DTU

Jakob Lemvig

Python i indledende universitetsmatematik

Nyt Polyteknisk Grundlag på DTU

Rammerne:

- "Nye" Kurser i: Prog., Mat., Fysik, Kemi, Stat., Bio.
- Ambitiøst nyt programmeringkursus i Python på 1. semester.
- Matematik 1 skal deles i to 10 ECTS kurser. Diskret Mat. skal indgå i stedet for Vektorkalkulus.
- Computational Thinking skal understøttes af Prog. og Mat.
- Skal udbydes fra Efterår 2023

Pilot-projekt

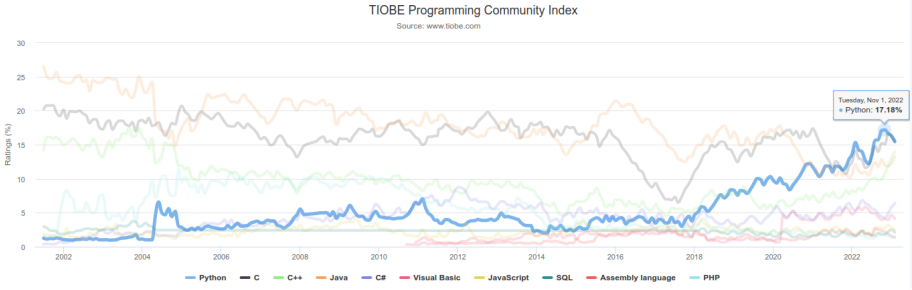
- Spørgsmål: Kan man bruge Python i stedet for Maple som CAS-værktøj?

Pilot-projekt

- Spørgsmål: Kan man bruge Python i stedet for Maple som CAS-værktøj?
- Min baggrund mht Programmingssprog: Næsten 20 års brug af Matlab og Maple. Jeg har kun ca 2 års erfaring med Python.
- Team: Hans Henrik Hermansen, Christian Mikkelstrup, Jakob Lemvig, Ulrik E. Pedersen
- Studieretninger: Matematik og Tekno. & Kunstig intelligens og Data (ca 160 studerende)
- Setup: 01005 Matematik 1 kører som under Maple. Intet nyt Prog. kursus i Python på 1. semester.
- De studerende har generelt meget lidt erfaring med programmering og har dårlig computerforståelse.

Hvorfor Python?

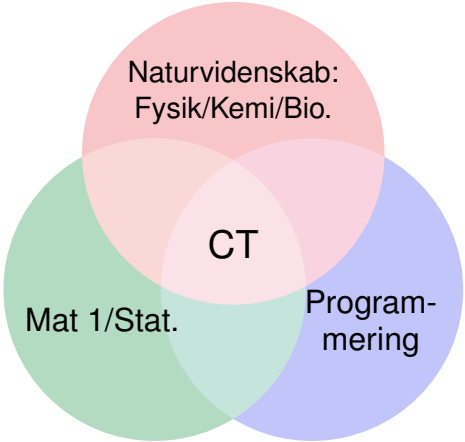
- Det er verdens mest populære progrmmeringssprog, jf `https://www.tiobe.com/tiobe-index/`



Hvorfor Python?

- Det er verdens mest populære progrmmeringssprog, jf `https://www.tiobe.com/tiobe-index/`
- Det er Open-Source, og koden er human-readable
- *Python is great for data science, AI programming, statistical programs, research projects, web sites, small glue programs and learning how to program*
- *Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.*

Polyteknisk Grundlag på DTU



- “Simpelt” setup med kun ét programmeringssprog.
- Prog. er forudsætning for CT og dermed for “alle” andre PG kurser
- Synergi ml. kurserne omkring CT
- Dogme: CT skal indgå i alle nævnte kurser

Python som CAS

- CAS-værktøjerne Maple, Mathematica, Sage og Python/SymPy har meget ens brugeroplevelser.

Python:

```
from sympy import *
init_printing()
```

[1] ✓ 0.3s

```
A = Matrix(2,3,[1,2,3,4,5,6])
A
```

[2] ✓ 1.0s

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
# integrate(sqrt(9*u**4 + 16*u**2),u) giver fejl
```

[3] ✓ 0.0s

```
u = symbols('u')
integrate(sqrt(9*u**4 + 16*u**2),u)
```

[4] ✓ 1.0s

$$\int \sqrt{9u^4 + 16u^2}\, du$$

```
ū = symbols('u', positive=True)
integrate(sqrt(9*ū**4 + 16*ū**2),ū)
```

[5] ✓ 0.3s

$$\frac{u^3\sqrt{9u^2+16}}{3} + \frac{16\sqrt{9u^2+16}}{27}$$

```
x = symbols('x', real=True)
ligning = Eq(cos(x), S(1)/2)
solveset(ligning)
```

[6] ✓ 0.3s

$$\left\{2n\pi + \frac{5\pi}{3} \,\middle|\, n \in \mathbb{Z}\right\} \cup \left\{2n\pi + \frac{\pi}{3} \,\middle|\, n \in \mathbb{Z}\right\}$$

Maple:

Start.mw | *maple_eksempler.mw

Text | Nonexecutable Math | Math | Maple Input

```
> with(LinearAlgebra):
> A := Matrix(2,3,[1,2,3,4,5,6])
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad (1)$$

```
> int(sqrt(9*u^4+16*u^2),u)
```

$$\frac{(9u^2+16)\sqrt{9u^4+16u^2}}{27u} \quad (2)$$

```
> ligning := cos(x) = 1/2
```

$$ligning := \cos(x) = \frac{1}{2} \quad (3)$$

```
> solve(ligning)
```

$$\frac{\pi}{3} \quad (4)$$

```
> solve(ligning, AllSolutions=true)
```

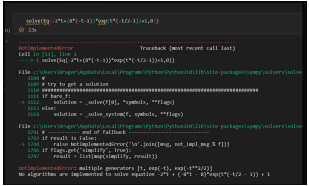
$$-\frac{2}{3}\pi\_B3\sim + \frac{1}{3}\pi + 2\pi\_Z3\sim \quad (5)$$

```
>
```

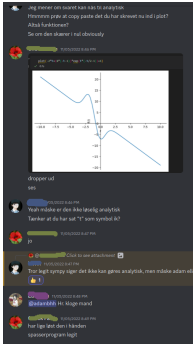
Python som CAS

- CAS-værktøjerne Maple, Mathematica, Sage og Python/SymPy har meget ens brugeroplevelser.
- Der er de samme problemer for de studerende som vi kender: Manglende matematisk forståelse

Fejlbesked:



Frustration:



Pros/Cons for Python

Ulmeper med Python:

- Python er ikke Maple. Mange undervisere skal omskoles.
- Manglende forståelse hos studerende for alm. computer-brug: Selve installationen på egen computer bliver det største problem. Potentielt set et stort problem for kursus med 1300 studerende.

Pros/Cons for Python

Ulmeper med Python:

- Python er ikke Maple. Mange undervisere skal omskoles. Efteruddannelse
- Manglende forståelse hos studerende for alm. computer-brug: Selve installationen på egen computer bliver det største problem. Potentielt set et stort problem for kursus med 1300 studerende. Python i skyen: `https://classroom.github.com/a/5I5AbEhe`

Pros/Cons for Python

Ulmeper med Python:

- Python er ikke Maple. Mange undervisere skal omskoles. Efteruddannelse
- Manglende forståelse hos studerende for alm. computer-brug: Selve installationen på egen computer bliver det største problem. Potentielt set et stort problem for kursus med 1300 studerende. Python i skyen: `https://classroom.github.com/a/5I5AbEhe`

Fordele med Python:

- Det er et brugt programmeringssprog (modsat Maple)
- Typisk DTU projektarbejde: Matematik modellering (CAS), (Numeriske) Beregninger, Præsentation (GUI). Alt kan laves i Python.

Det nye Matematik 1

7 Tema-øvelser & Stort afsluttende Projekt

Opgaver/projekter med tværfaglige elementer:

- Naturvidenskaben
- Ingeniørvidenskaben
- Computer Science

De bør så vidt muligt indeholde Prog./Python med klare elementer af CT som underbygger den matematiske forståelse/tænkning

Mini CT-opgaver

På dage med underviser indgår mini CT-opgaver. Fx skrivning/forståelse af pseudo-kode eller Python kode. Disse opgaver skal underbygge den matematiske forståelse/tænkning.

Hvad er Computational Thinking (CT)?

- ISTE (International Society for Technology in Education): *Computational thinking is a problem-solving process that includes (but is not limited to) formulating problems, analyzing and representing data, and algorithmic thinking.*
- Kalelioglu, Gulbahar, and Kukul (2016) defines the following framework for CT:
  1. identifying the problem: abstraction, decomposition
  2. gathering, representing and analysing data: data collection, analysis, pattern recognition, conceptualising, data representation
  3. generating, selecting and planning solutions: mathematical reasoning, building algorithms and procedures, parallelisation
  4. implementing solutions: Automation, modelling and simulations
  5. assessing solutions and continue for improvement: testing, debugging and generalisation

Hvad er Computational Thinking (CT)?

En simplere og mere brugbar definition er:

Citat fra Steve Wolfram

But first, let's try to define what we mean by "computational thinking". As far as I'm concerned, its intellectual core is about formulating things with enough clarity, and in a systematic enough way, that one can tell a computer how to do them. Mathematical thinking is about formulating things so that one can handle them mathematically, when that's possible. Computational thinking is a much bigger and broader story, because there are just a lot more things that can be handled computationally.

Generelt om CAS og CT i Matematik-undervisning

Farer:

- CAS som overdommer (over matematikkens egne regler)
- CT går i de samme fælder som vi har gjort med CAS (fx fancy grafik og for kompliceret kode)
- Computational Thinking uden thinking: "Metoder" og "algoritmer" uden forståelse
- CT bliver til Numeriske Algoritmer (overdrevet fokus på problemer der udspringer fra den Matematiske Analyse)

Generelt om CAS og CT i Matematik-undervisning

Farer:

- CAS som overdommer (over matematikkens egne regler)
- CT går i de samme fælder som vi har gjort med CAS (fx fancy grafik og for kompliceret kode)
- Computational Thinking uden thinking: "Metoder" og "algoritmer" uden forståelse
- CT bliver til Numeriske Algoritmer (overdrevet fokus på problemer der udspringer fra den Matematiske Analyse)

Mål:

- Vi skal lære dem det der er svært – ikke lav-praktiske digitale kompetencer.
- (Fortsat) fokus på matematisk indsigt