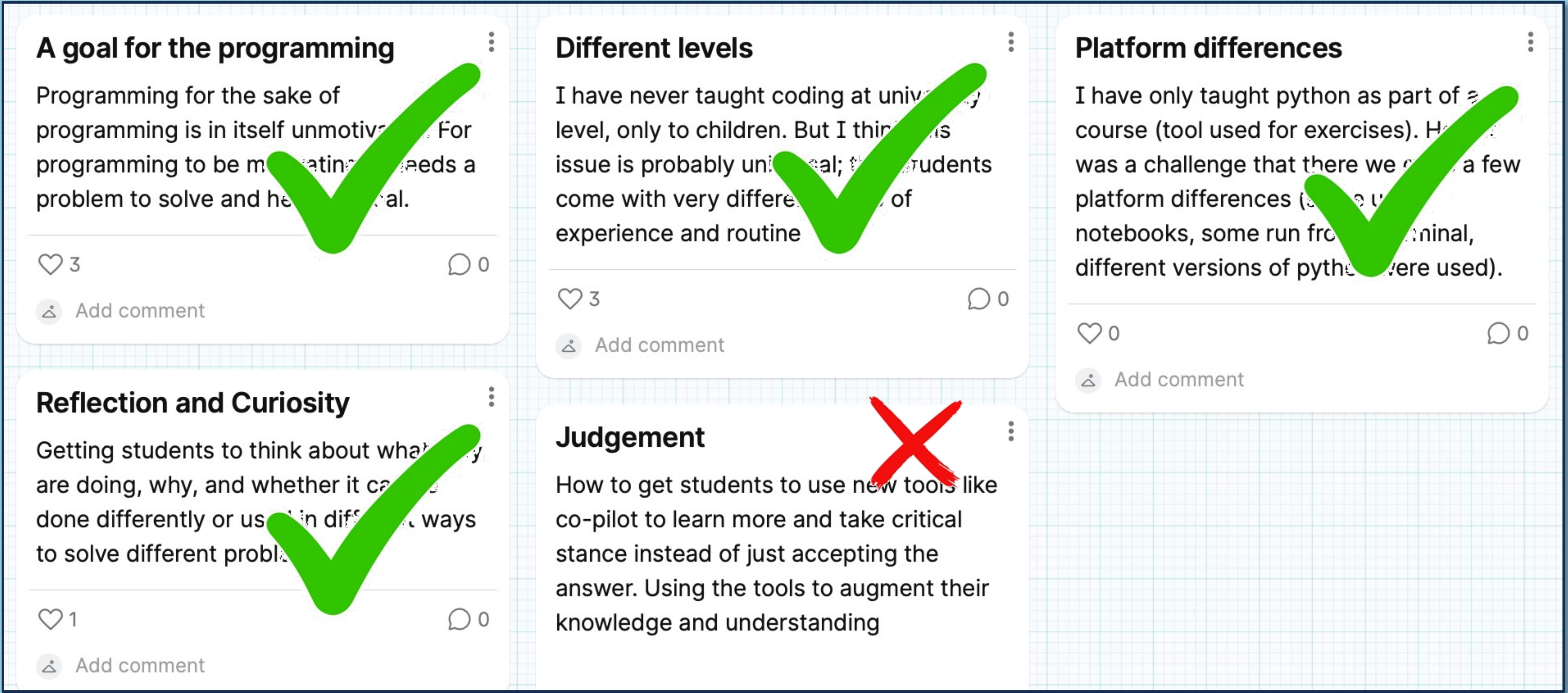
Programming for physics students at the University of Amsterdam

Ivo van Vulpen

Workshop programming education - Copenhagen -18 June 2024

Your remarks & discussion points



A goal for the programming

Programming for the sake of programming is in itself unmotiva . For programming to be m ating eeds a problem to solve and he al.

♡ 3 💬 0

Add comment

Different levels

I have never taught coding at univ y level, only to children. But I thin is issue is probably uni al; t udents come with very differe of experience and routine

♡ 3 💬 0

Add comment

Platform differences

I have only taught python as part of a course (tool used for exercises). H was a challenge that there we a few platform differences (notebooks, some run fro minal, different versions of pyth ere used).

♡ 0 💬 0

Add comment

Reflection and Curiosity

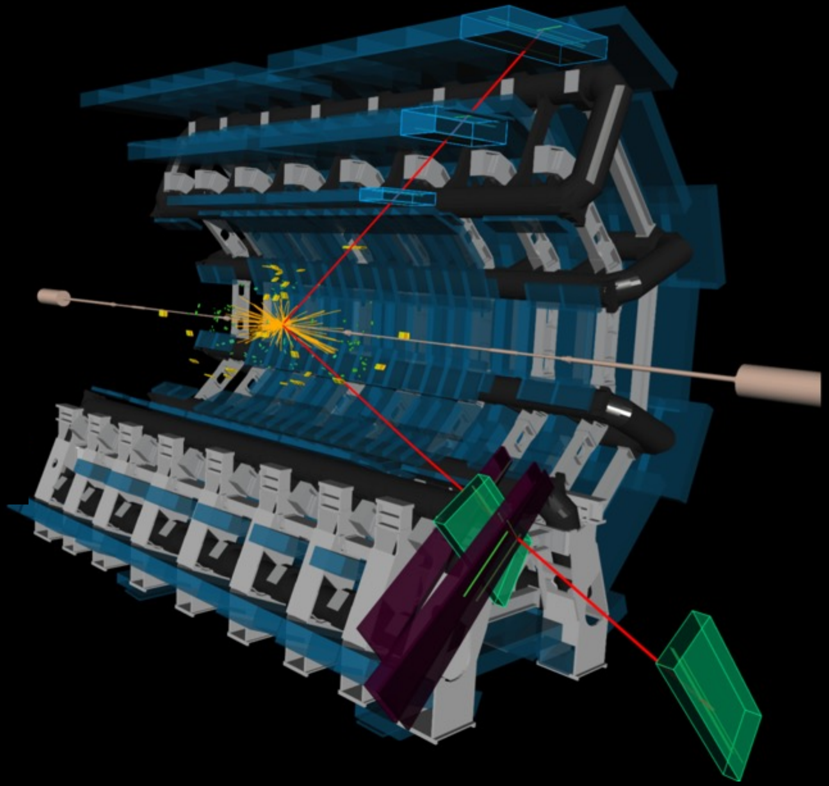
Getting students to think about wha y are doing, why, and whether it ca e done differently or us in dif ways to solve different probl

♡ 1 💬 0

Add comment

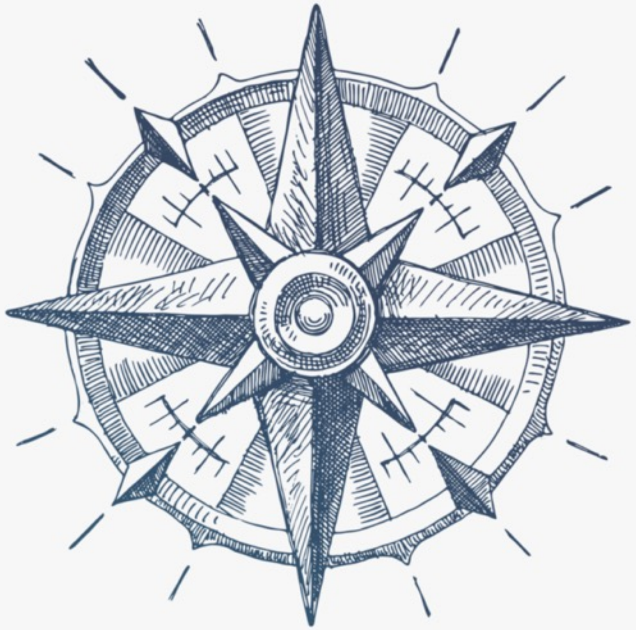
Judgement

How to get students to use new tools like co-pilot to learn more and take critical stance instead of just accepting the answer. Using the tools to augment their knowledge and understanding



Elementary particles (CERN)

Working at a university



Research

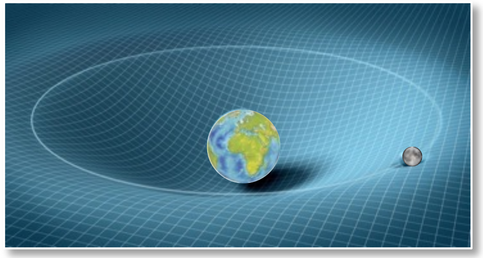
Organisation

Education

Outreach

Why students come to study physics

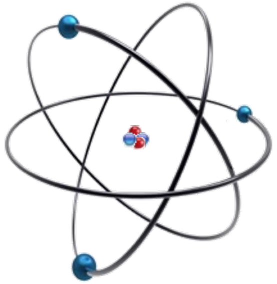
Cosmology



$$R_{\mu\nu} - \frac{1}{2}Rg_{\mu\nu} + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4}T_{\mu\nu}$$

Theory of relativity

Atomic physics



$$i\hbar\frac{d}{dt}\Psi(\vec{r},t) = \left[-\frac{\hbar^2}{2m}\vec{\nabla}^2 + V(\vec{r},t)\right]\Psi(\vec{r},t)$$

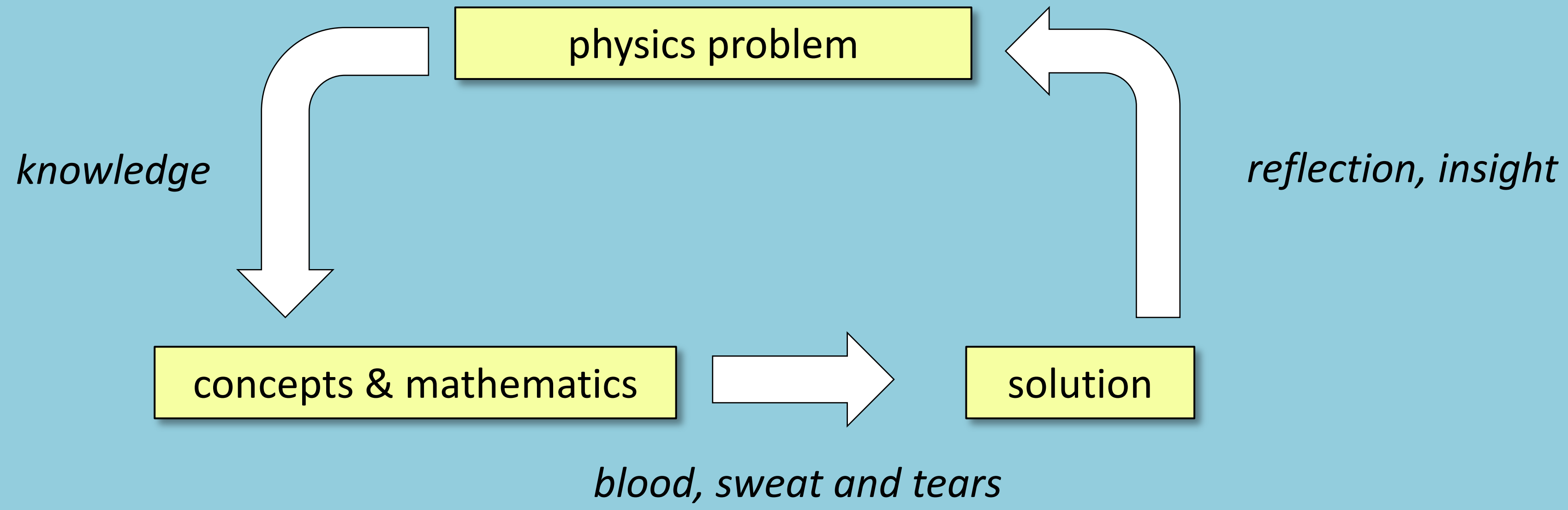
Quantum mechanics

Particle physics

$$(i\gamma^\mu\partial_\mu - m)\psi = 0$$

Quantum Field Theory

Learning physics



physics problem

knowledge

reflection, insight

concepts & mathematics

solution

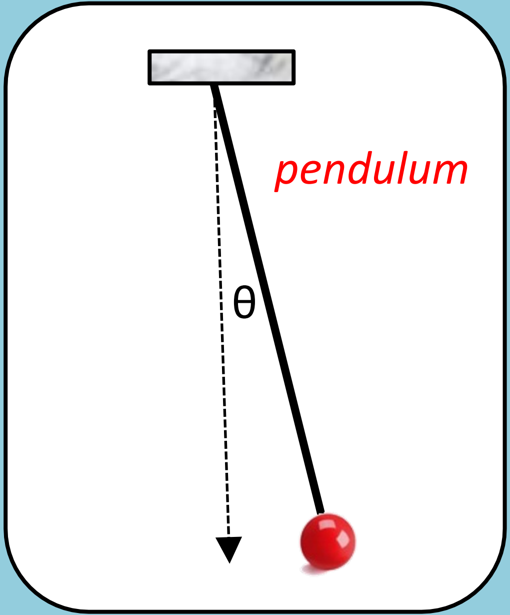
blood, sweat and tears

Observation:

- Too much focus on mathematics & too little on reflection/concepts
- University is not a sieve to produce our (few) successors
- Many simple problems do not have a (simple) analytic solution

→ Research requires more skills than just knowledge and math

Simple physics, tricky math



Euler-Lagrange

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = 0$$

differential equation

$$\ddot{\theta} + \frac{g}{l}\sin(\theta) = 0$$

general solution

$$T = 2\pi\sqrt{\frac{l}{g}}\sum_{n=0}^{\infty}\left[\left(\frac{(2n)!}{(2^n n!)^2}\right)^2 \sin^{2n}\left(\frac{\theta_0}{2}\right)\right]$$

small angles

$$T = 2\pi\sqrt{\frac{l}{g}}$$

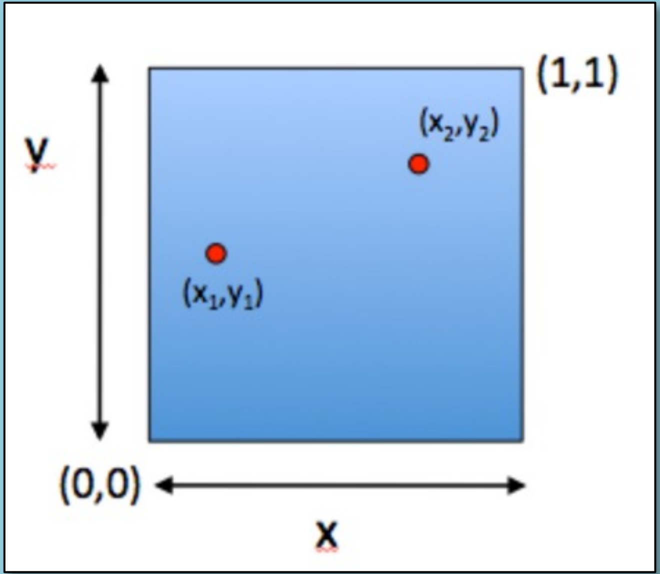
What about strings, friction, coupled, mass, external forces etc.?

Difficult easy problems

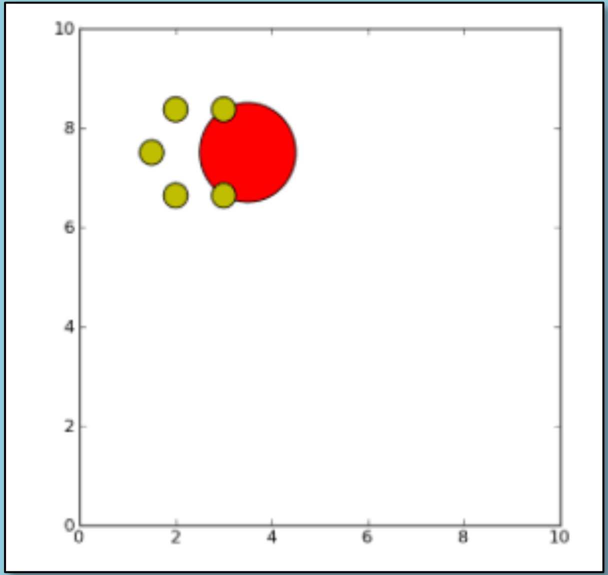
Is 23456789 a prime number?

$$\int_0^1 x^x dx$$

Elementary mathematics



Mean distance between two points in a square?



Velocity distribution of particles in a two-dimensional box

What was the third hottest 18th of June in Copenhagen since 1901?

How would you approach these problems?

First year, first block (7 weeks)

special relativitity

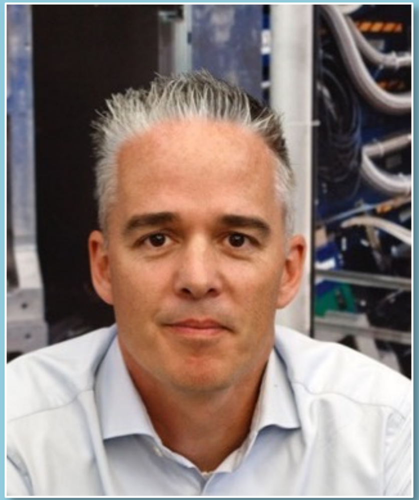
calculus

programming

Computing course

1st year introduction course - 3EC

Join forces: physics and computer science



Ivo van Vulpen (physics)

Martijn Stegeman (computer science)

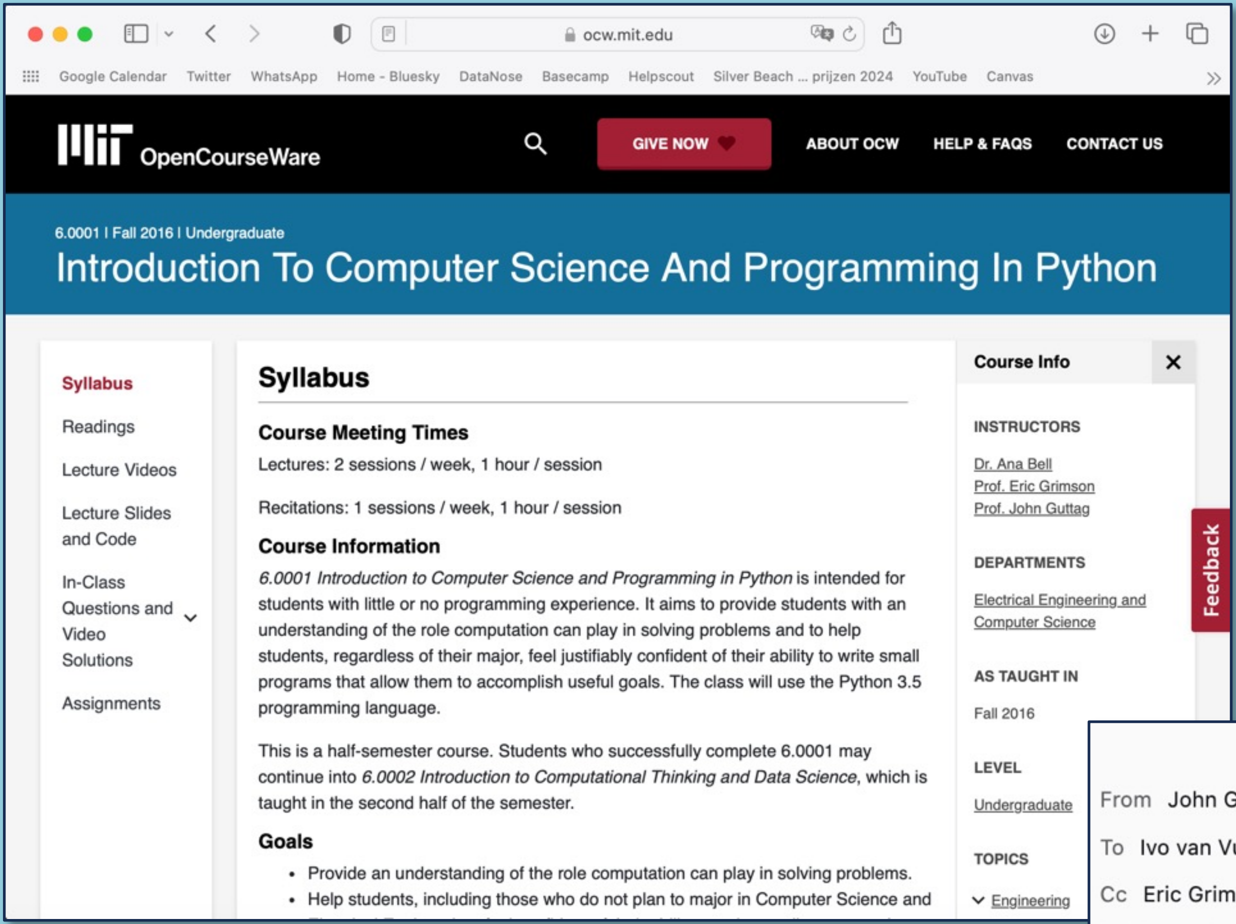
- This choice has had an enormous impact!

python™

- Popular and used in research & industry
- Easy to start, visualisation, open source, large community
- Full range: ‘advanced calculator’ to data-analysis

● **No** easy discussion when we started 10 years ago (C++, Mathematica, MatLab, …)

MIT online course



6.0001 | Fall 2016 | Undergraduate

Introduction To Computer Science And Programming In Python

Syllabus

Course Meeting Times

Lectures: 2 sessions / week, 1 hour / session

Recitations: 1 sessions / week, 1 hour / session

Course Information

6.0001 Introduction to Computer Science and Programming in Python is intended for students with little or no programming experience. It aims to provide students with an understanding of the role computation can play in solving problems and to help students, regardless of their major, feel justifiably confident of their ability to write small programs that allow them to accomplish useful goals. The class will use the Python 3.5 programming language.

This is a half-semester course. Students who successfully complete 6.0001 may continue into *6.0002 Introduction to Computational Thinking and Data Science*, which is taught in the second half of the semester.

Goals

- Provide an understanding of the role computation can play in solving problems.
- Help students, including those who do not plan to major in Computer Science and

Course Info

INSTRUCTORS

Dr. Ana Bell
Prof. Eric Grimson
Prof. John Guttag

DEPARTMENTS

Electrical Engineering and Computer Science

AS TAUGHT IN

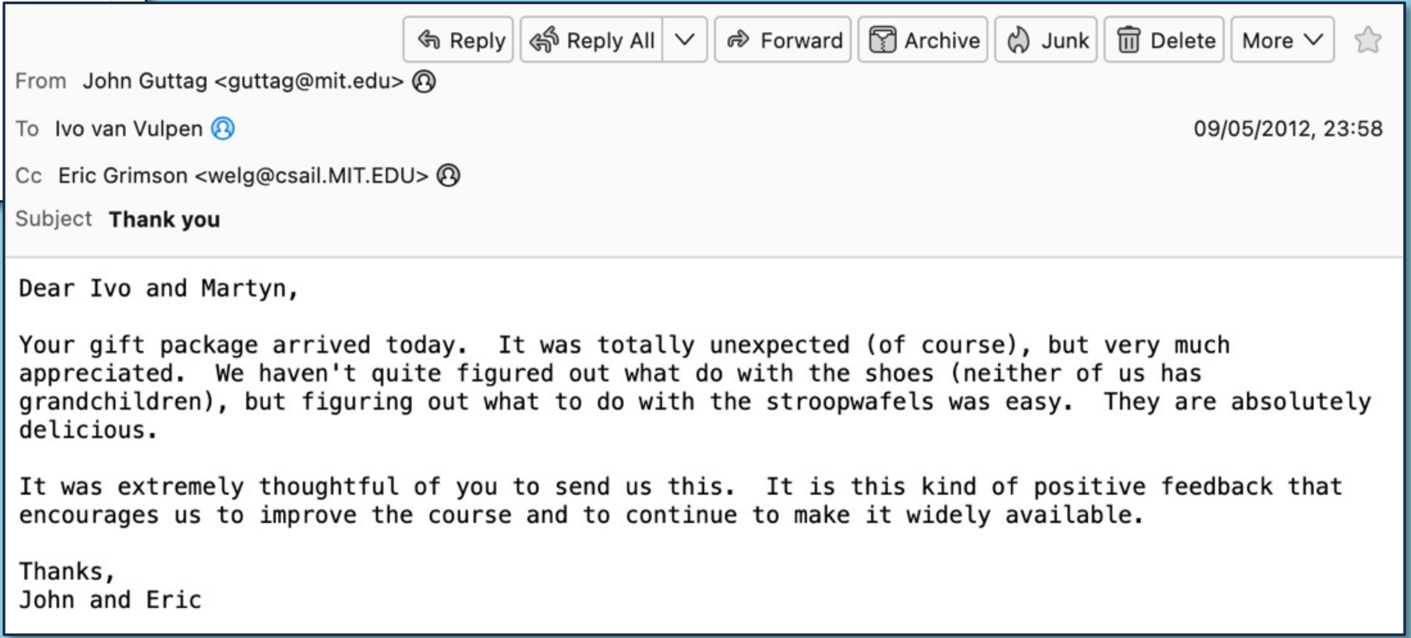
Fall 2016

LEVEL

Undergraduate

TOPICS

Engineering



From John Guttag <guttag@mit.edu>
To Ivo van Vulpen
09/05/2012, 23:58
Cc Eric Grimson <welg@csail.MIT.EDU>
Subject **Thank you**

Dear Ivo and Martyn,

Your gift package arrived today. It was totally unexpected (of course), but very much appreciated. We haven't quite figured out what do with the shoes (neither of us has grandchildren), but figuring out what to do with the stroopwafels was easy. They are absolutely delicious.

It was extremely thoughtful of you to send us this. It is this kind of positive feedback that encourages us to improve the course and to continue to make it widely available.

Thanks,
John and Eric

- Why reinvent the wheel?

Philosophy of our approach

Goals:

- Show that programming is an essential skill (for a scientist)
- Demystify and take away fear for programming → build confidence
- Focus on creativity and computational thinking - not programming itself

How:

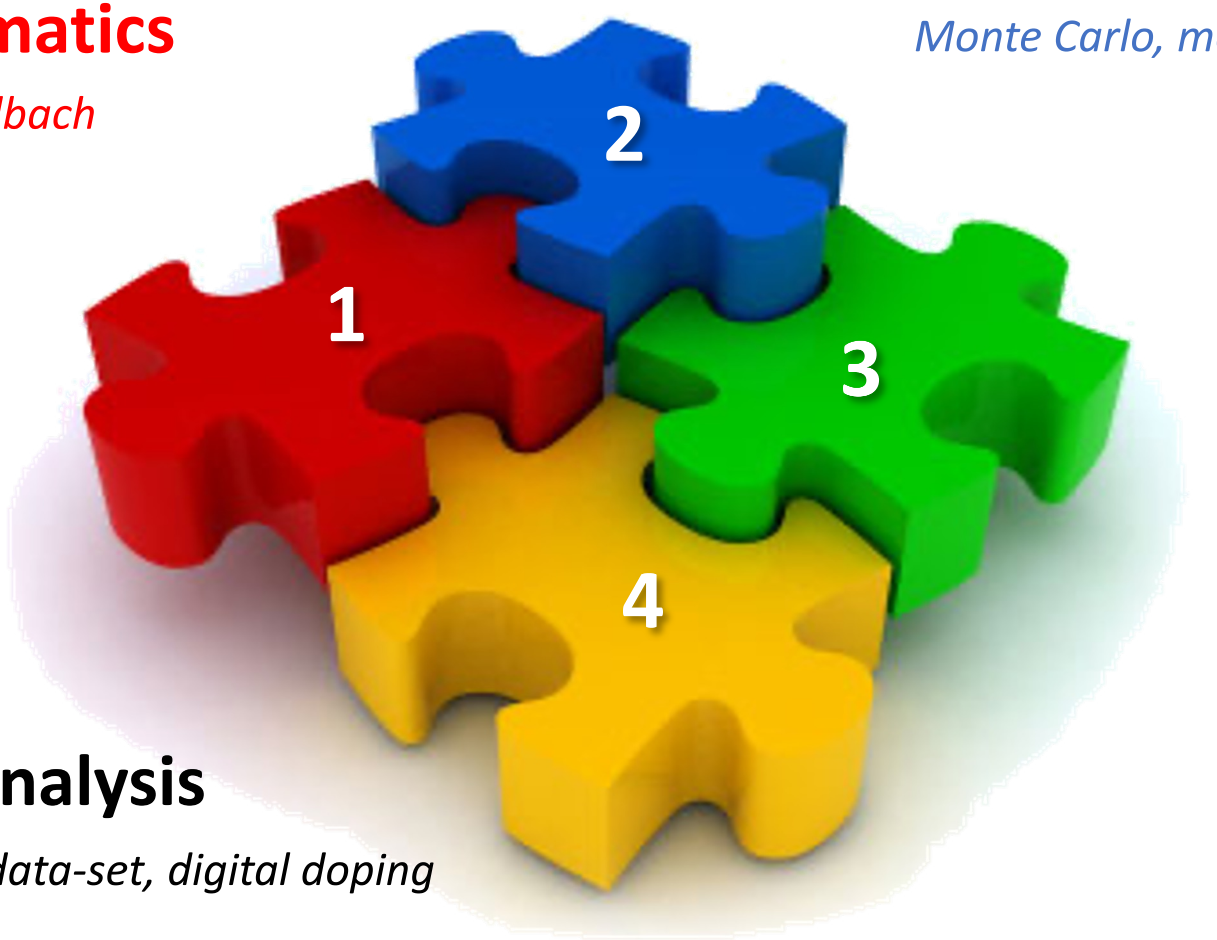
- Explore in 4 module different areas where computing is used
- Start from mathematics and physics problems instead of programming itself
- No standard libraries, ***do it yourself using only a few building blocks***
- No magic, on your own laptop!

(1) Basic mathematics

Prime numbers, Goldbach

(2) Numerical techniques

Monte Carlo, modelling, fitting



(3) simulations

Dynamics, 2d-collisions, prey-predator, Monopoly

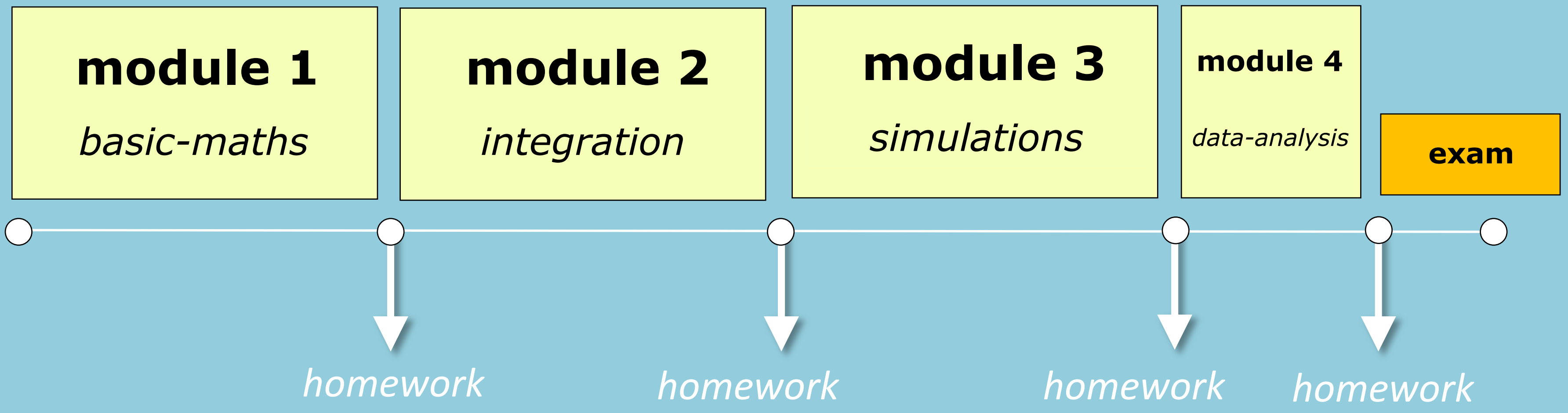
(4) data-analysis

Weather data-set, digital doping

Python: visualisation *(graphs, animation),* ***concepts*** *(random numbers, Monte-Carlo),* ***data-processing*** *(data I/O, simulations)*

Setting up the course

7 weeks in total



module 1

basic-maths

module 2

integration

module 3

simulations

module 4

data-analysis

exam

homework

homework

homework

homework

Grading: 40% homework and 60% exam

Computing course

160+ students, so logistics matter

Practical set-up

Single all-hands-on-deck lecture

Groups of 40 students (2 TA's)

2 x 4 hours per week (16 hours per module)

Teaching assistants

Selah

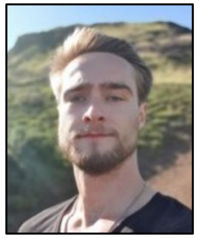
Liesbeth

Sietse

Suzanne

Thijs

Peter


Ezra

Nassim

+

Beau

Each TA supervises a group of 20 students.
In every tutorial room there are 40 students (2 TA)

- Extra TA for students that need extra attention to build confidence and ‘get over the 1st hurdle’

Setting up the course

https://progns.proglab.nl/

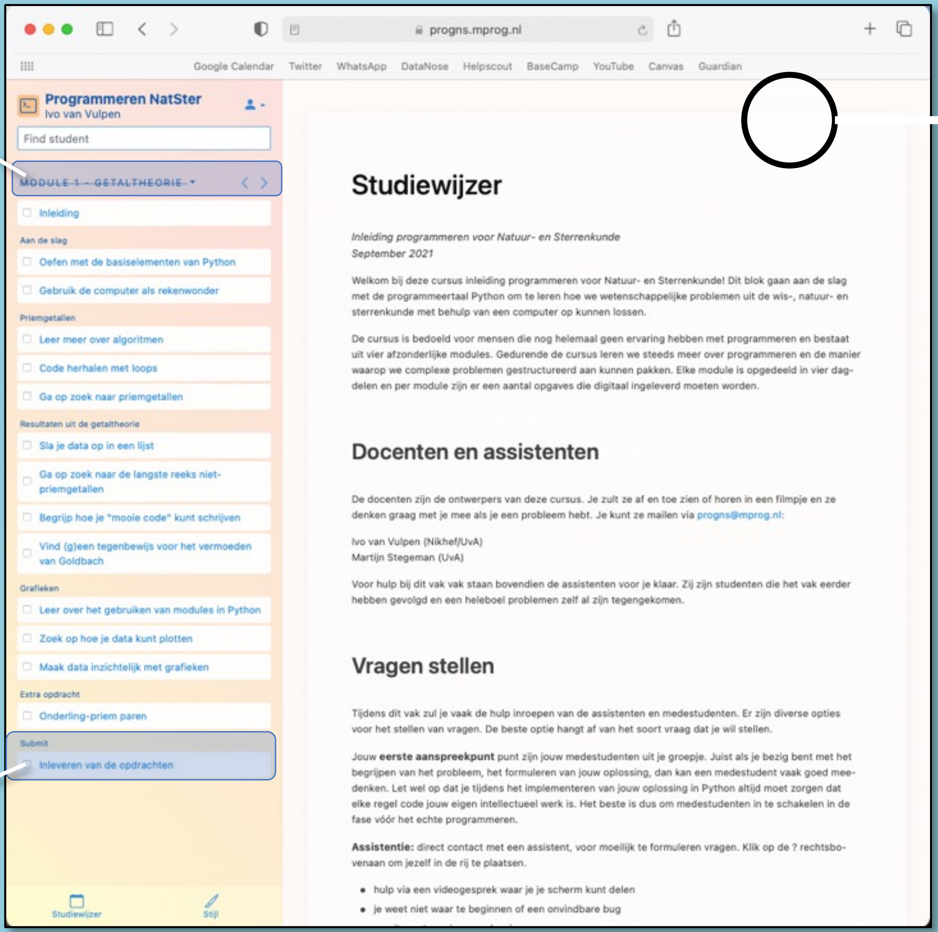
[4] discussion assistant (queue)

Navigate modules (text and clips)

[1] basic-exercises
- submit: day 4

[2] extra exercise

[3] hand in



☐ Inleiding

Aan de slag

☐ Oefen met de basiselementen van Python

☐ Gebruik de computer als rekenwonder

Priemgetallen

☐ Leer meer over algoritmen

☐ Code herhalen met loops

☐ Ga op zoek naar priemgetallen

Resultaten uit de getaltheorie

☐ Sla je data op in een lijst

☐ Ga op zoek naar de langste reeks niet-priemgetallen

☐ Begrijp hoe je "mooie code" kunt schrijven

☐ Vind (g)een tegenbewijs voor het vermoeden van Goldbach

Grafieken

☐ Leer over het gebruiken van modules in Python

☐ Zoek op hoe je data kunt plotten

☐ Maak data inzichtelijk met grafieken

Extra opdracht

☐ Onderling-priem paren

Submit

☐ Inleveren van de opdrachten

Studiewijzer

Inleiding programmeren voor Natuur- en Sterrenkunde
September 2021

Welkom bij deze cursus inleiding programmeren voor Natuur- en Sterrenkunde! Dit blok gaan aan de slag met de programmeertaal Python om te leren hoe we wetenschappelijke problemen uit de wis-, natuur- en sterrenkunde met behulp van een computer op kunnen lossen.

De cursus is bedoeld voor mensen die nog helemaal geen ervaring hebben met programmeren en bestaat uit vier afzonderlijke modules. Gedurende de cursus leren we steeds meer over programmeren en de manier waarop we complexe problemen gestructureerd aan kunnen pakken. Elke module is opgedeeld in vier dagdelen en per module zijn er een aantal opgaves die digitaal ingeleverd moeten worden.

Docenten en assistenten

De docenten zijn de ontwerpers van deze cursus. Je zult ze af en toe zien of horen in een filmpje en ze denken graag met je mee als je een probleem hebt. Je kunt ze mailen via progns@mprog.nl:

Ivo van Vulpen (Nikhef/UvA)
Martijn Stegeman (UvA)

Voor hulp bij dit vak vak staan bovendien de assistenten voor je klaar. Zij zijn studenten die het vak eerder hebben gevolgd en een heleboel problemen zelf al zijn tegengekomen.

Vragen stellen

Tijdens dit vak zul je vaak de hulp inroepen van de assistenten en medestudenten. Er zijn diverse opties voor het stellen van vragen. De beste optie hangt af van het soort vraag dat je wil stellen.

Jouw **eerste aanspreekpunt** punt zijn jouw medestudenten uit je groepje. Juist als je bezig bent met het begrijpen van het probleem, het formuleren van jouw oplossing, dan kan een medestudent vaak goed meedenken. Let wel op dat je tijdens het implementeren van jouw oplossing in Python altijd moet zorgen dat elke regel code jouw eigen intellectueel werk is. Het beste is dus om medestudenten in te schakelen in de fase vóór het echte programmeren.

Assistentie: direct contact met een assistent, voor moeilijk te formuleren vragen. Klik op de ? rechtsbovenaan om jezelf in de rij te plaatsen.

- hulp via een videogesprek waar je je scherm kunt delen
- je weet niet waar te beginnen of een onvindbare bug

Single website!

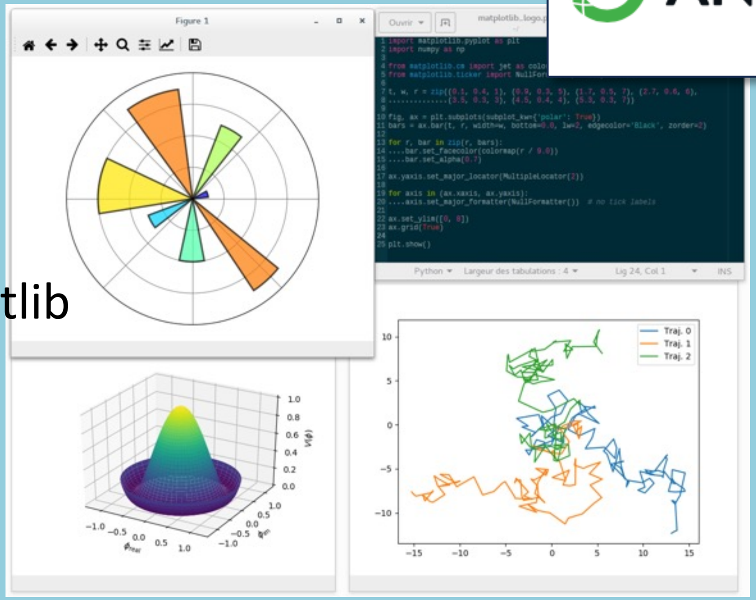
Students can only use functions that are introduced in the course

Computing set-up

Python and packages

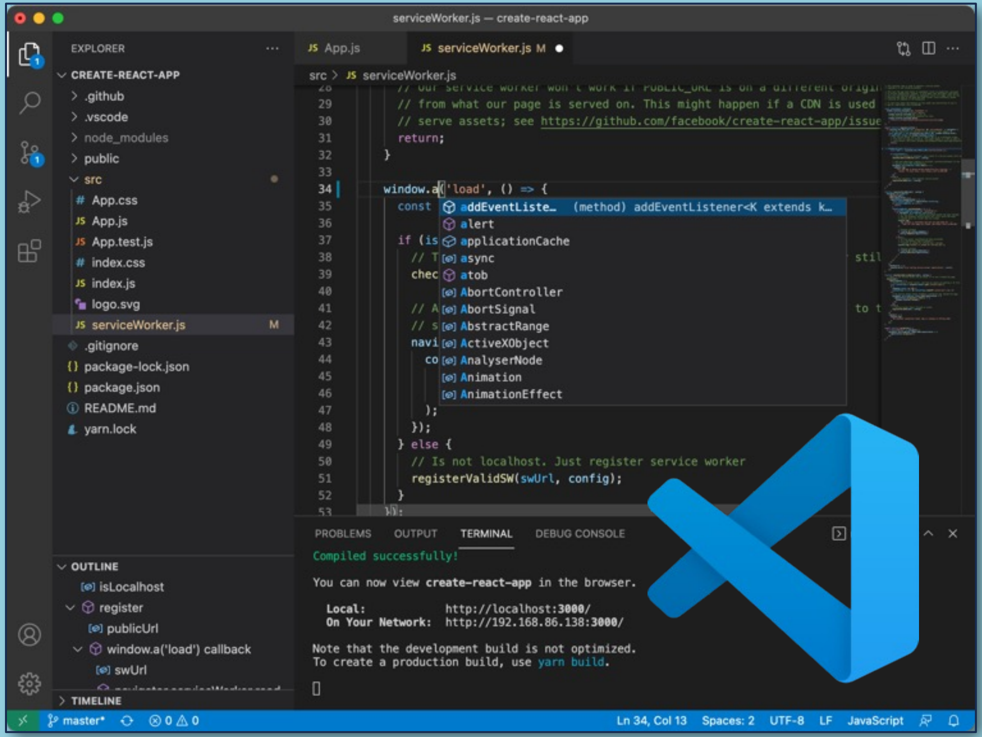
ANACONDA

matplotlib



```
import matplotlib.pyplot as plt
import numpy as np

from matplotlib.cm import jet as colo
from matplotlib.ticker import NullFor

t, w, r = zip((0.1, 0.4, 1), (0.9, 0.3, 5), (1.7, 0.5, 7), (2.7, 0.6, 6),
...............(3.5, 0.3, 3), (4.5, 0.4, 4), (5.3, 0.3, 7))

fig, ax = plt.subplots(subplot_kw={'polar': True})
bars = ax.bar(t, r, width=w, bottom=0.0, lw=2, edgecolor='Black', zorder=2)

for r, bar in zip(r, bars):
....bar.set_facecolor(colormap(r / 9.0))
....bar.set_alpha(0.7)

ax.yaxis.set_major_locator(MultipleLocator(2))

for axis in (ax.xaxis, ax.yaxis):
....axis.set_major_formatter(NullFormatter())  # no tick labels

ax.set_ylim([0, 8])
ax.grid(True)

plt.show()
```

Visual studio code - IDE - command line



```
      // from what our page is served on. This might happen if a CDN is used
      // serve assets; see https://github.com/facebook/create-react-app/issue
      return;
    }

  window.a('load', () => {
    const
    
    if (is
      // T                                                       stil
      chec

      // A                                                       to t
      // s
      navi
        co


      );
    });
  } else {
    // Is not localhost. Just register service worker
    registerValidSW(swUrl, config);
  }
```

```
Compiled successfully!

You can now view create-react-app in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.86.138:3000/

Note that the development build is not optimized.
To create a production build, use yarn build.

▯
```

- We did not use Jupyter notebooks, nor Git

Module

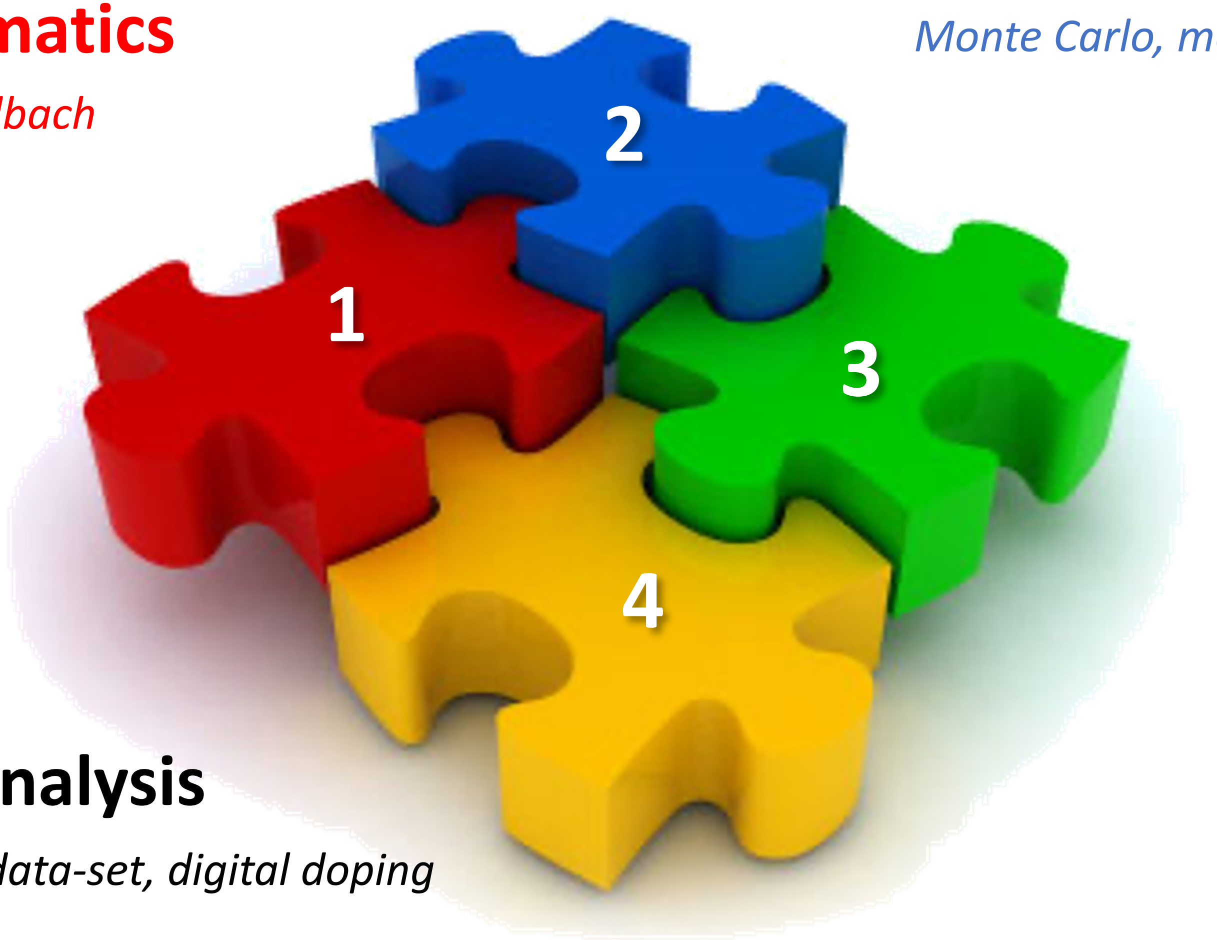
4 different areas where computing enters

(2) Numerical techniques

Monte Carlo, modelling, fitting

(1) Basic mathematics

Prime numbers, Goldbach



(3) simulations

Dynamics, 2d-collisions, prey-predator, Monopoly

(4) data-analysis

Weather data-set, digital doping

Python: visualisation *(graphs, animation),* ***concepts*** *(random numbers, Monte-Carlo),* ***data-processing*** *(data I/O, simulations)*

Module 1: mathematics

Python: variables, operators, logic, functions, loops, lists

Goall: *test Goldbach conjecture: "Every even number can be written as the sum of two prime numbers". 18 = 13+5, but what about 16788 = ? + ?*

step 1: intro Python: for-loops, functions, logic, printing **[clips, text and extra exercises]**

step 2: small piece of code (function) to test if single number is prime

step 3: program to find 100th prime number
… and the longest stretch of non-prime-numbers below n=10000

step 4: test Goldbach's claim up to n=1000

Teach students that intelligence/logic/ideas come from them.

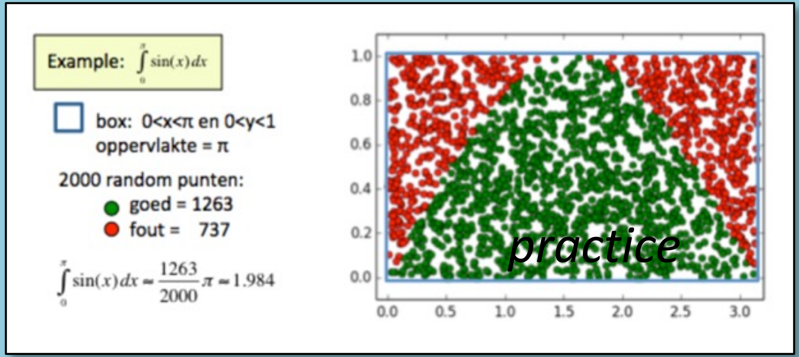
Test if 113 a prime number:

- First implementation: divide by 2 until 112
- Need only to test: 2, 3, 5, 7

Module 2: numerical techniques

Python: random numbers, graphs

Theory (Monte Carlo integration)



Example: $\int_0^{\pi} \sin(x)dx$

☐ box: 0<x<π en 0<y<1
oppervlakte = π

2000 random punten:
● goed = 1263
● fout = 737

$\int_0^{\pi} \sin(x)dx \approx \frac{1263}{2000}\pi \approx 1.984$

1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
practice

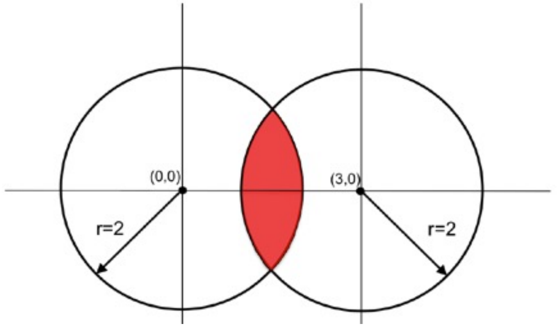
Practice/test

$$\int_0^{\pi} \sin(x)\mathrm{d}x$$

Homework

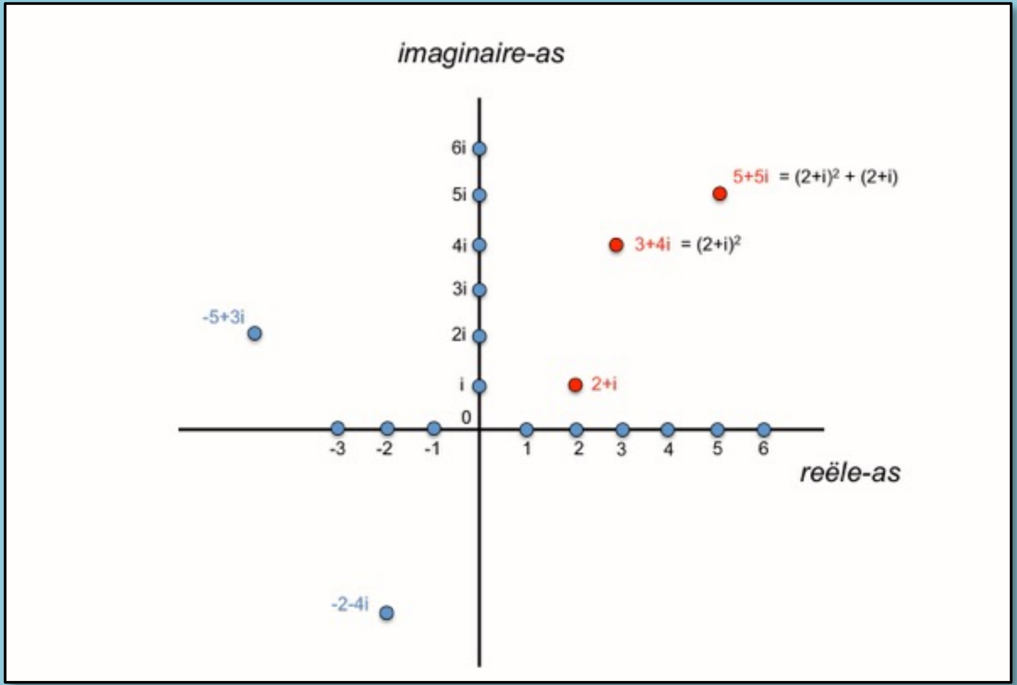
$$\int_{0.2}^{2.2} \tan(\cos(\sin(x)))\, dx$$

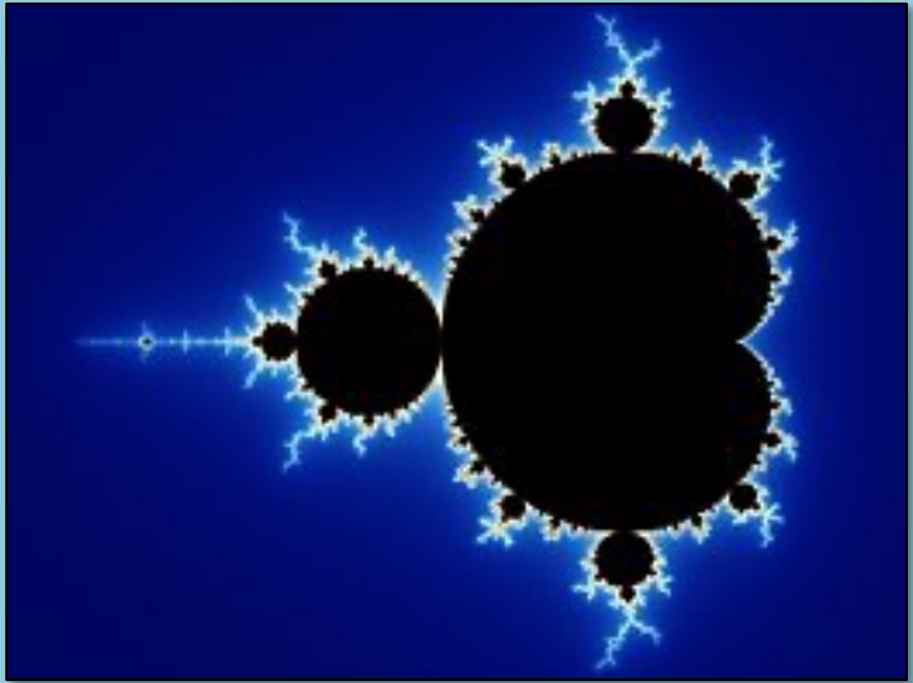
Exam



Compute red area

Extra exercise





Lear about complex numbers and construct the Mandelbrot set yourself

- Challenging 'extra' exercise (no restrictions) for last point score (1/10)
  Motivate the good students

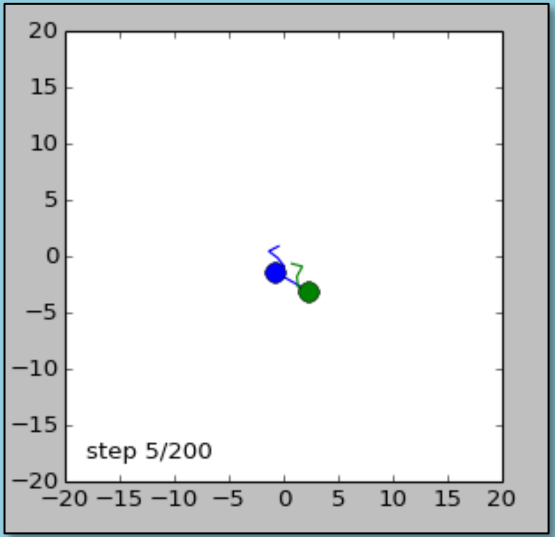
Module 3: simulations

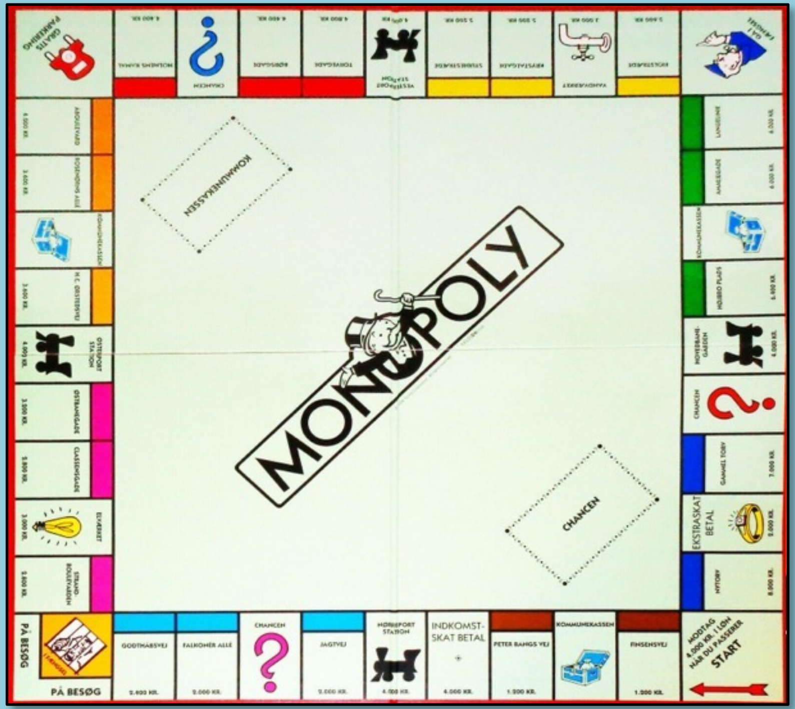
Python: animations and simulations

Effect friction in free fall

"Terminal velocity and how much longer can the base jumper enjoy the view thanks to air resistance?"

random walk





Nytorv meets de Kalverstraat

Throw dice with random number and walk around Copenhagen buying real estate like a billionaire

step 1: ***1 player, infinite money*** *- how much turns does it take on average to buy all streets?*

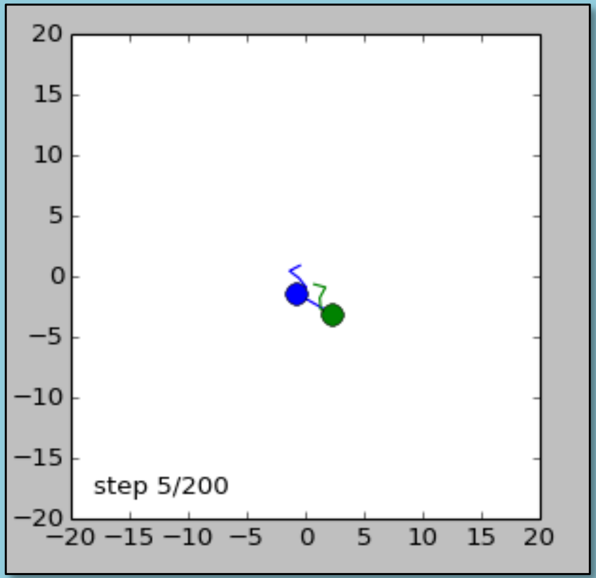
step 2: ***1 player, finite money*** *- how much turns does it take on average to buy all streets?*

step 3: ***2 players, finite mone****y - how much more streets does player 1 have when all streets are sold?*

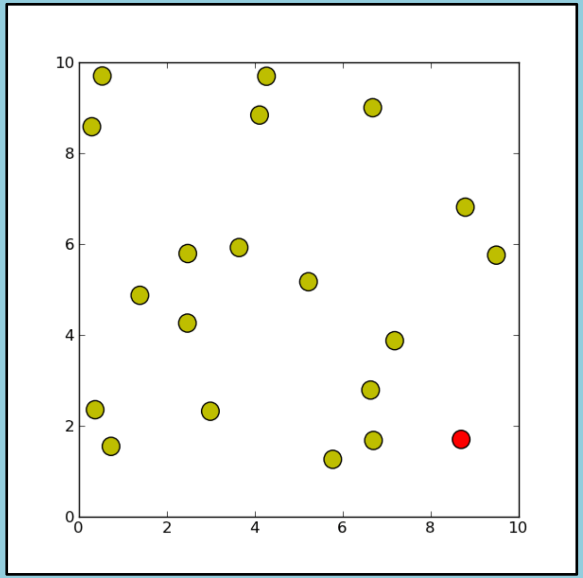
step 4: How much more money should we giver player 2 at the start to make sure the two players on average end up with the same number of streets?

Extra exercise

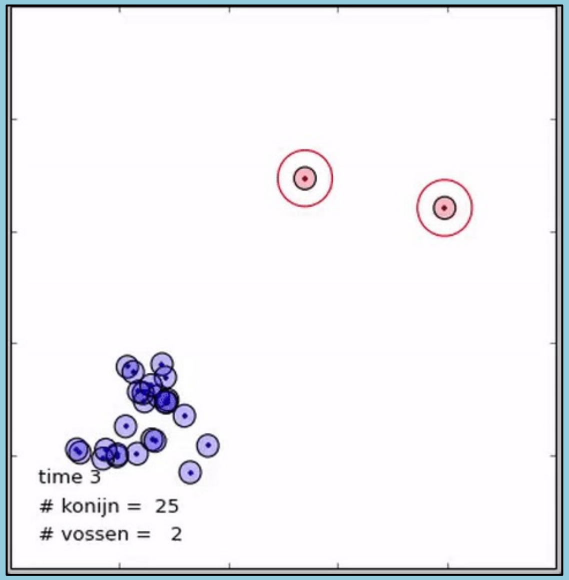
Python: animations



random walk



Thermodynamics



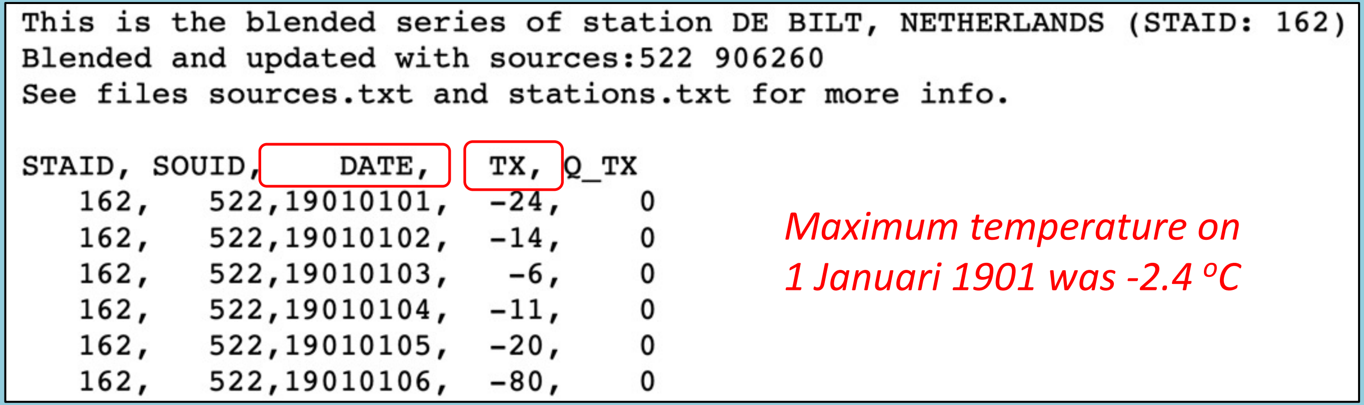
Prey-predator

- Thermodynamics hypnotizing. Many alternatives: potentials, astro, bio, …

Module 4: big data

Python: input/output

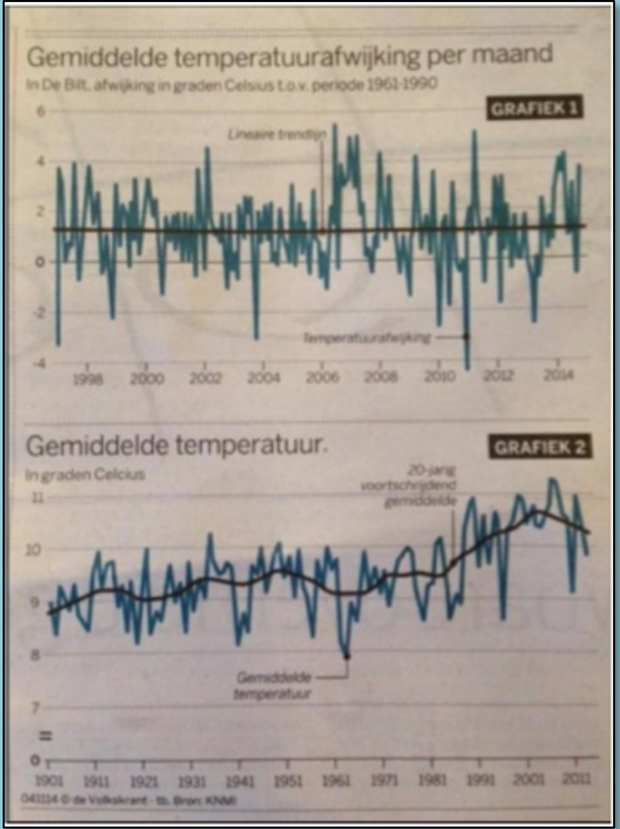
Info from Royal Metreology Institute on max and min temp per day



```
This is the blended series of station DE BILT, NETHERLANDS (STAID: 162)
Blended and updated with sources:522 906260
See files sources.txt and stations.txt for more info.

STAID, SOUID,      DATE,   TX, Q_TX
   162,    522,19010101,  -24,    0
   162,    522,19010102,  -14,    0
   162,    522,19010103,   -6,    0
   162,    522,19010104,  -11,    0
   162,    522,19010105,  -20,    0
   162,    522,19010106,  -80,    0
```

Maximum temperature on 1 Januari 1901 was -2.4 °C

- Warmest and coldest day?
- Longest period with temperatures below zero?
- Reproduce graphs from newspapers
- How special was the double heat-wave from last year?



Gemiddelde temperatuurafwijking per maand

In De Bilt, afwijking in graden Celsius t.o.v. periode 1961-1990

GRAFIEK 1

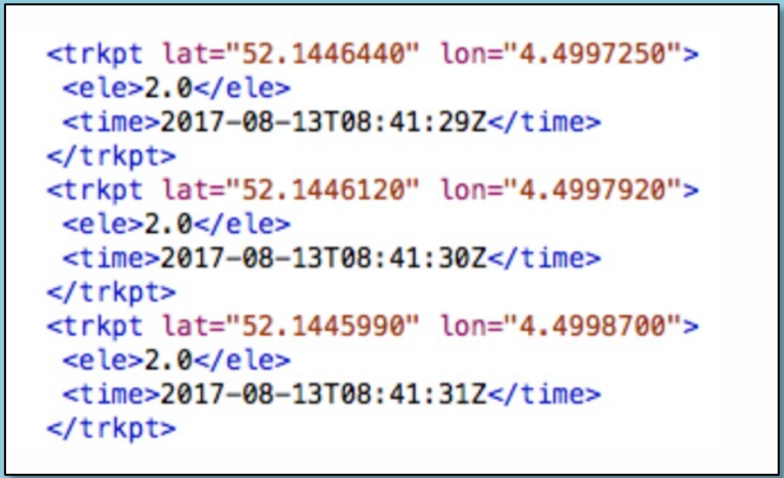
Gemiddelde temperatuur

In graden Celcius

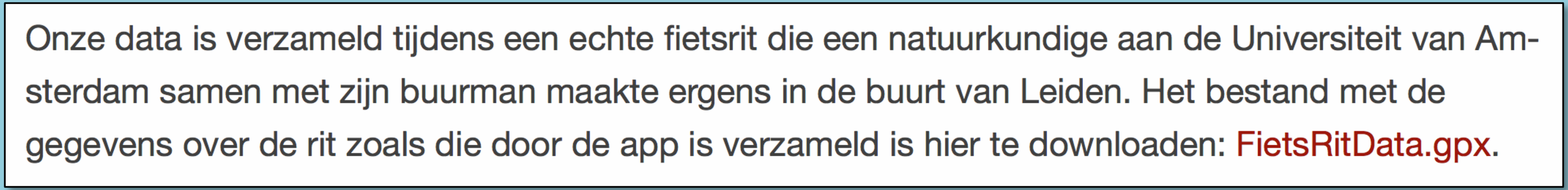
GRAFIEK 2

041114 © de Volkskrant - tb. Bron: KNMI

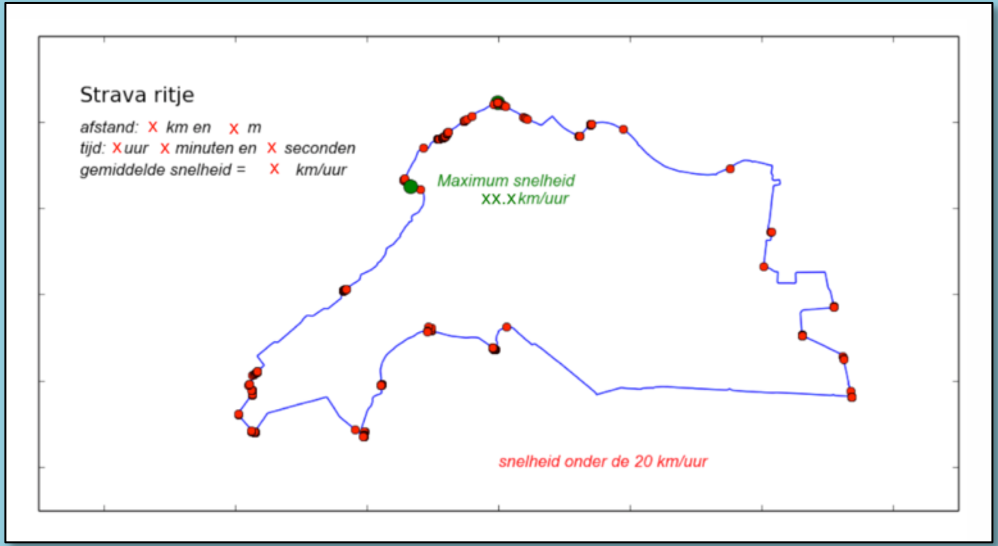
Alternative: digital doping



```
<trkpt lat="52.1446440" lon="4.4997250">
 <ele>2.0</ele>
 <time>2017-08-13T08:41:29Z</time>
</trkpt>
<trkpt lat="52.1446120" lon="4.4997920">
 <ele>2.0</ele>
 <time>2017-08-13T08:41:30Z</time>
</trkpt>
<trkpt lat="52.1445990" lon="4.4998700">
 <ele>2.0</ele>
 <time>2017-08-13T08:41:31Z</time>
</trkpt>
```



Onze data is verzameld tijdens een echte fietsrit die een natuurkundige aan de Universiteit van Amsterdam samen met zijn buurman maakte ergens in de buurt van Leiden. Het bestand met de gegevens over de rit zoals die door de app is verzameld is hier te downloaden: FietsRitData.gpx.



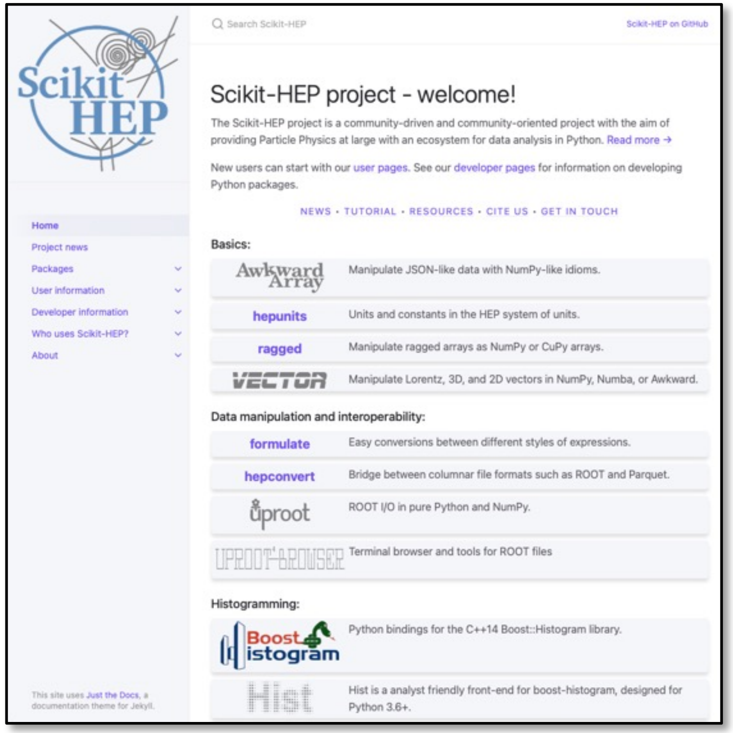
If they can do this they can also analyse other data (research project)

Extra exercise

Students are asked to collect their own data-set and make a graph … and a story!

Things we do not do

Complex libraries/environments



Scikit-HEP project - welcome!

The Scikit-HEP project is a community-driven and community-oriented project with the aim of providing Particle Physics at large with an ecosystem for data analysis in Python. Read more →

New users can start with our user pages. See our developer pages for information on developing Python packages.

Basics:

Awkward Array	Manipulate JSON-like data with NumPy-like idioms.
hepunits	Units and constants in the HEP system of units.
ragged	Manipulate ragged arrays as NumPy or CuPy arrays.
VECTOR	Manipulate Lorentz, 3D, and 2D vectors in NumPy, Numba, or Awkward.

Data manipulation and interoperability:

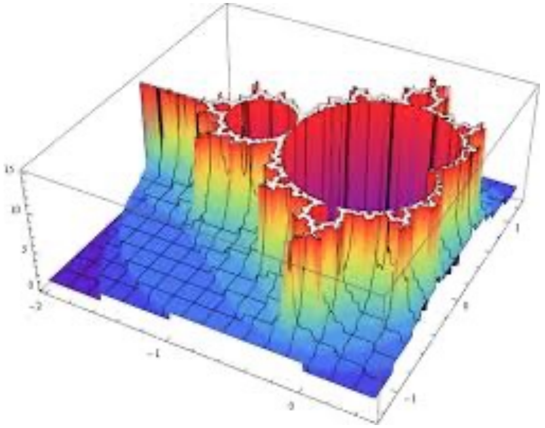
formulate	Easy conversions between different styles of expressions.
hepconvert	Bridge between columnar file formats such as ROOT and Parquet.
uproot	ROOT I/O in pure Python and NumPy.
UPROOT-BROWSER	Terminal browser and tools for ROOT files

Histogramming:

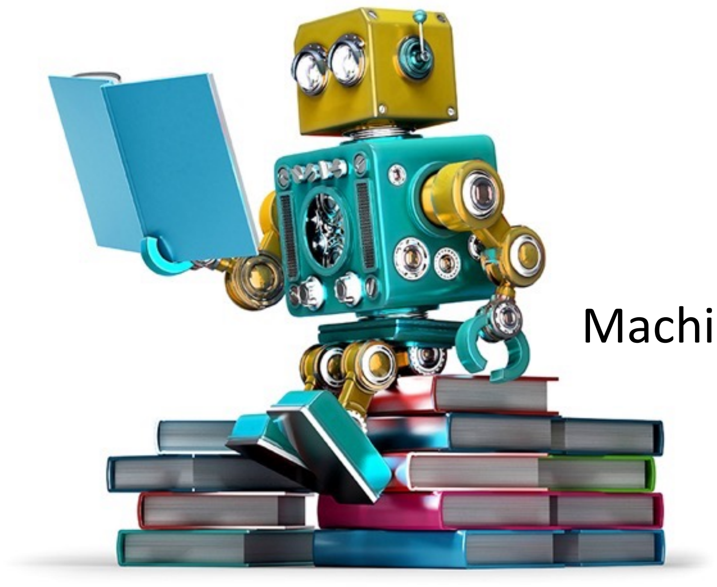
Boost Histogram	Python bindings for the C++14 Boost::Histogram library.
Hist	Hist is a analyst friendly front-end for boost-histogram, designed for Python 3.6+.

This site uses Just the Docs, a documentation theme for Jekyll.

Wolfram Mathematica



GitLab



Machine Learning

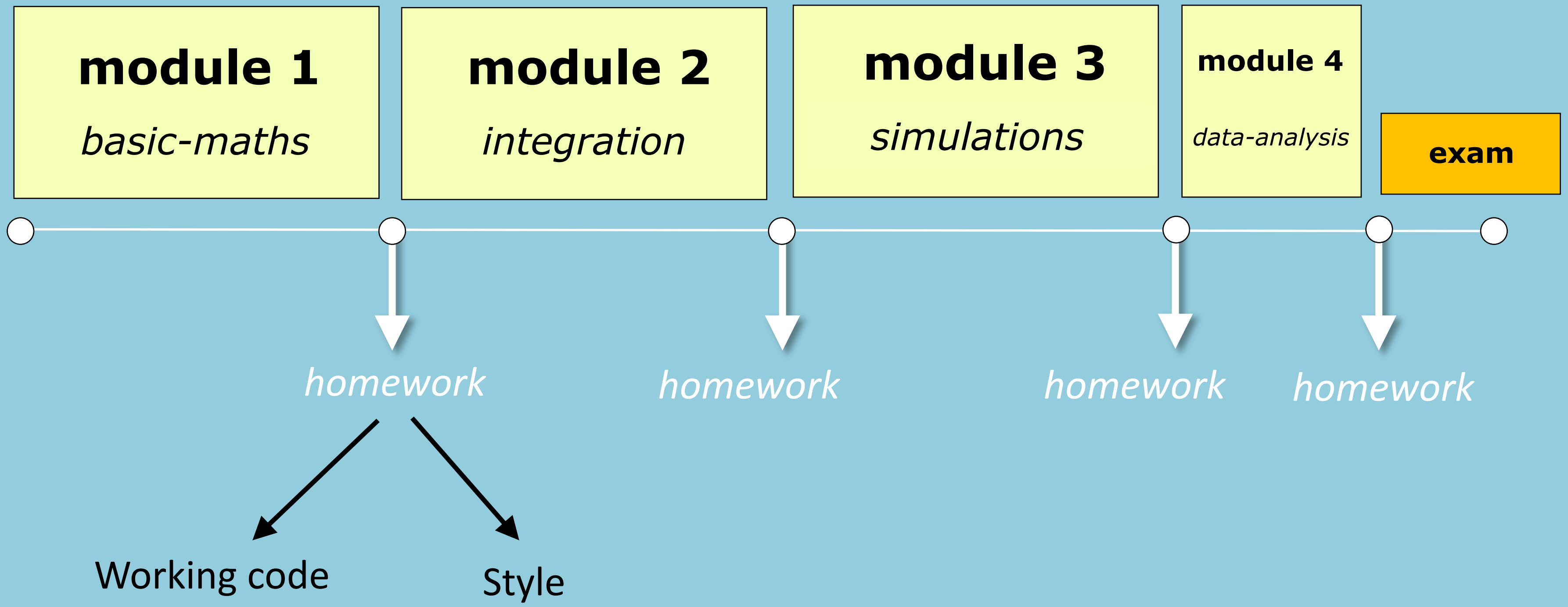
Grading

How do we come up with a final mark

hand in online, online feedback, plagiarism, tutorials, etc.

Homework (40%)

Students hand in code online and also get feedback online


module 1

basic-maths

module 2

integration

module 3

simulations

module 4

data-analysis

exam

homework

homework

homework

homework

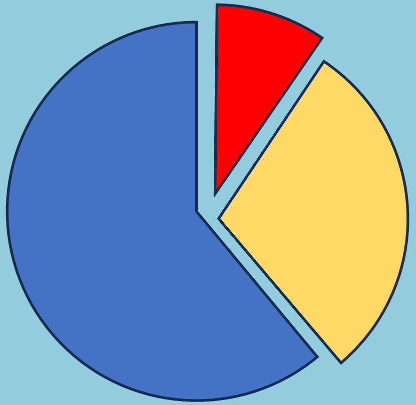
Working code

Style

Homework (40%)

Challenging exercise (1/9)

→ *Challenge for the good students*
(no restrictions on what you can use)



Working code (5/9)

→ *Deadline 1:*
students can test their answer
using using the same tool we do

Style of the code (3/9)

→ *Deadline 2: 2 days after first*
*AFTER a **code-review** with fellow student*

Code review

We tell students we are serious about style.
→ **Students hand in code twice**: 1x answers and (2 days later) 1x style (after **code-review**)

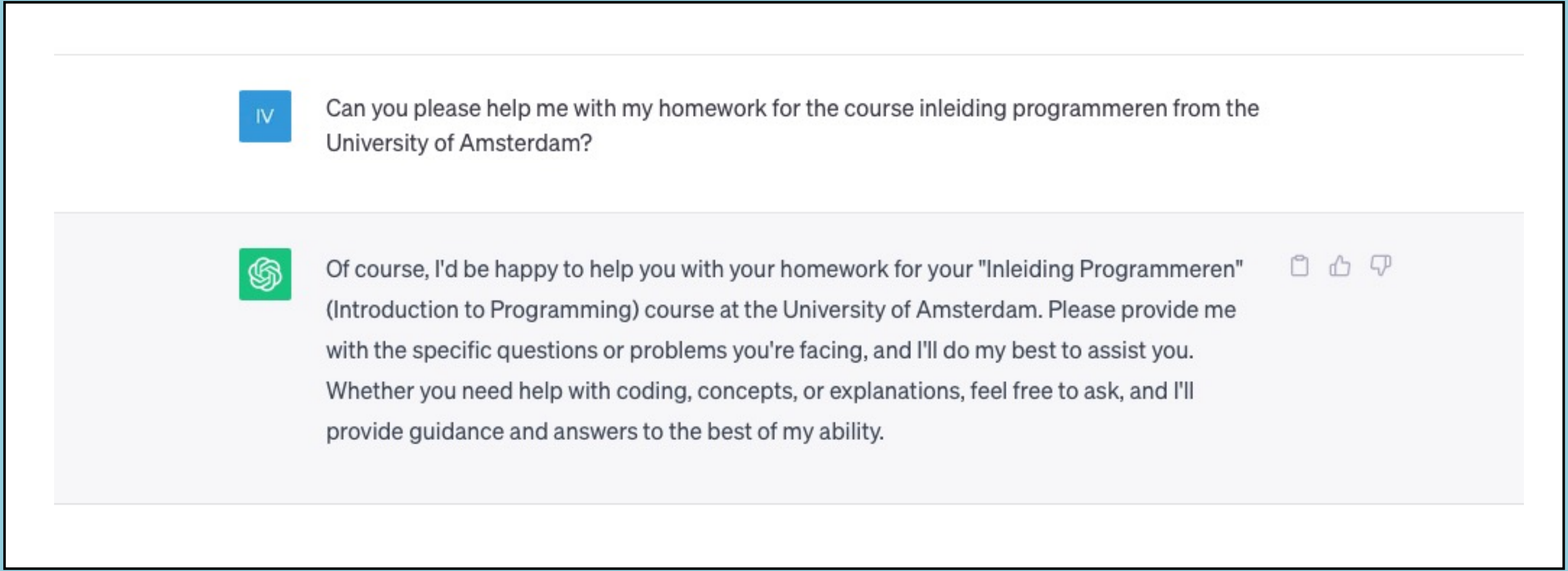
Code review:

Students pair up and review each others code (30 min max):

- read through code of other student and **think out loud**
- go through points in style guide (naming conventions, comments, readability, indentation etc.)
- advice your partner to make changes

→ Then time to adopt changes and hand in that same evening.

Plagiarism



Can you please help me with my homework for the course inleiding programmeren from the University of Amsterdam?

Of course, I'd be happy to help you with your homework for your "Inleiding Programmeren" (Introduction to Programming) course at the University of Amsterdam. Please provide me with the specific questions or problems you're facing, and I'll do my best to assist you. Whether you need help with coding, concepts, or explanations, feel free to ask, and I'll provide guidance and answers to the best of my ability.

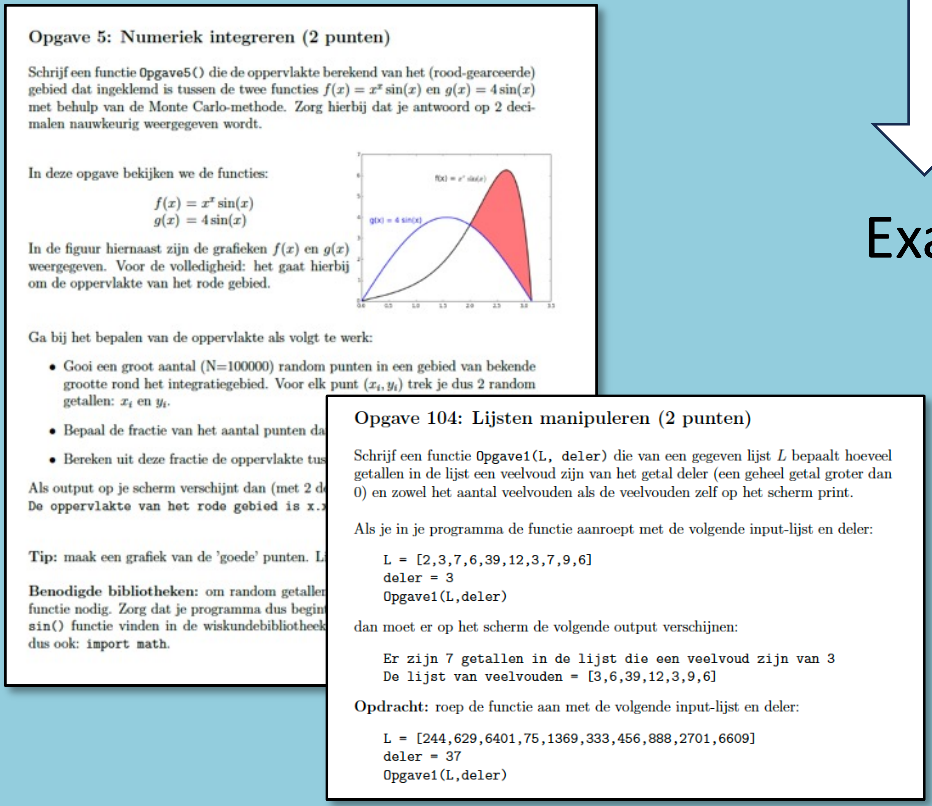
Family, friends, fellow students, Google, ChatGPT, ...

We use tools to cross-correlate code between students (and can do among years)

Exam (60%)

Final mark: homework 40% and exam 60%

Exam: 5 small exercises, 2,5 hours, no internet



Opgave 5: Numeriek integreren (2 punten)

Schrijf een functie `Opgave5()` die de oppervlakte berekend van het (rood-gearceerde) gebied dat ingeklemd is tussen de twee functies $f(x) = x^x \sin(x)$ en $g(x) = 4\sin(x)$ met behulp van de Monte Carlo-methode. Zorg hierbij dat je antwoord op 2 decimalen nauwkeurig weergegeven wordt.

In deze opgave bekijken we de functies:

$$f(x) = x^x \sin(x)$$
$$g(x) = 4\sin(x)$$

In de figuur hiernaast zijn de grafieken $f(x)$ en $g(x)$ weergegeven. Voor de volledigheid: het gaat hierbij om de oppervlakte van het rode gebied.

Ga bij het bepalen van de oppervlakte als volgt te werk:

- Gooi een groot aantal (N=100000) random punten in een gebied van bekende grootte rond het integratiegebied. Voor elk punt (x_i, y_i) trek je dus 2 random getallen: x_i en y_i.
- Bepaal de fractie van het aantal punten da
- Bereken uit deze fractie de oppervlakte tus

Als output op je scherm verschijnt dan (met 2 d
`De oppervlakte van het rode gebied is x.x`

Tip: maak een grafiek van de 'goede' punten. L

Benodigde bibliotheken: om random getallen
functie nodig. Zorg dat je programma dus begin
`sin()` functie vinden in de wiskundebibliotheek
dus ook: `import math`.

Opgave 104: Lijsten manipuleren (2 punten)

Schrijf een functie `Opgave1(L, deler)` die van een gegeven lijst L bepaalt hoeveel getallen in de lijst een veelvoud zijn van het getal deler (een geheel getal groter dan 0) en zowel het aantal veelvouden als de veelvouden zelf op het scherm print.

Als je in je programma de functie aanroept met de volgende input-lijst en deler:

```
L = [2,3,7,6,39,12,3,7,9,6]
deler = 3
Opgave1(L,deler)
```

dan moet er op het scherm de volgende output verschijnen:

```
Er zijn 7 getallen in de lijst die een veelvoud zijn van 3
De lijst van veelvouden = [3,6,39,12,3,9,6]
```

Opdracht: roep de functie aan met de volgende input-lijst en deler:

```
L = [244,629,6401,75,1369,333,456,888,2701,6609]
deler = 37
Opgave1(L,deler)
```

Note: need to pass the exam separately!

Summary:

- Mix advanced calculator & computational thinking
- Focus on physics problems, not programming itself
- Code review was a good choice
- Create a programming 'line' in curriculum

https://progns.proglab.nl/

New skills,
new possibilities

Now what?

1. Think about how to improve and find ways to share more widely High schools, other disciplines, international? **Invest in it or not?**

2. Expand into my own research field (statistics). Many courses too difficult/specific. Need to disuss that with like-minded people like Troels. **Invest in it or not?**

Exercise

Are we missing something?

Any ideas for a new module, or one adapted to your needs