Programmering i skolen

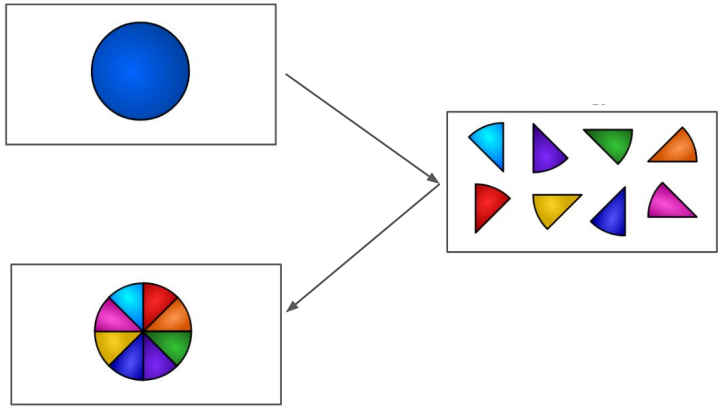
Et kræsjkurs i Python for realsfagslærere: Del 2

Tekna

kodeskolen

simula

Algoritmisk tankegang trekkes inn som et kjerneelement i matematikkfaget



"Vi har vektlagt algoritmisk tenkning fordi dette er en viktig problemløsningsstrategi.

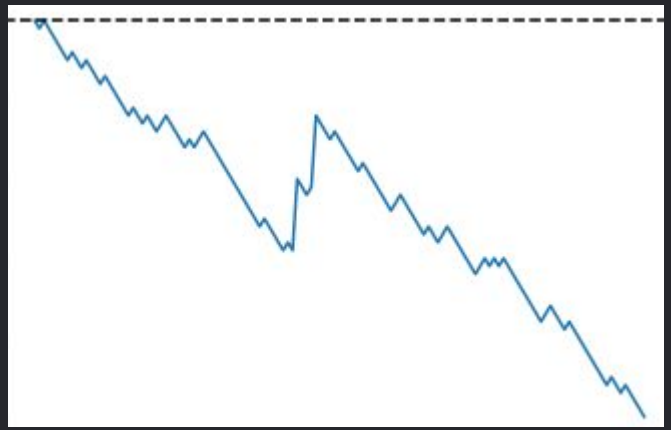
Når elevene bruker programmering til å utforske og løse problemer, kan det være et godt verktøy for å utvikle matematisk forståelse."

Med programmering kan man enkelt simulere problemer fra sannsynlighetsteori

S1: bruke digitale verktøy til å simulere utfall i stokastiske forsøk

S2: bruke digitale verktøy til å simulere utfall i statistiske fordelinger

Med programmering kan man enkelt simulere problemer fra sannsynlighetsteori



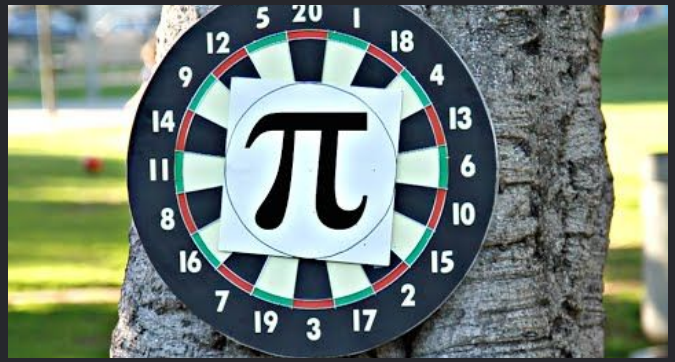
```python
def pengespill():
    total = kast_2d6()
    if total <= 8:
        return -10
    elif total <= 11:
        return 10
    else:
        return 90

# Telle variabel
penger = 500

# Liste for å huske resultatene over tid
pengehistorikk = []
pengehistorikk.append(penger)

# Løkke for å gjenta spillet helt til vi går tom for penger
while penger > 0:
    penger += pengespill()
    pengehistorikk.append(penger)

n = len(pengehistorikk)
# Plot resultatet
plot(pengehistorikk)
axhline(500, color='black', linestyle='--')
show()
print(f'Du spilte {n} ganger før du gikk tom for penger')

```

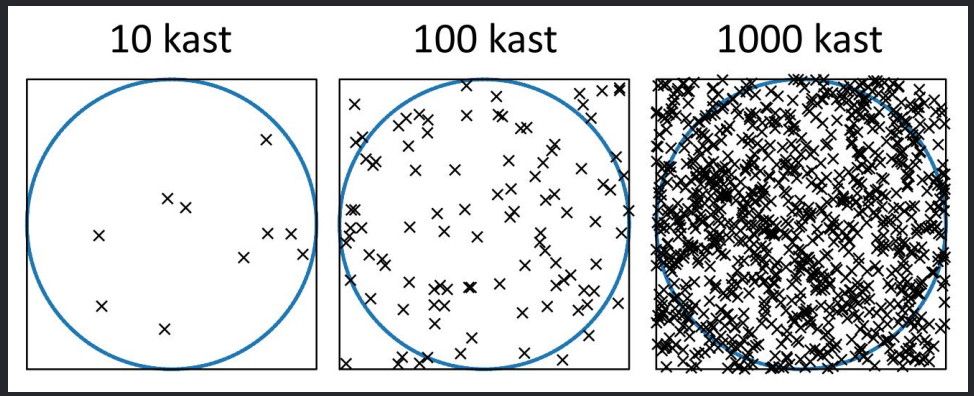
Med programmering kan man enkelt simulere problemer fra sannsynlighetsteori

S1: bruke digitale verktøy til å simulere utfall i stokastiske forsøk

S2: bruke digitale verktøy til å simulere utfall i statistiske fordelinger

Med programmering kan man enkelt simulere problemer fra sannsynlighetsteori






```python
from random import uniform
from math import sqrt

antall_kast = 10
antall_treff = 0

for kast in range(antall_kast):
    # Kast en pil
    x = uniform(-1, 1)
    y = uniform(-1, 1)

    # Sjekk om den traff
    avstand = sqrt(x**2 + y**2)
    if avstand <= 1:
        antall_treff += 1


# Estimer pi basert på kastene
pi = 4*antall_treff/antall_kast

# Skriv ut resultater
print(f"Antall kast: {antall_kast}")
print(f"Antall treff: {antall_treff}")
print(f"Estimert pi: {pi}")
```

Programmering kan ta i bruk numeriske tilnærminger av den deriverte til å simulere fysiske prosesser

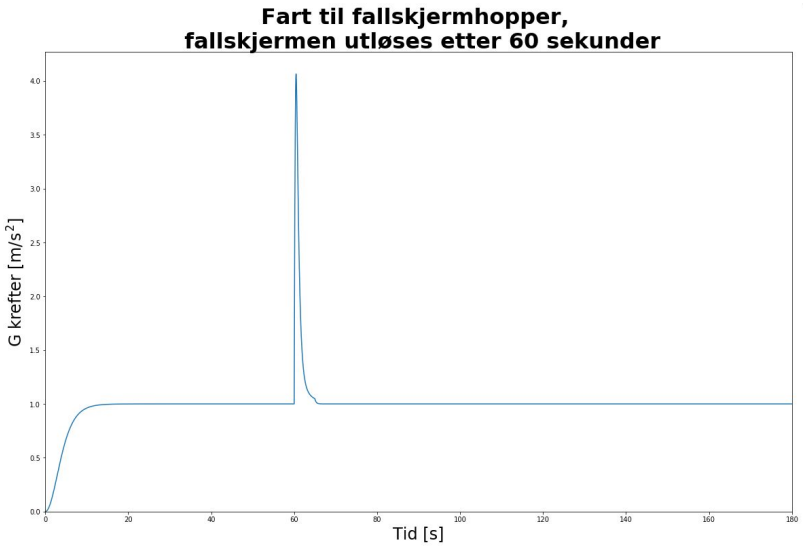
R1: bruke programmering til å gjøre beregninger og utforsking av egenskaper til funksjoner

S2: bruke programmering til å utforske rekursive sammenhenger og presentere egne framgangsmåter

Programmering kan ta i bruk numeriske tilnærminger av den deriverte til å simulere fysiske prosesser

```
30 # Simuler de forste 60 sekundene
31 for i in range(0, 60/dt):
32     t[i+1] = t[i] + dt
33     v[i+1] = v[i] + a(v[i])*dt
34     gforces[i] = 1 - a(v[i])/g
35
36 # Simulerer de neste 5 sekundene
37 for i in range(60/dt, 65/dt):
38     C += (C_p-C)/(5/dt)
39     A += (A_p-A)/(5/dt)
40
41     t[i+1] = t[i] + dt
42     v[i+1] = v[i] + a(v[i])*dt
43     gforces[i] = 1 - a(v[i])/g
44
45 # Simuler de siste 115 sekundene
46 for i in range(65/dt, 180/dt):
47     t[i+1] = t[i] + dt
48     v[i+1] = v[i] + a(v[i])*dt
49     gforces[i] = 1 - a(v[i])/g
50
```

Programmering kan ta i bruk numeriske tilnærminger av den deriverte til å simulere fysiske prosesser

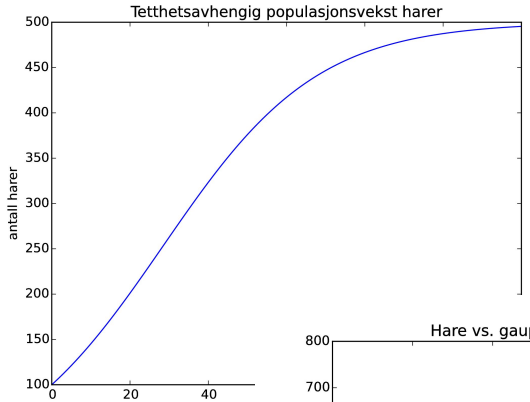


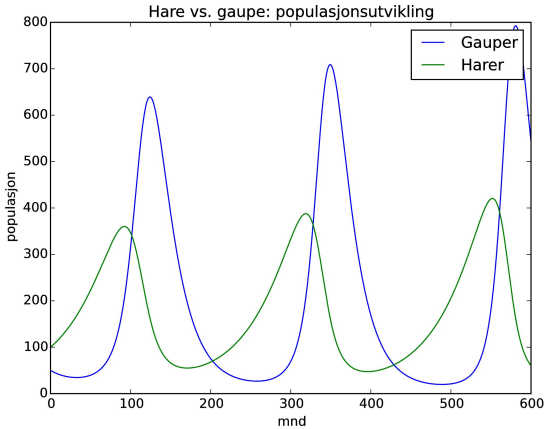
Med programmering kan vi simulere forskjellige typer populasjonsvekst

Kjerneelement:
Elevene skal forstå, skape og bruke teknologi, inkludert programmering og modellering, i arbeid med naturfag.

Naturfag vg1: bruke og vurdere programmer som løser eller simulerer naturfaglige problemstillinger.

Med programmering kan vi simulere forskjellige typer populasjonsvekst



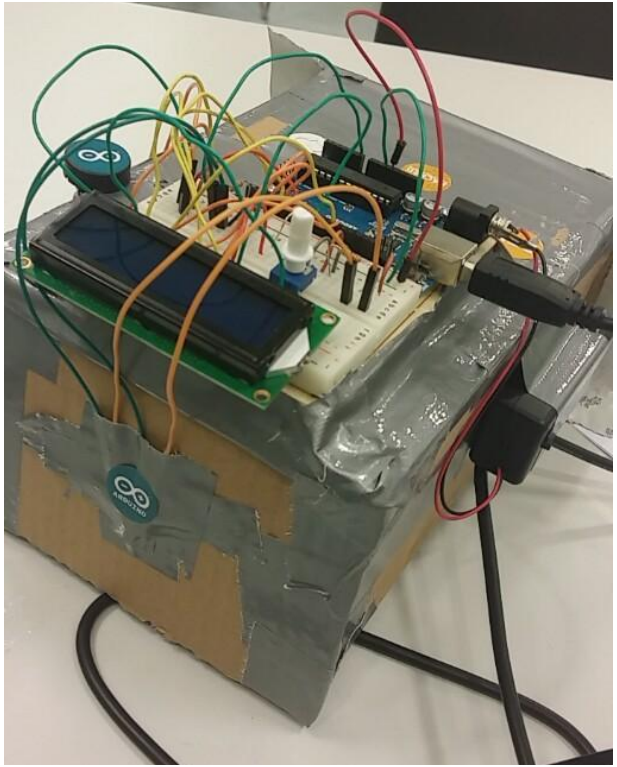



```python
import numpy as np
n = 12*50    #antall tidsintervaller
y0 = 100     #antall byttedyr når vi starter
x0 = 50    #antall rovdyr når vi starter
index_set = range(n+1)

x = np.zeros(len(index_set))
y = np.zeros(len(index_set))


a = 0.05   # dødsrate gauper
b = 0.0003 # reproduksjonsrate gauper

c = 0.02   # vekstrare harer
d = 0.0001 # dødsrate harer


y[0] = y0
x[0] = x0
for k in index_set[:-1]:
    #print y[k]
    y[k+1] = y[k] + c*y[k] - d*y[k]*x[k]
    x[k+1] = x[k] - a*x[k] + b*x[k]*y[k]
```

Arduino og micro:bit kan la elever lage programmer som regner på data hentet fra sensorer

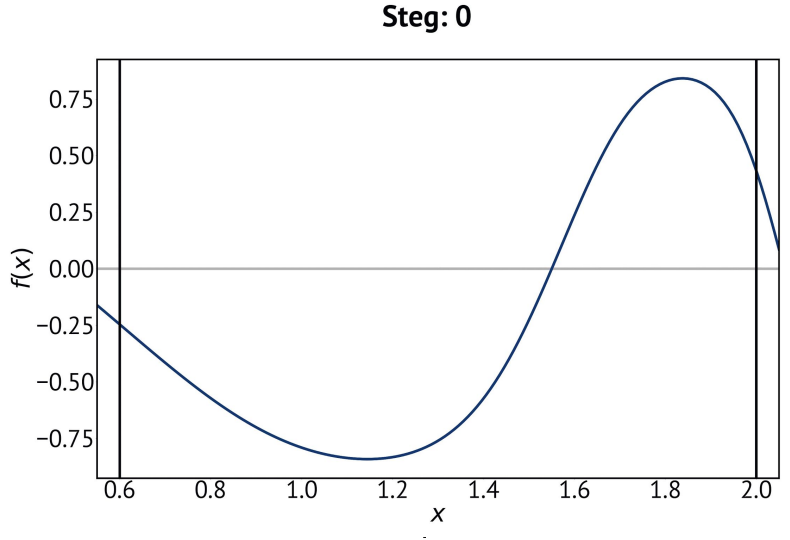
Digitale ferdigheter i naturfag er å kunne bruke digitale verktøy til å utforske, registrere, beregne, visualisere, programmere, modellere, dokumentere og publisere data fra forsøk, feltarbeid og andres studier.

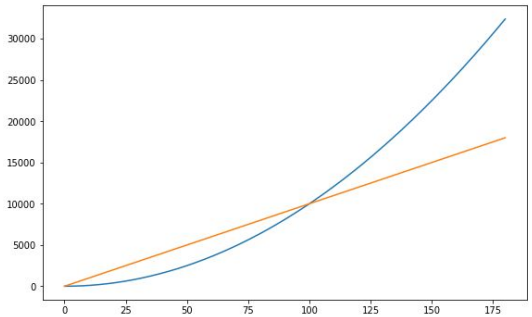
Naturfag VG1: utforske en selvvalgt naturfaglig problemstilling, presentere funn og argumentere for valg av metoder



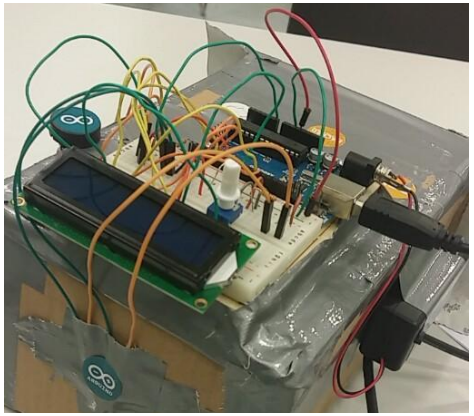
Med programmering kan vi finne løsninger på likninger vi ikke kan løse for hånd

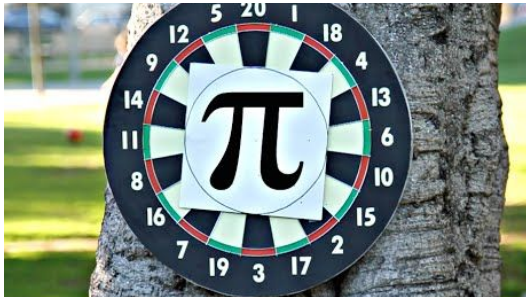
1T: formulere og løyse problem ved hjelp av ulike problemløysingstrategiar og digitale verktøy

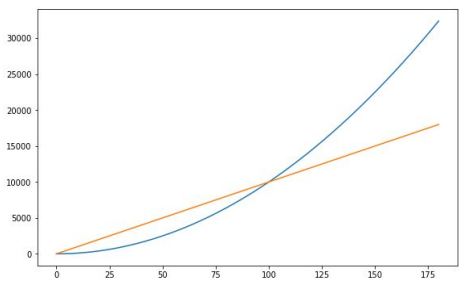




Som vi har sett har programmering mange anvendelser i realfag

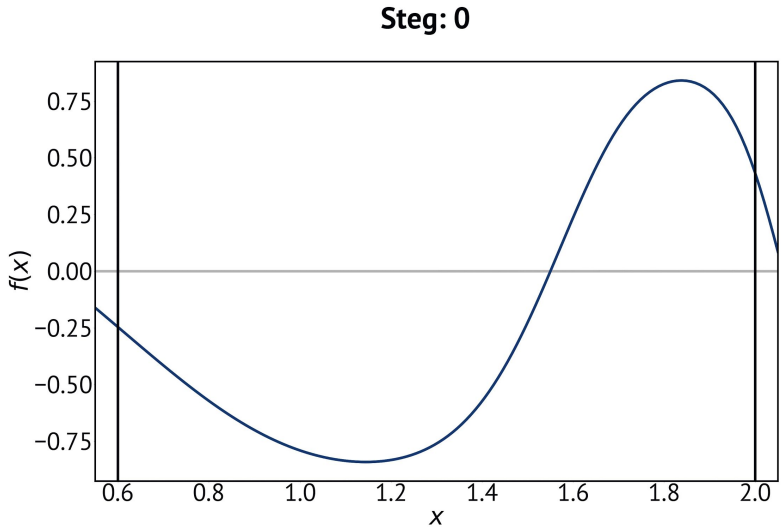






I dag: Videre Python og et opplegg til klasserommet

- Funksjoner
- Plotting
- Prosjekt: Programmere en likningsløser som bruker *halveringsmetoden*



Det finnes overraskende lite forskning innen informatikkdidaktikk

PLOS | COMPUTATIONAL BIOLOGY

EDUCATION

Ten quick tips for teaching programming

Neil C. C. Brown[1☯], **Greg Wilson**[2☯*]

1 Department of Informatics, King's College London, London, United Kingdom, **2** DataCamp, Toronto, Ontario, Canada

☯ These authors contributed equally to this work.
* gvwilson@third-bit.com

This is a *PLOS Computational Biology* Education paper.

Introduction

Research from educational psychology suggests that teaching and learning are subject-specific activities [1]: learning programming has a different set of challenges and techniques than learning physics or learning to read and write. Computing is a younger discipline than mathematics, physics, or biology, and while there have been correspondingly fewer studies of how best to teach it, there is a growing body of evidence about what works and what doesn't. This paper presents 10 quick tips that should be the foundation of any teaching of programming, whether formal or informal.

These tips will be useful to anyone teaching programming at any level and to any audience.