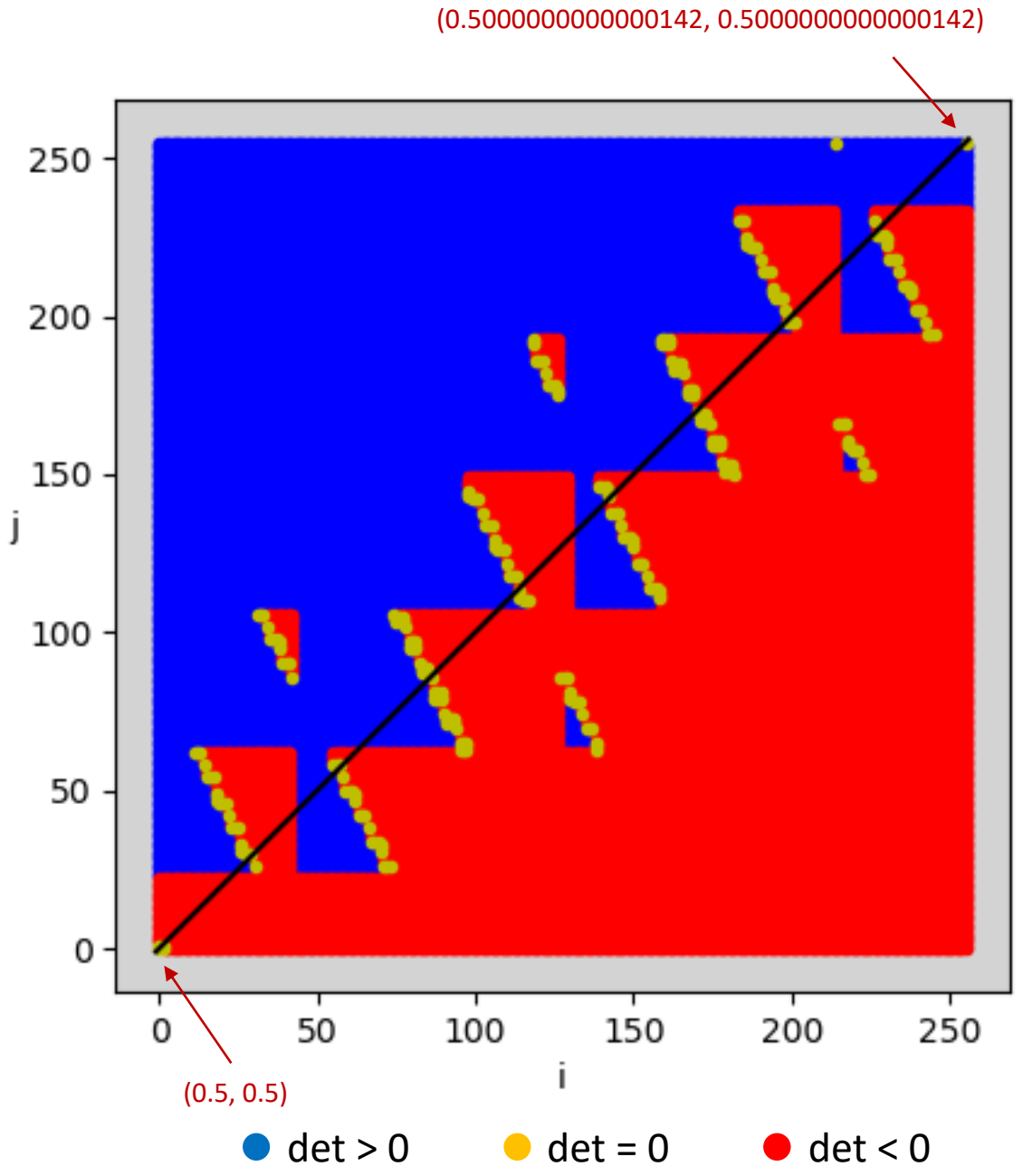


$$\det = \begin{vmatrix} 1 & q_x & q_y \\ 1 & r_x & r_y \\ 1 & p_x & p_y \end{vmatrix} = \underbrace{r_x p_y - p_x r_y - q_x p_y + p_x q_y + q_x r_y - r_x q_y}_{6\,! = 720 \text{ different orders to add}}$$

Kettner, Mehlhorn, Pion, Schirra, Yap:
[Classroom Examples of Robustness Problems in Geometric Computations]

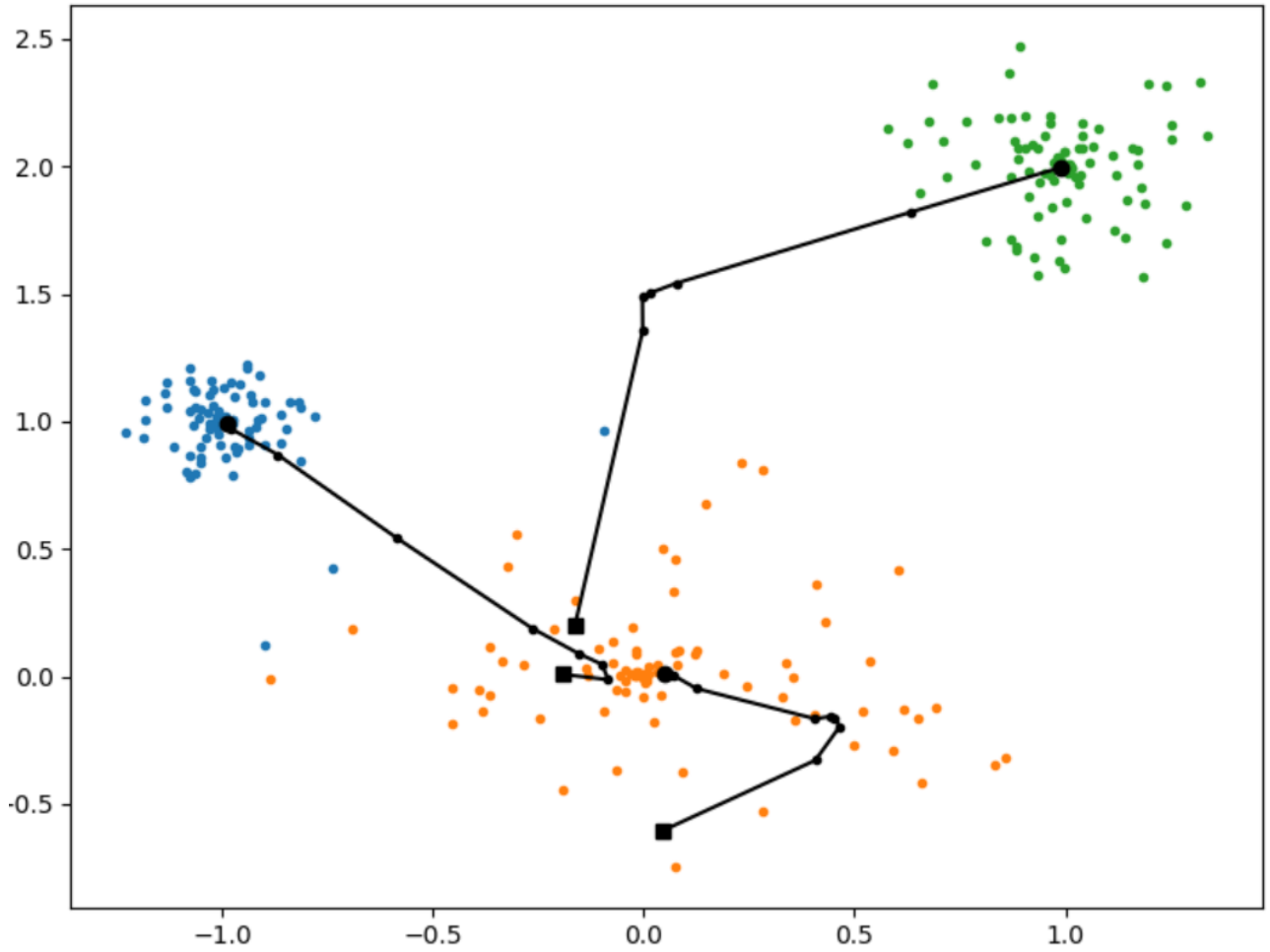
sign-plot.py

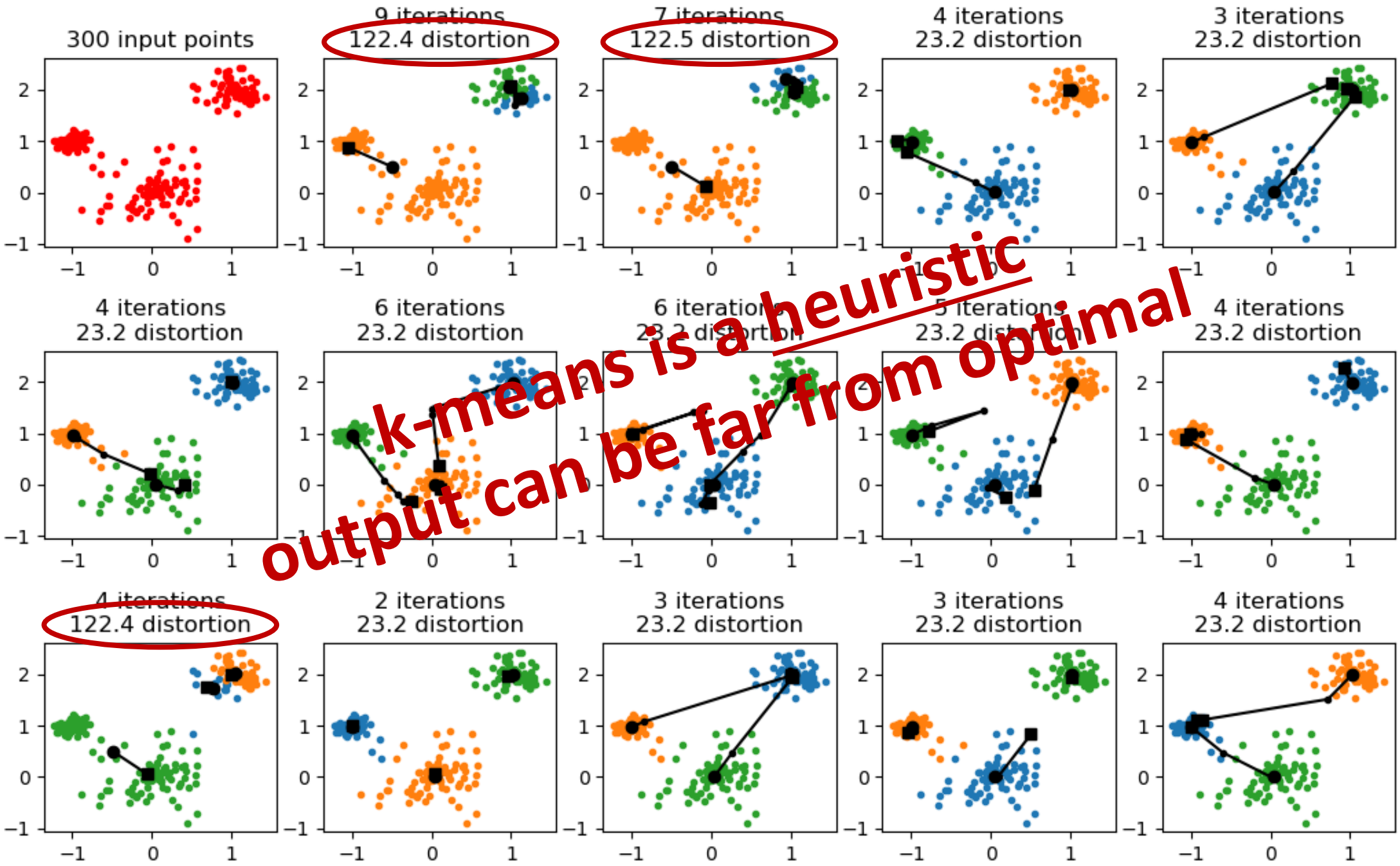
```
import matplotlib.pyplot as plt

N = 256
delta = 1 / 2**54
q = (12, 12)
r = (24, 24)
P = []  # points (i, j, det)

for i in range(N):
    for j in range(N):
        p = (1/2 + i * delta, 1/2 + j * delta)
        det = (q[0]*r[1] + r[0]*p[1] + p[0]*q[1]
             - r[0]*q[1] - p[0]*r[1] - q[0]*p[1])
        P.append((i, j, det))

pos = [(i, j) for i, j, det in P if det > 0]
neg = [(i, j) for i, j, det in P if det < 0]
zero = [(i, j) for i, j, det in P if det == 0]

plt.subplot(facecolor='lightgrey', aspect='equal')
plt.xlabel('i')
plt.ylabel('j', rotation=0)

for points, color in [(pos, "b"), (neg, "r"), (zero, "y")]:
    X = [x for x, y in points]
    Y = [y for x, y in points]
    plt.plot(X, Y, color + ".")

plt.plot([-1, N], [-1, N], "k-")
plt.show()
```



k-means - Lloyd's method (pseudo code)

```
centroids = k distinct random input points
while centroids change:
    create clusters C by assigning points to the nearest centroid
    centroids = average of each cluster
```



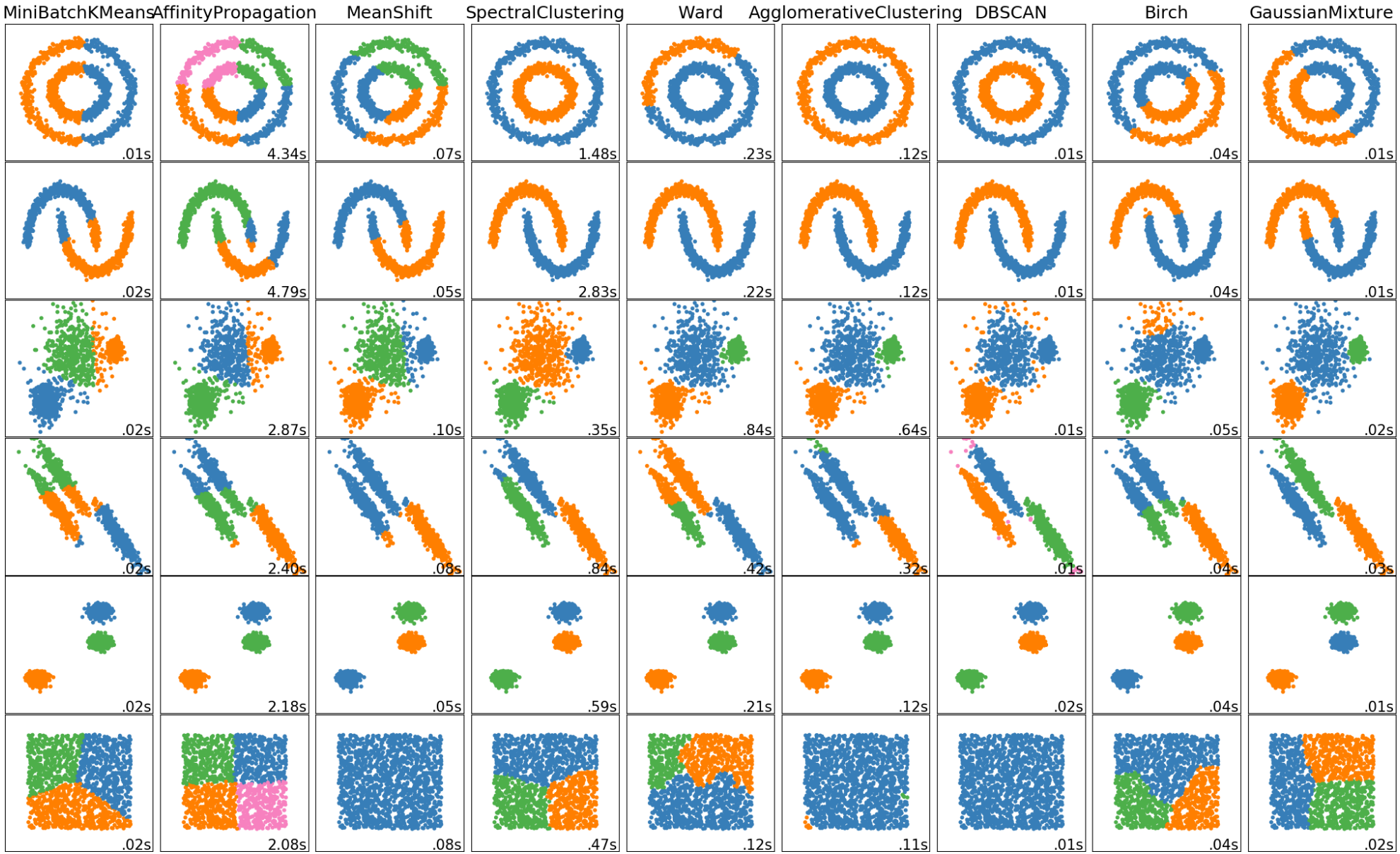

300 input points
9 iterations
122.4 distortion
7 iterations
122.5 distortion
4 iterations
23.2 distortion
3 iterations
23.2 distortion
4 iterations
23.2 distortion
6 iterations
23.2 distortion
6 iterations
23.2 distortion
5 iterations
23.2 distortion
4 iterations
23.2 distortion
4 iterations
122.4 distortion
2 iterations
23.2 distortion
3 iterations
23.2 distortion
3 iterations
23.2 distortion
4 iterations
23.2 distortion
k-means is a heuristic
output can be far from optimal

k-means

k_means.py

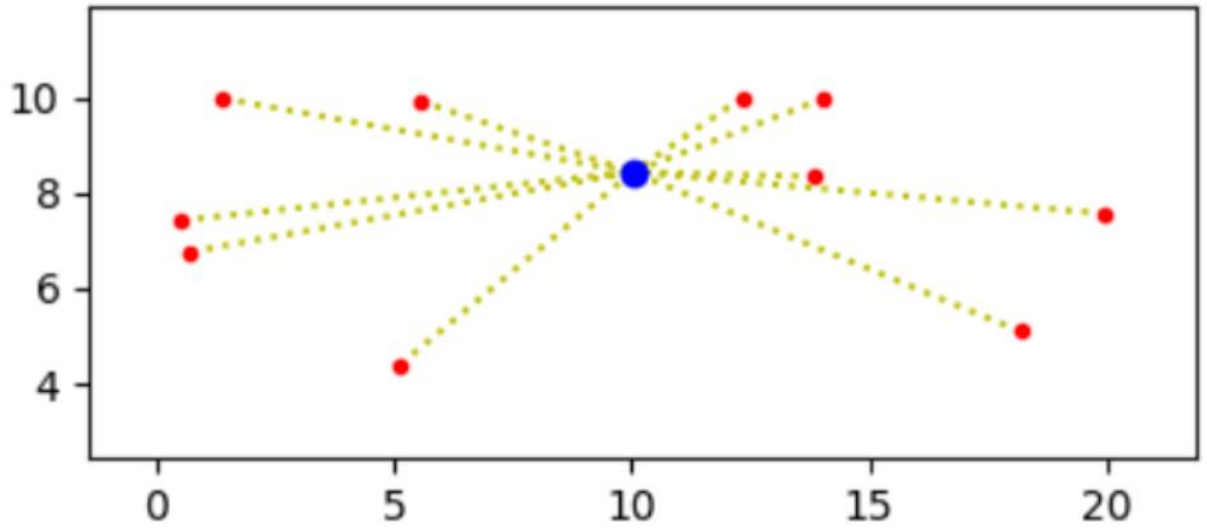
```
from random import sample
from numpy import argmin

def k_means(points, k):
    centroid = sample(points, k)
    centroids = [ centroid ]

    while True:
        clusters = [[] for _ in centroid]
        for p in points:
            i = argmin([dist(p, c) for c in centroid])
            clusters[i].append(p)

        centroid = [tuple(map(mean, zip(*c))) for c in clusters]

        if centroid == centroids[-1]:
            break

        centroids.append(centroid)
        if min(len(c) for c in clusters) == 0:
            print("Not good - empty cluster")
            break

    return clusters
```

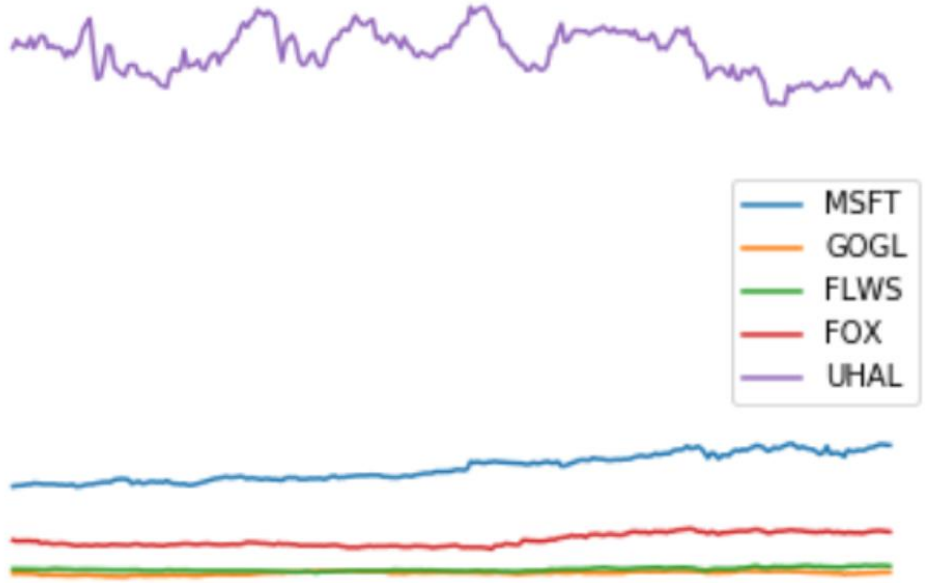
Other Python clustering methods - `sklearn.cluster`

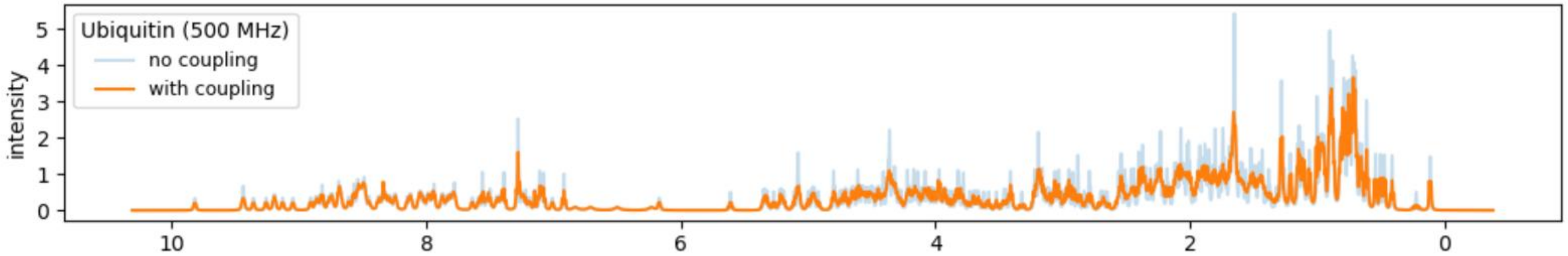


http://scikit-learn.org/stable/modules/clustering.html

Afsluttende projekter (del af karakter, emne valgfrit)







Overordnet erfaringer

- Kurset overordnet godt modtaget
- Kæmpe arbejde med at lave et bredt teknisk kursus
  - specielt når man vil målrette til de forskellige studenters baggrund

- Python godt valg som introducerende programmeringssprog
  - Simpelt programmeringssprog
  - Avanceret programmeringssprog
  - Kæmpe økosystem af pakker (+100.000) til næsten hvad-som-helst