Å lære bort programmering

Hva, hvordan og hvorfor?

Tekna

kodeskolen

simula

slido

Hvor er du nå?

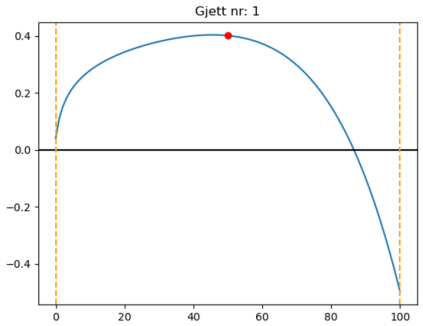
ⓘ Start presenting to display the poll results on this slide.

slido

Hva er din erfaring med programmering?

ⓘ Start presenting to display the poll results on this slide.

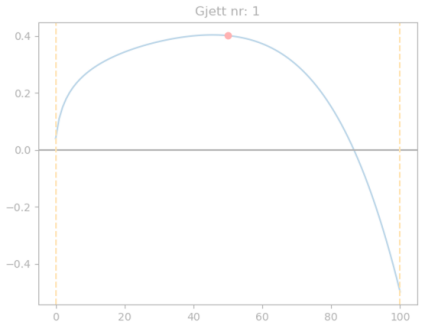
Hva er programmering?



Hva skal elevene lære?

Tips til kodeundervisning

Hva er programmering?



Hva skal elevene lære?

Tips til kodeundervisning

slido

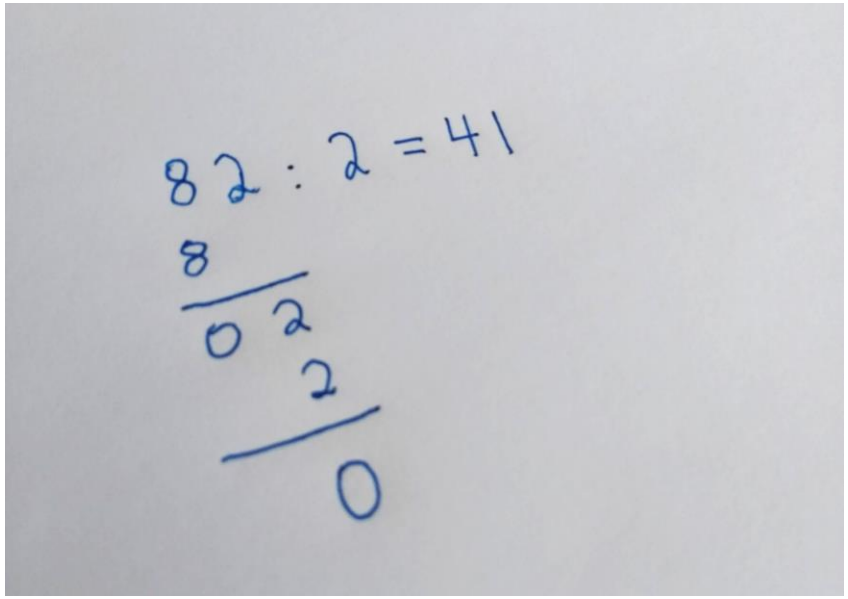
Hva forbinder du med ordet "algoritme"?

ⓘ Start presenting to display the poll results on this slide.

En algoritme er en nøyaktig serie med instrukser for å løse et problem

En algoritme er en nøyaktig serie med instrukser for å løse et problem

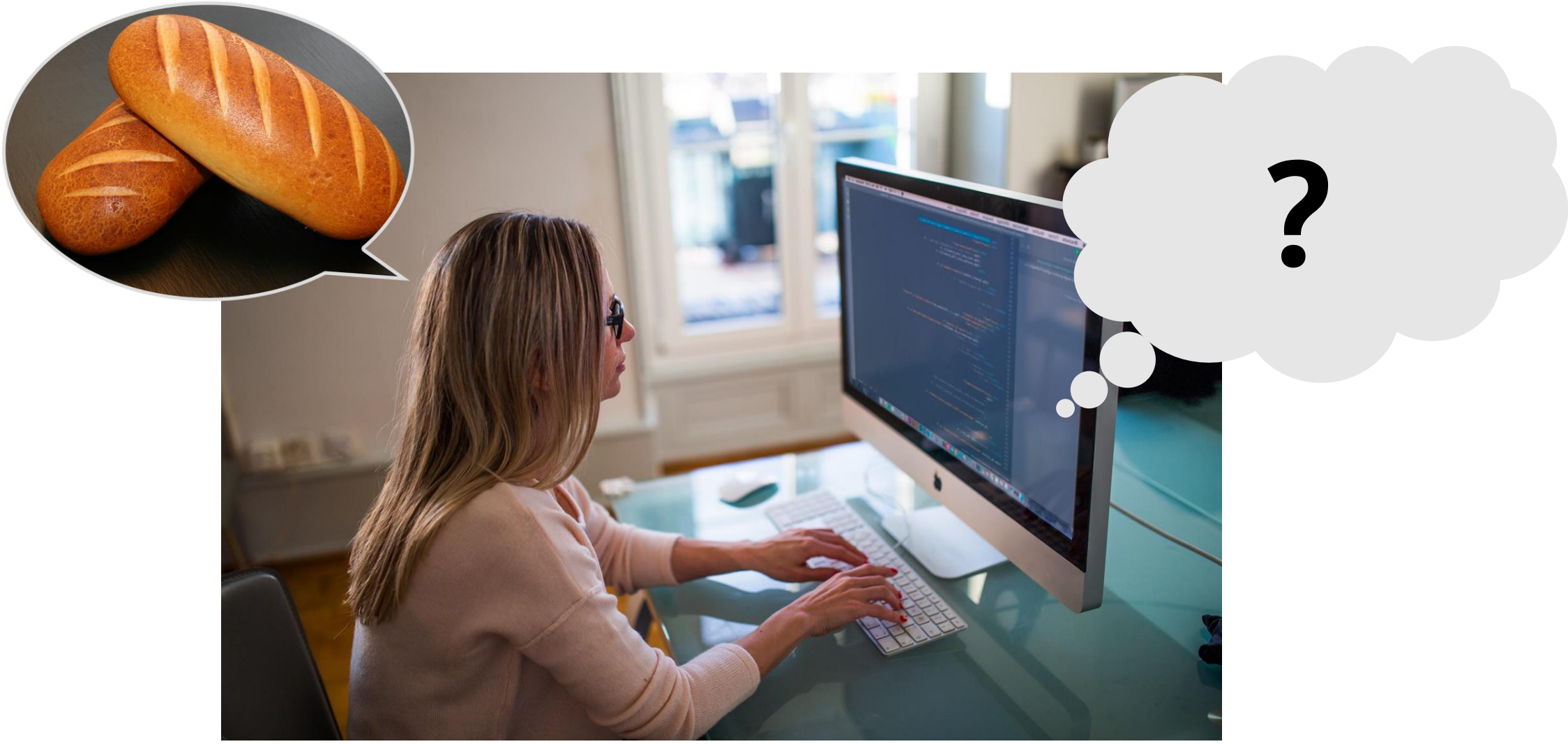
I matematikkfaget finner du mange eksempler på algoritmer



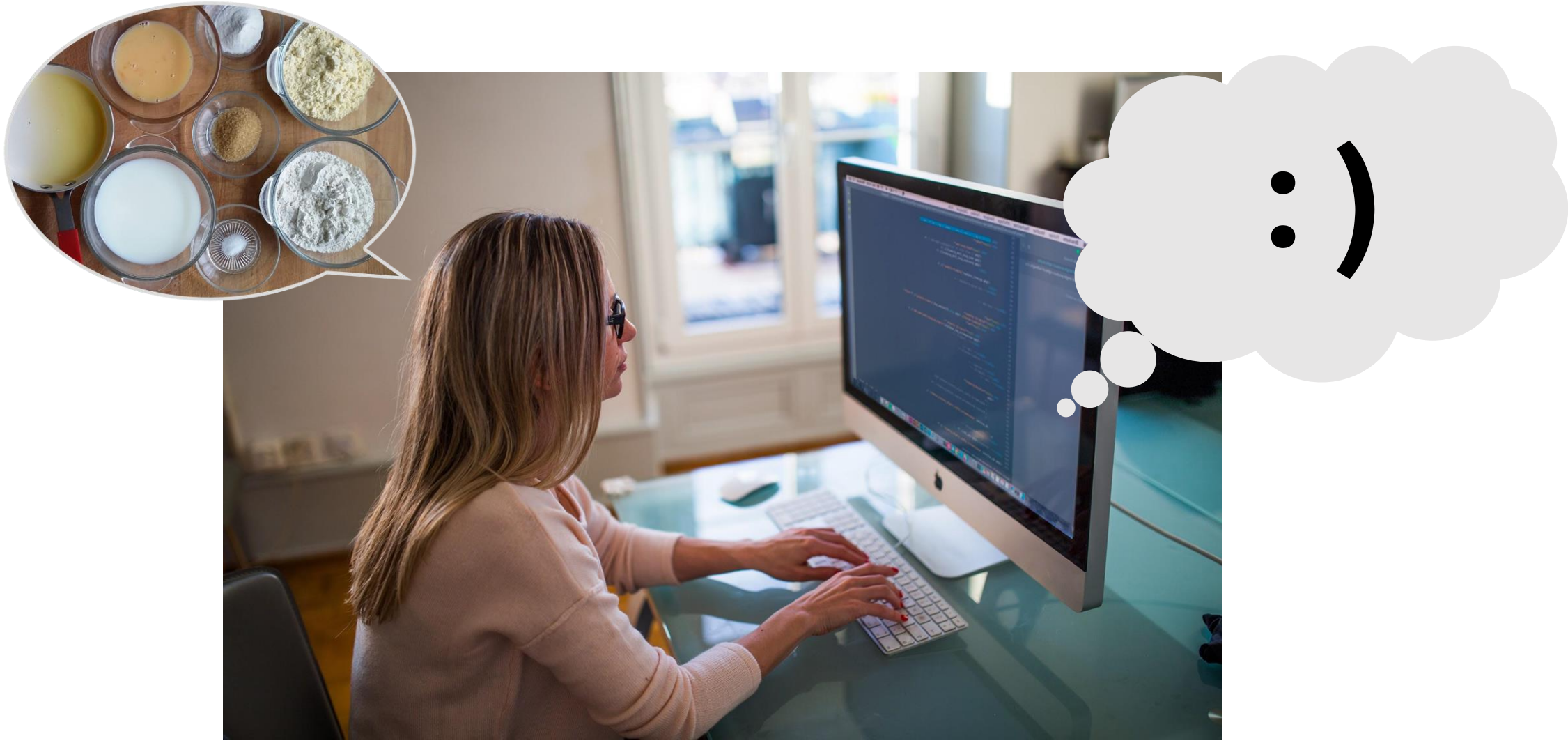
Et data program er en algoritme skrevet slik at en datamaskin kan det forstå det


```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]
```

Datamaskinen er dum, så den trenger nøyaktige instrukser



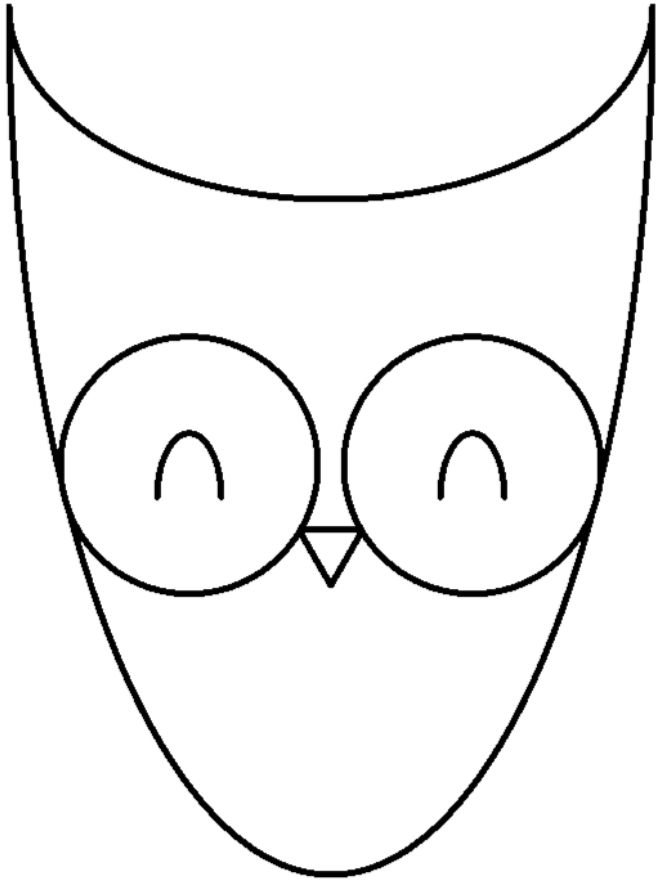
Datamaskinen er dum, så den trenger nøyaktige instrukser



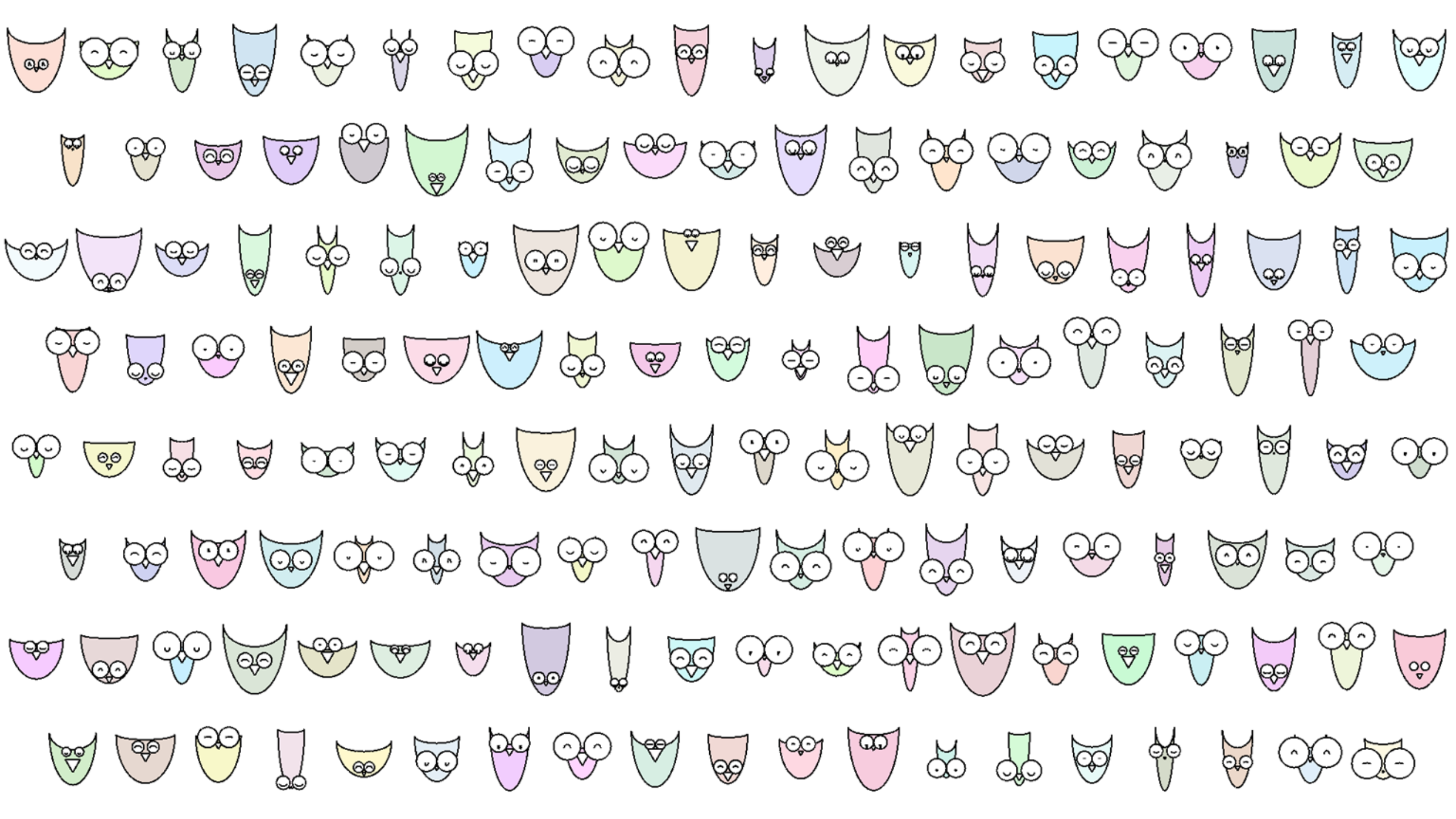
slido

Hva tegnet du?

ⓘ Start presenting to display the poll results on this slide.



```python
from p5 import *

def setup():
    size(1920, 1080)
    stroke_weight(5)

def equilateral_triangle(center_x, center_y, width):
    triangle((center_x, center_y),
             (center_x - width/2, center_y - 0.5*sqrt(3)*width),
             (center_x + width/2, center_y - 0.5*sqrt(3)*width))

def eye(center_x, center_y, radius, eyelid_width, eyelid_height):
    circle(center_x, center_y, radius, mode="CENTER")
    arc((center_x, center_y + 0.25*eyelid_height), eyelid_width, eyelid_height, PI, TWO_PI)

def body(center_x, top_y, width, height, feather_height):
    arc((center_x, top_y), width, height, 0, PI)
    arc((center_x, top_y), width, feather_height, 0, PI)

def owl(center_x, center_y, owl_width, owl_height, feather_height, nose_offset, nose_width,
        eye_offset, eye_radius, eye_distance, eyelid_width, eyelid_height):
    top_y = center_y - owl_height / 4
    nose_offset += feather_height / 2
    eye_center_y = top_y + nose_offset - eye_offset - nose_width*0.5*sqrt(3)

    body(center_x, top_y, owl_width, owl_height, feather_height)
    equilateral_triangle(center_x, top_y + nose_offset, nose_width)
    offset = eye_radius/2 + eye_distance / 2
    eye(center_x + offset, eye_center_y, eye_radius, eyelid_width, eyelid_height)
    eye(center_x - offset, eye_center_y, eye_radius, eyelid_width, eyelid_height)

def draw():
    background(255, 255, 255)
    owl(width / 2, height / 2, 500, 1350, 300, 300, 50, 50, 200, 20, 50, 100)
    save("single_owl.png")

run()
```

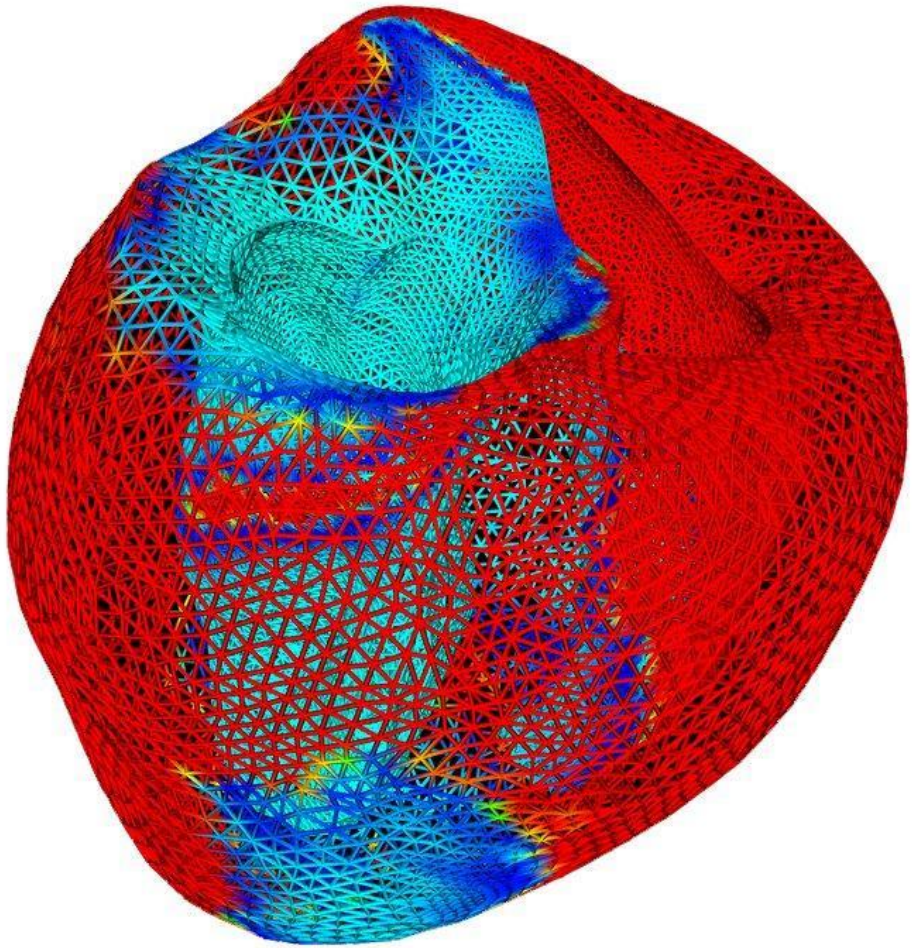


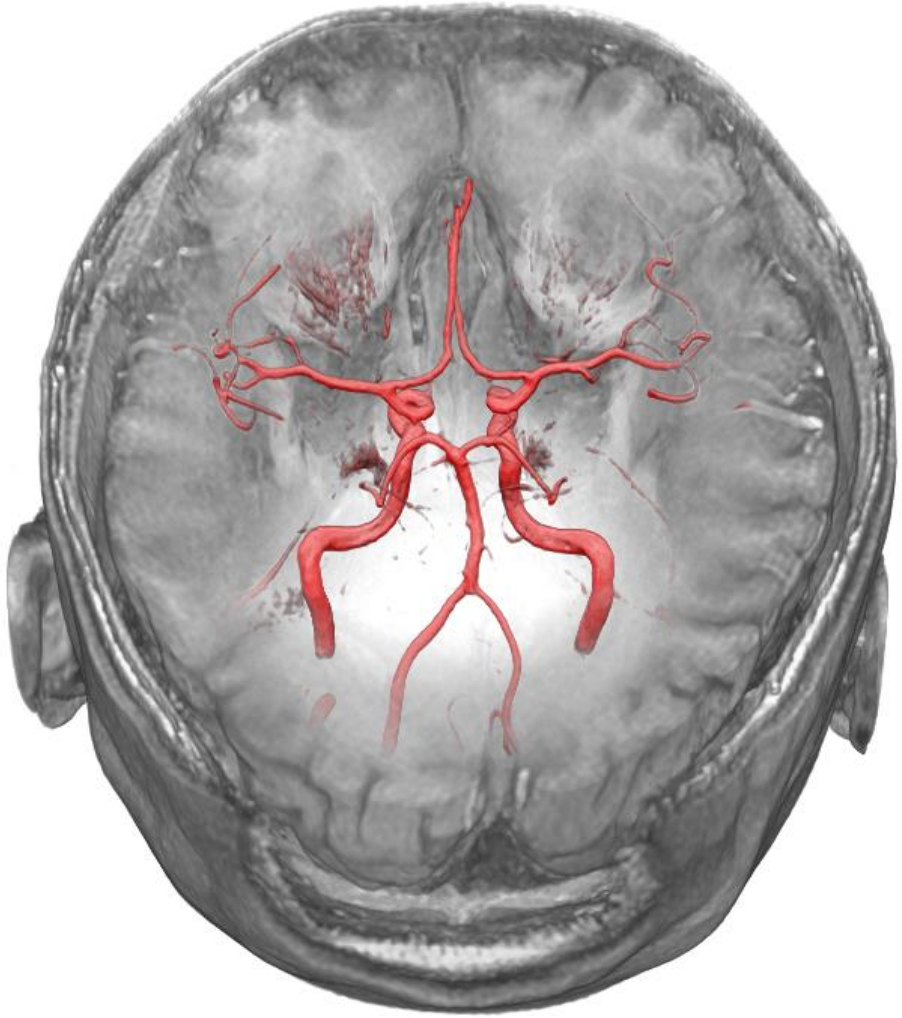
Algoritmisk tankegang handler blant annet om å bryte opp komplekse problemer i små biter, og løse dem steg for steg.

Samfunnet og yrkeslivet blir i større og større grad avhengig av digitale løsninger og verktøy

“Blir det ikke som å lære alle som skal kjøre bil å bli bilmekanikere?”

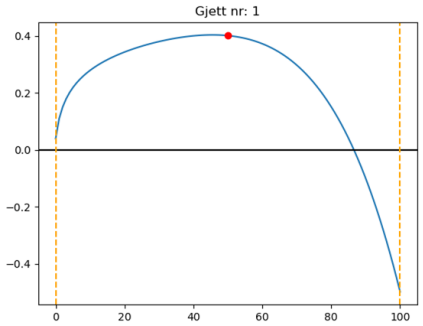
Koding er samfunnsnyttig, spesielt i kombinasjon med realfag





Matte er et verktøy for å løse problemer og programmering utvider hvilke problemer du kan løse

Hva er programmering?



Hva skal elevene lære?

Tips til kodeundervisning

Grunnleggende programmeringsferdigheter er lagt til barneskolen

Etter 2. trinn

- lage og følgje reglar og trinnvise instruksjonar i leik og spel

Etter 4. trinn

- lage algoritmar og uttrykkje dei ved bruk av variablar, vilkår og lykkjer

Etter 5. trinn

- lage algoritmar med bruk av variablar, vilkår og lykkjer og programmere desse

Etter 6. Trinn

- bruke variablar, lykkjer, vilkår og funksjonar i programmering til å utforske geometriske figurar og mønster

Programmeringsferdigheter rettet mot matematikken er lagt til ungdomsskolen

Etter 8. trinn

- utforske korleis algoritmar kan skapast, testast og forbetrast ved hjelp av programmering

Etter 9. trinn

- bruke programmering til å utforske matematiske eigenskapar og samanhengar

Etter 10. trinn

- bruke digitale verktøy til å utforske og samanlikne eigenskapar til ulike funksjonar

På ungdomsskolen er programmering og teknologi også et kjerneelement i naturfag

Kjerneelement

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

Etter 10. trinn

- bruke programmering til å simulere naturfaglige problemstillinger

På videregående blir programmering nevnt i naturfag og matematikk

Naturfag

- vurdere og lage programmer som modellerer naturfaglige fenomener

Matematikk 1T

- formulere og løyse problem ved hjelp av algoritmisk tenking, ulike problemløysingsstrategiar, digitale verktøy og programmering

Matematikk 1P

- identifisere variable storleikar i ulike situasjonar og bruke dei til utforsking og generalisering
- bruke digitale verktøy i utforsking og problemløysing knytt til eigenskapar ved funksjonar, og diskutere løysingane

Matematikk 2P

- forklare og bruke prosent, prosentpoeng og vekstfaktor til modellering av praktiske situasjonar med digitale verktøy

I de matematiske valgfagene er fokus på bruk av programmering for å utforske matematiske problemstillinger

Matematikk R1

- bestemme den deriverte i et punkt geometrisk, algebraisk og ved numeriske metoder, og gi eksempler på funksjoner som ikke er deriverbare i gitte punkter
- modellere og analysere eksponentiell og logistisk vekst i reelle datasett

Matematikk R2

- utforske rekursive sammenhenger ved å bruke programmering og presentere egne framgangsmåter
- utvikle algoritmer for å beregne integraler numerisk, og bruke programmering til å utføre algoritmene

Matematikk S1

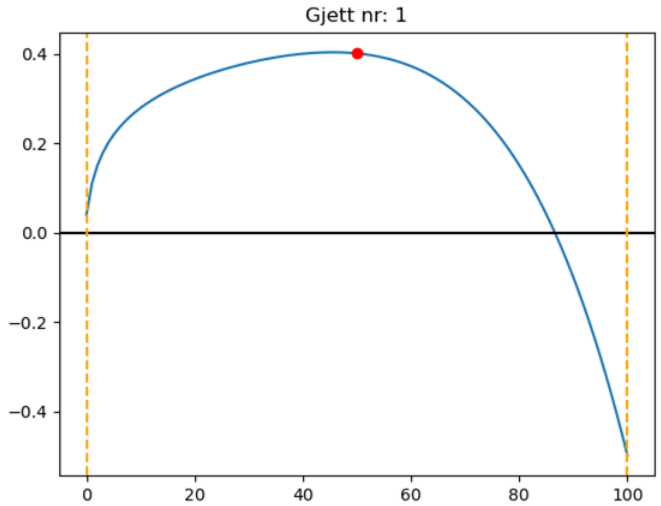
- bruke digitale verktøy til å simulere og utforske utfall i stokastiske forsøk, og forstå begrepet stokastiske variabler

Matematikk S2

- utforske rekursive sammenhenger ved å bruke programmering og presentere egne framgangsmåter
- modellere og analysere eksponentiell og logistisk vekst i reelle datasett
- simulere utfall i, utforske og tolke ulike statistiske fordelinger, og gi eksempler på reelle anvendelser av disse fordelingene

Gode programmeringsopplegg kan bruke kode til å løse problemer som ikke lett løses for hånd

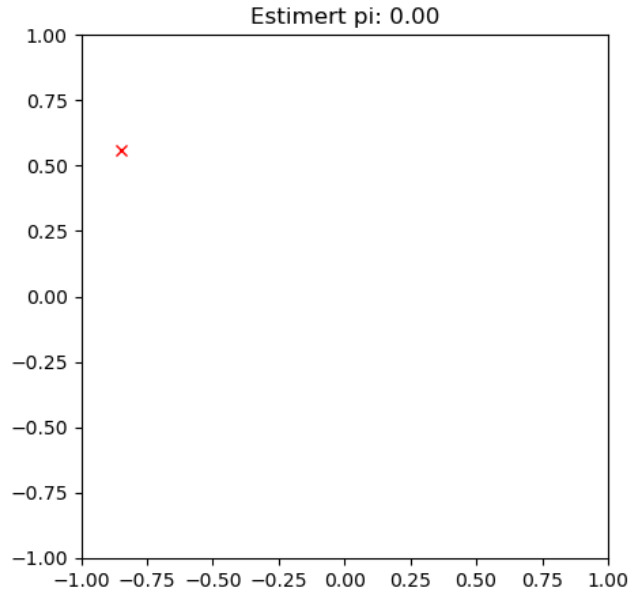
```
maks_tall = 100
min_tall = 0
feilterskel = 0.01

for forsøk in range(100):
    gjett = midtpunkt(min_tall, maks_tall)
    plot_søkeområde(x, y, gjett)

    if abs(f(gjett)) < feilterskel:
        break
    elif sign(f(gjett)) == sign(f(maks_tall)):
        maks_tall = gjett
    else:
        min_tall = gjett
```



[Webb et al. (2012)]

Gode programmeringsopplegg fremhever utforsking og kreativ problemløsning

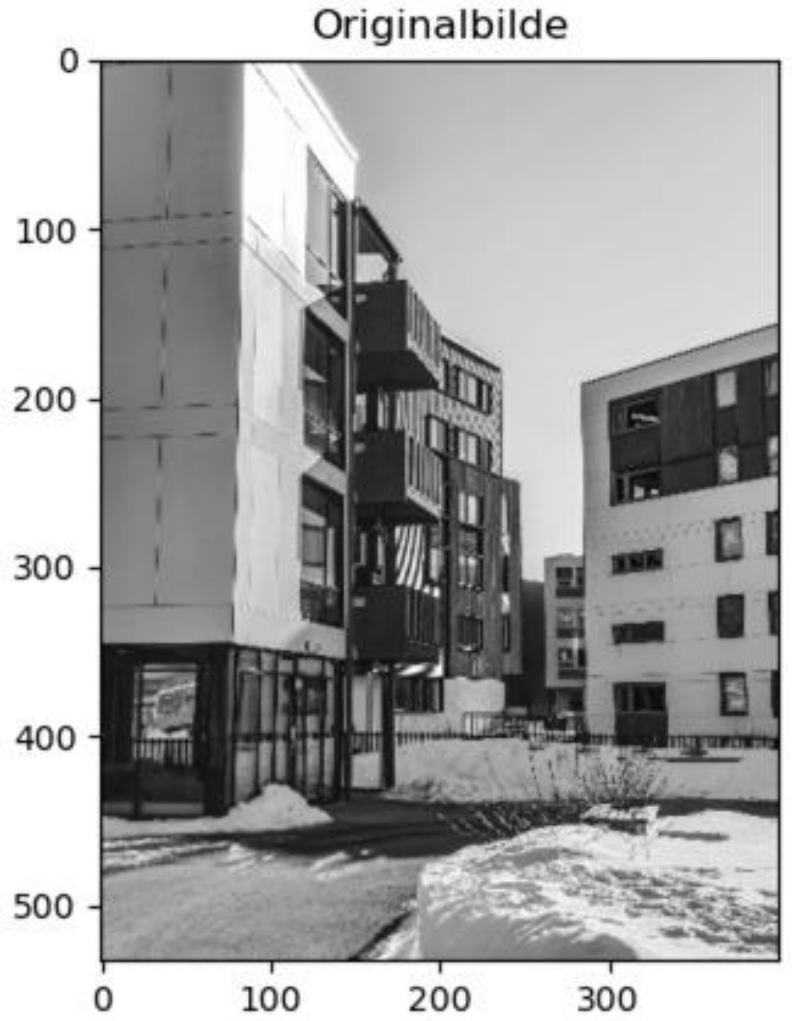
```
for kast in range(antall_kast):
    dart_x, dart_y = kast_dart()

    if traff_blinken(dart_x, dart_y):
        plot(dart_x, dart_y, "gx")
        antall_treff += 1
    else:
        plot(dart_x, dart_y, "rx")

    estimert_pi = 4 * antall_treff / (kast + 1)
    title(f"Estimert pi: {estimert_pi:.2f}")
    xlim(-1, 1)
    ylim(-1, 1)
    pause(0.1)
```



[Webb et al. (2012)]

Med programmering kan man bruke matematikk for å behandle ekte data

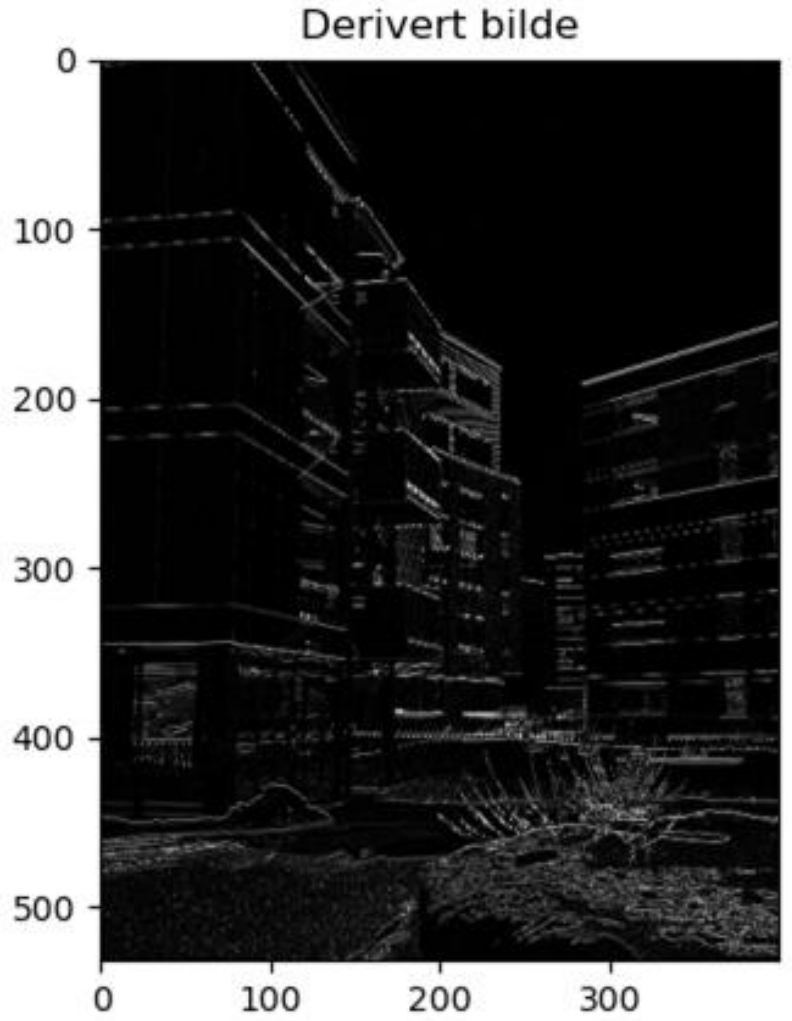
```
bilde = imread(“hus.png", as_gray=True)
imshow(bilde, cmap="gray")
title("Originalbilde")
show()

derivert_bilde = abs(deriver(bilde))
imshow(derivert_bilde, cmap="gray")
title("Derivert bilde")
show()
```



[Guzdial et al. (2013)]

Med programmering kan man bruke matematikk for å behandle ekte data

```
bilde = imread(“hus.png", as_gray=True)
imshow(bilde, cmap="gray")
title("Originalbilde")
show()

derivert_bilde = abs(deriver(bilde))
imshow(derivert_bilde, cmap="gray")
title("Derivert bilde")
show()
```

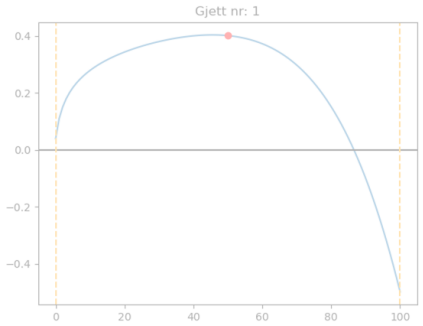


[Guzdial et al. (2013)]

Et godt programmeringsopplegg kan bestå av “kalkulatorprogram” som lar en utforske matematiske sammenhenger

```
from math import pi

radius = 3
kulevolum = 4/3 * pi * radius**3

print(f“En kule med radius {radius} har volumet {kulevolum}")
```

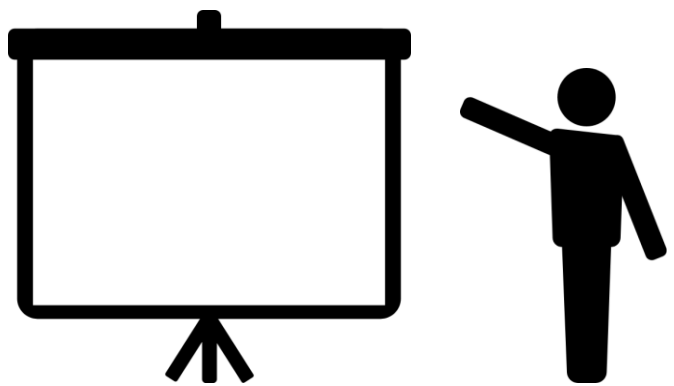
Hva er programmering?

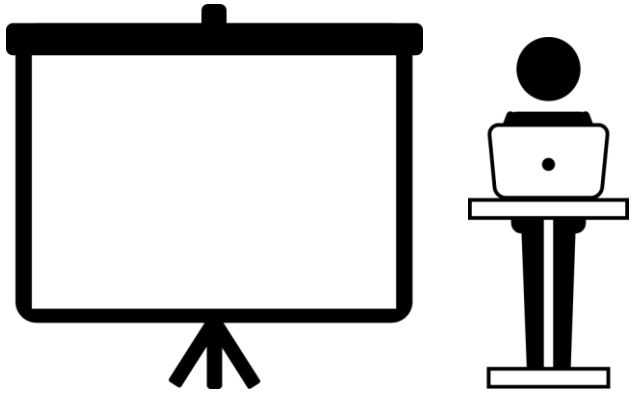


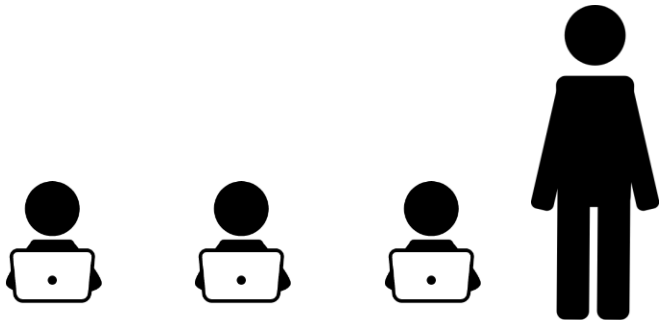
Hva skal elevene lære?

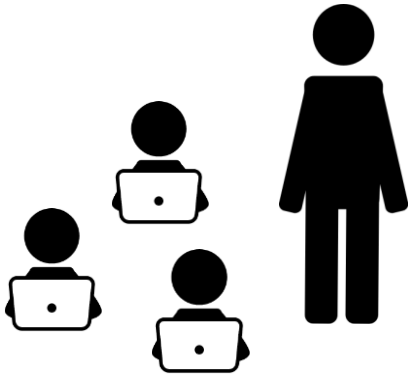
Tips til kodeundervisning

Som i alle fag kan undervisning gjøres på mange måter



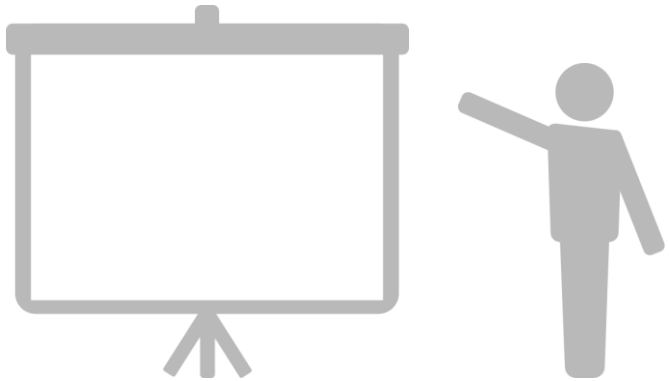


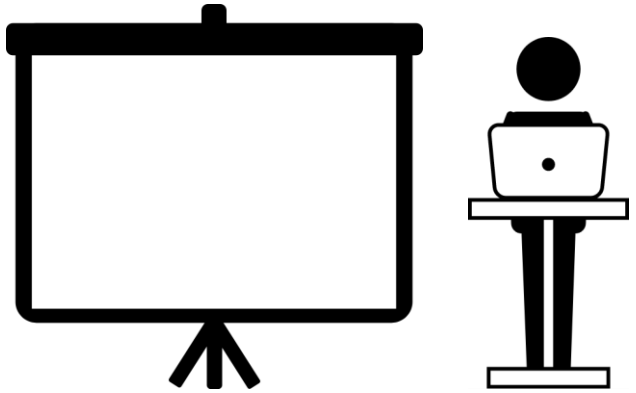


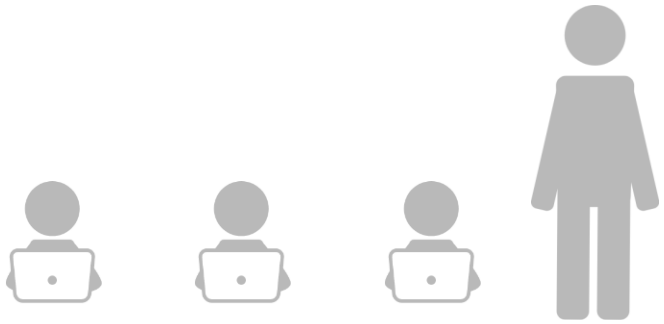


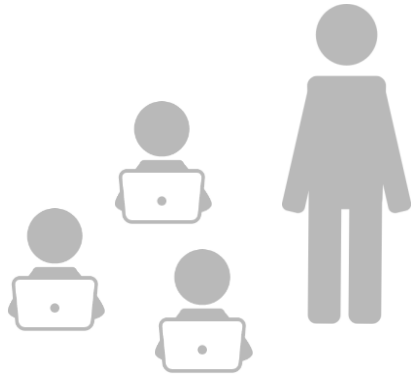
[Brown & Wilson (2018)]

For dette foredraget fokuserer vi på livekoding









[Brown & Wilson (2018)]

slido

Hva var forskjellig med demo nummer to?

ⓘ Start presenting to display the poll results on this slide.

Tips 1: ta med elevene på hele "kodereisen", ikke bare endepunktet

[Brown & Wilson (2018)]

Tips 1: ta med elevene på hele “kodereisen”, ikke bare endepunktet

[Brown & Wilson (2018)]

Tips 1: ta med elevene på hele "kodereisen", ikke bare endepunktet

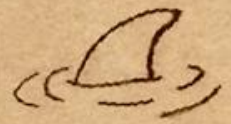
[Brown & Wilson (2018)]

Tips 1: ta med elevene på hele "kodereisen", ikke bare endepunktet

"Koden gir feil resultat, men ingen feilmelding"
grotta

Start

Ferdig kode



Komplisert problem
bukta

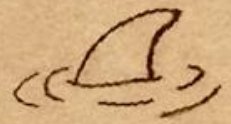
[Brown & Wilson (2018)]

Tips 1: ta med elevene på hele "kodereisen", ikke bare endepunktet

"Koden gir feil resultat, men ingen feilmelding"
grotta

Start

Ferdig kode



Komplisert problem
bukta

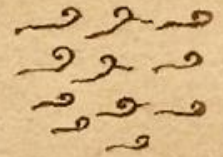
Mystisk sytaksfeil
klippene

[Brown & Wilson (2018)]

Tips 1: ta med elevene på hele "kodereisen", ikke bare endepunktet

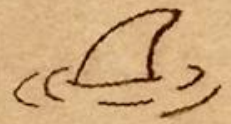
"Koden gir feil resultat, men ingen feilmelding"
grotta

Start



"Glemt importering"
myr

Ferdig kode



Komplisert problem
bukta

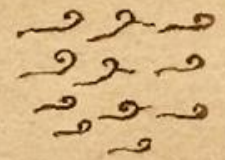
Mystisk sytaksfeil
klippene

[Brown & Wilson (2018)]

Tips 1: ta med elevene på hele "kodereisen", ikke bare endepunktet

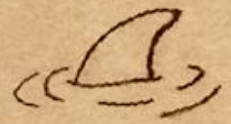
"Koden gir feil resultat, men ingen feilmelding"
grotta

Start



"Glemt importering"
myr

Ferdig kode



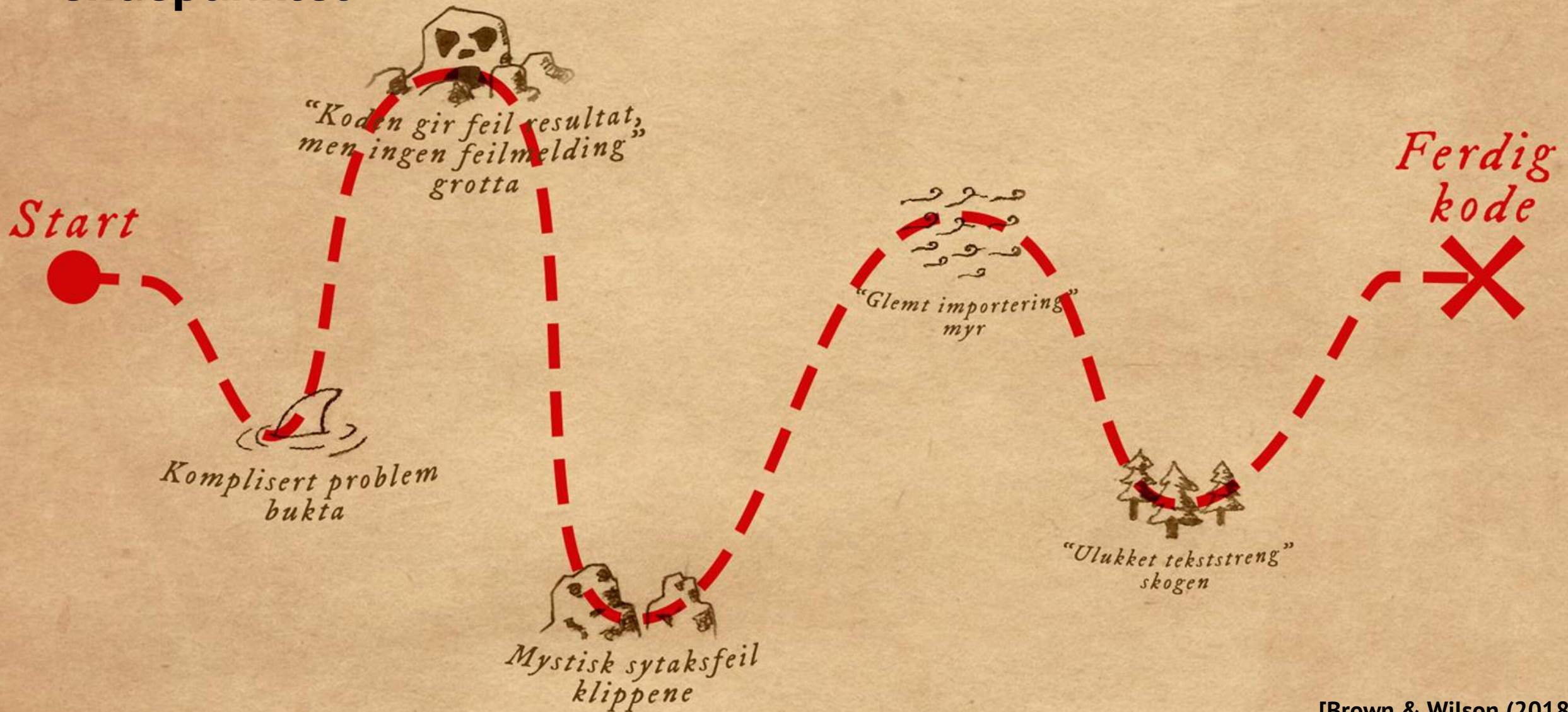
Komplisert problem
bukta

"Ulukket tekststreng"
skogen

Mystisk sytaksfeil
klippene

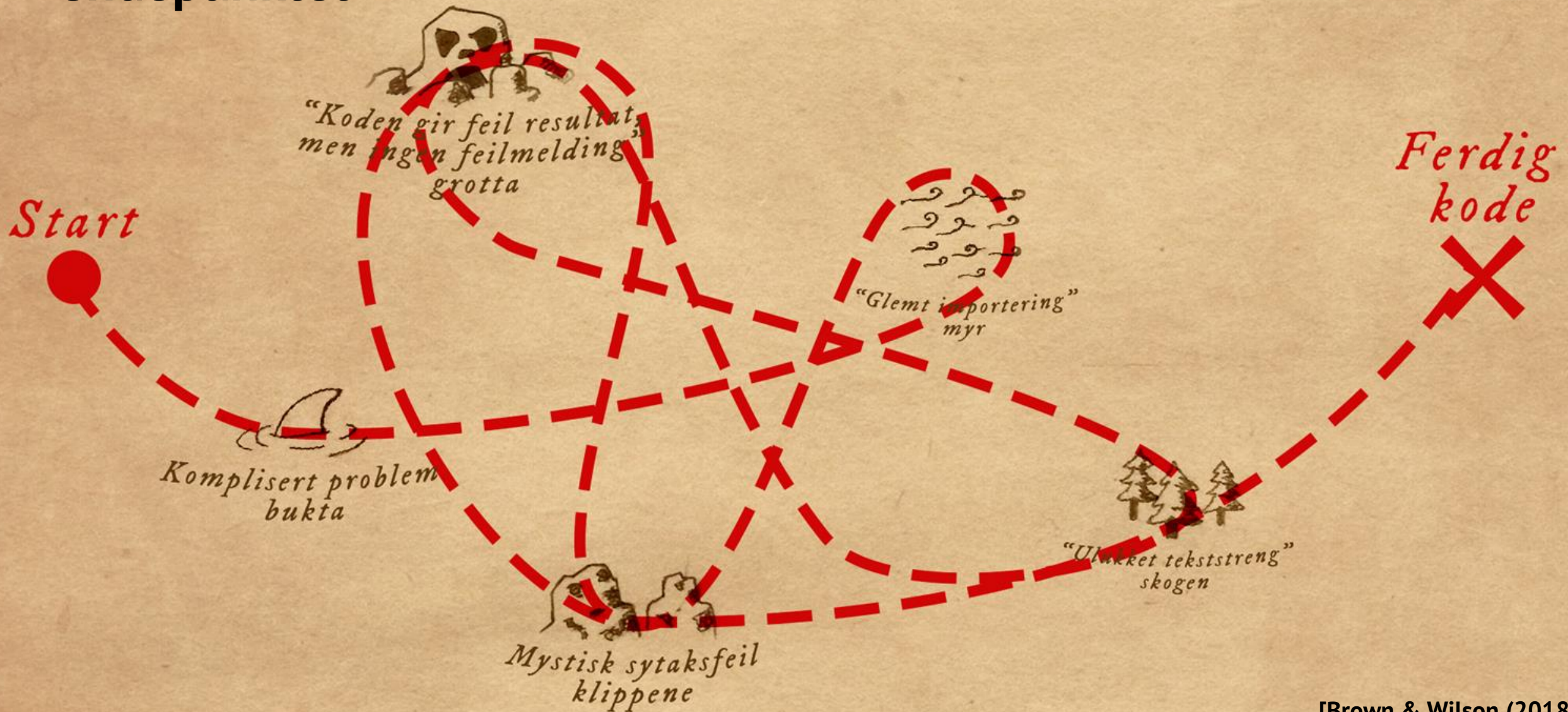
[Brown & Wilson (2018)]

Tips 1: ta med elevene på hele "kodereisen", ikke bare endepunktet



[Brown & Wilson (2018)]

Tips 1: ta med elevene på hele "kodereisen", ikke bare endepunktet



[Brown & Wilson (2018)]

Tips 2: Lag en plan for hva du skal gå igjennom

[Nederbragt et al. (2020)]

Tips 3: Ikke vær redd for å gjøre feil, for det er en viktig del av programmering

[Steele-Johnson & Kalinoski (2012)]

Tips 4: La elevene forutse hva koden din gjør før du kjører den

[Crouch et al. (2004), Miller et al. (2013)]

Tips 5: Skriv god og leselig kode, for kode som er lett å lese er ofte lettere å forstå, både for andre og deg selv

```
def f(x):
    xx = y[0]

    for x in y:
        if x < xx:
            xx = x

    return xx
```

Tips 5: Skriv god og leselig kode, for kode som er lett å lese er ofte lettere å forstå, både for andre og deg selv

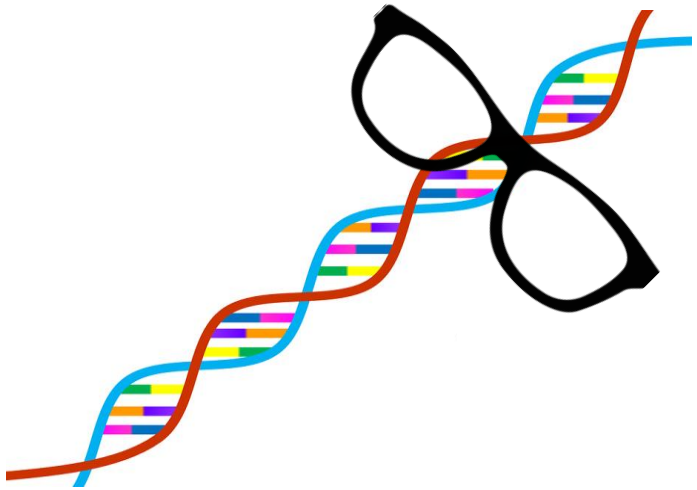
```
def finn_minste_tall(tall_liste):
    foreløpig_minste_tall = tall_liste[0]

    for tall in tall_liste:
        if tall < foreløpig_minste_tall:
            foreløpig_minste_tall = tall

    return foreløpig_minste_tall
```

Tips 6: Du må ikke alltid starte med en blank fil, skjellettkode kan hjelpe elevene å utforske kule problemstillinger tidlig i løpet

```
antall_dyr = 10
antall_år = 20

print(f"Vi starter med {antall_dyr} dyr på øya")
for år in range(1, antall_år):
    antall_dyr = # Din kode her

    print(f"Etter {antall_år} år har øya{antall_dyr} dyr")
```

[Brown & Wilson (2018)]

Bonustips: Dere vil nok oppleve at nivåskillet fra elev til elev er større enn i andre fag

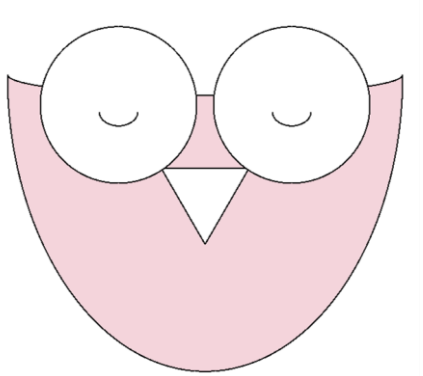


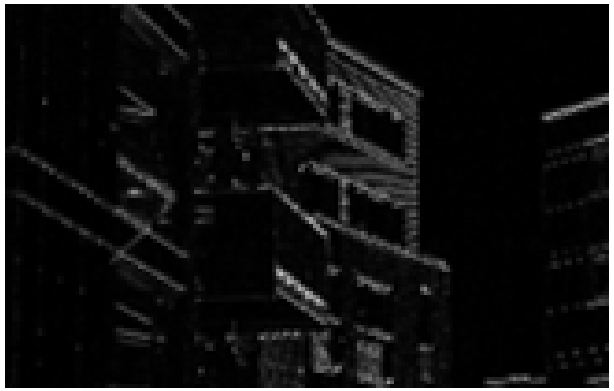
[Patitsas et al. (2019)]

Her er noen nyttige kilder for videre lesing

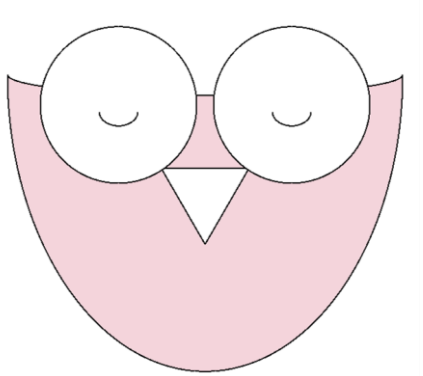
- Crouch, C., Fagen, A.P., Callan, J.P. and Mazur, E., (2004). Classroom demonstrations: Learning tools or entertainment?. American journal of physics, 72(6), pp.835-838.
- Webb, D. C., Repenning, A., & Koh, K. H. (2012, February). Toward an emergent theory of broadening participation in computer science education. In Proceedings of the 43rd ACM technical symposium on Computer Science Education (pp. 173-178).
- Guzdial, M. (2013, August). Exploring hypotheses about media computation. In Proceedings of the ninth annual international ACM conference on International computing education research (pp. 19-26).
- Miller, K., Lasry, N., Chu, K. and Mazur, E., (2013). Role of physics lecture demonstrations in conceptual learning. Physical review special topics-physics education research, 9(2), p.020113.
- Steele-Johnson, D., & Kalinoski, Z. T. (2014). Error framing effects on performance: Cognitive, motivational, and affective pathways. The Journal of psychology, 148(1), 93-111.
- **Brown, N.C. and Wilson, G., (2018). Ten quick tips for teaching programming. PLoS computational biology, 14(4), p.e1006023**
- Patitsas, E., Berlin, J., Craig, M. and Easterbrook, S., (2019). Evidence that computer science grades are not bimodal. Communications of the ACM, 63(1), pp.91-98.
- Nederbragt, A., Harris, R. M., Hill, A. P., & Wilson, G. (2020). Ten quick tips for teaching with participatory live coding. PLoS Computational Biology, 16(9), e1008090.

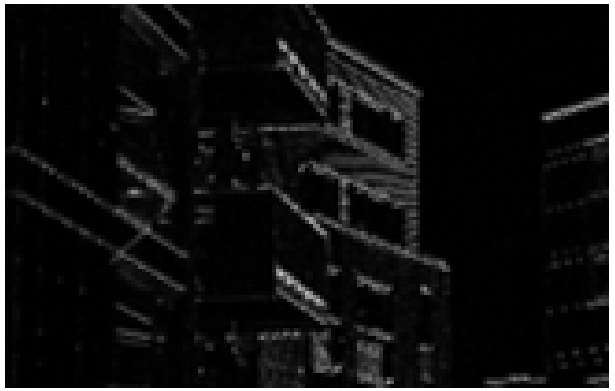
Kort oppsummert er programmering et mangfoldig verktøy som byr på både spennende utfordringer og kreative muligheter





Kort oppsummert er programmering et mangfoldig verktøy som byr på både spennende utfordringer og kreative muligheter





Spørsmål?

Bonustips: En god måte å jobbe to og to er *parprogrammering*