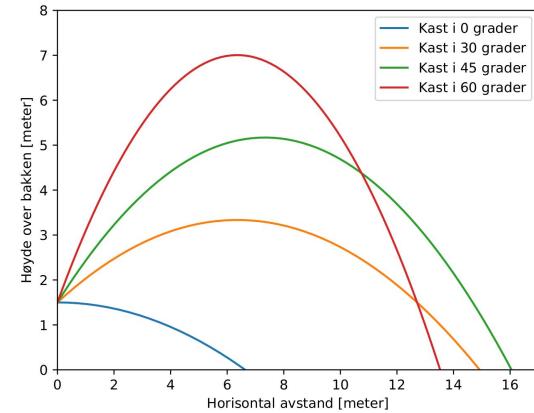


Gruppediskusjon: Undervisningsstrategier for programmering

simula
kodeskolen

```
1 import numpy as np
2 n = 12*50    #antall tidsintervaller
3 y0 = 100     #antall byttedyr når vi starter
4 x0 = 50      #antall rovdyr når vi starter
5 index_set = range(n+1)
6
7 x = np.zeros(len(index_set))
8 y = np.zeros(len(index_set))
9
10
11 a = 0.05   # dødsrate gauper
12 b = 0.0003 # reproduksjonsrate
13
14 c = 0.02   # vekstrate haren
15 d = 0.0001 # dødsrate harer
16
17
18 y[0] = y0
19 x[0] = x0
20 for k in index_set[:-1]:
21     #print y[k]
22     y[k+1] = y[k] + c*y[k] -
23             x[k] - a*x[k] +
```



Hva skal elevene lære?

Ny høring av læreplanene (Udir.no)

Algoritmisk tankegang trekkes inn som et kjernelement 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.”

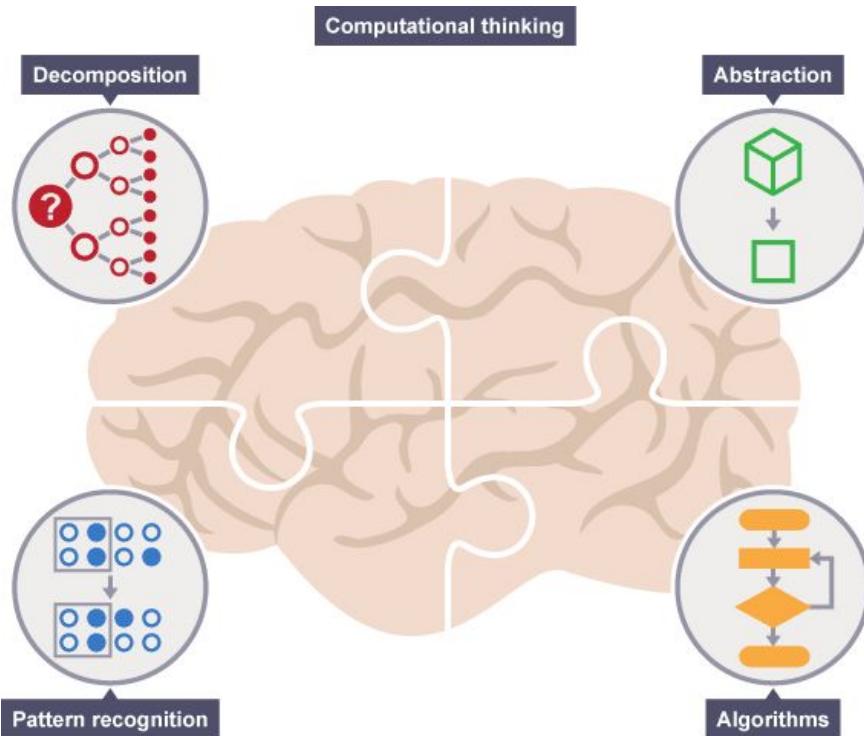
Algoritmisk tankegang trekkes inn som et kjerneelement i matematikkfaget

“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.”

I naturfag blir teknologi et kjernelement, og programmering skal inngå som en del av dette

Derimot er det ikke enighet om nøyaktig hva som inngår i begrepet *algoritmisk tankegang*



Dekomposisjon

Det å bryte ned problemer i mindre deler

Mønstre

Gjenkjenne mønstre og utnytte likheter

Abstraksjon

Identifisere viktig informasjon og fjerne unødvendige detaljer

Algoritme

Formulere steg-for-steg beskrivelser og regler

Evaluering

Vurdere resultat og produkt

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

På videregående nivå er ikke programering nevnt eksplisitt i så mange av kompetanse målene

Matematikk 1T

- formulere og løyse problem ved hjelp av ulike problemløysingstrategiar og digitale verktøy

Matematikk R1

- bruke digitale verktøy i beregninger og utforsking av egenskaper ved funksjoner

Matematikk R2

- bruke **programmering** til å utforske rekursive sammenhenger og presentere egne framgangsmåter
- formulere algoritmer som beregner bestemte integraler numerisk og bruke digitale verktøy til å utføre algoritmene

Matematikk S1

- bruke digitale verktøy til å simulere utfall i stokastiske forsøk

Matematikk S2

- bruke **programmering** til å utforske rekursive sammenhenger og presentere egne framgangsmåter
- bruke digitale verktøy til å simulere utfall i statistiske fordelinger

I naturfag er programmering og teknologi også et kjerneelement, men nevnes meget generelt i kompetanse målene

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

VG1

- bruke og vurdere programmer som løser eller simulerer naturfaglige problemstillinger

Kort fortalt: Lærere får selv en utrolig stor frihet til å velge hvor, hvordan og hvor mye programmering skal brukes

Frihet gir autonomi, men kan også lede til utfordringer og uklarheter rundt hva som forventes og hva som er rimelig fokus/mengde.

Kort fortalt: Lærere får selv en utrolig stor frihet til å velge hvor, hvordan og hvor mye programmering skal brukes

Frihet gir autonomi, men kan også lede til utfordringer og uklarheter rundt hva som forventes og hva som er rimelig fokus/mengde.

Diskusjon:

1. Hva syns du om at programmering skal inn i matematikk/naturfag?
2. Er du selv motivert for å bruke programmering i klasserommet?
3. Hva syns du, per nå, er den største personlige utfordringen?

Hvor er det
hensiktsmessig å bruke
programmering?

For at programmering i skolen skal være vellykket:

- Må være relevant
- Må være forståelig
- Må gi mestringsfølelse

Diskusjon:

- Er du enig i disse “kravene”? Ville du lagt til noen fler?
- Hvordan kan vi gå frem for å sikre at disse ivaretas?

Programmeringens styrke i realfagene ligger i at det er et verktøy som lar oss løse kjente problemer på nye måter

Å se kjente problemstillinger fra en ny vinkel kan øke forståelse og være god mengdetrenings

Programmering kan også la oss gå lenger i visse problemstillinger, og *utvide* forståelse



Hvordan undervise eller
inkludere programmering?

Det finnes overraskende lite forskning innen informatikkdidaktikk



PLOS | COMPUTATIONAL
BIOLOGY

EDUCATION

Ten quick tips for teaching programming

Neil C. C. Brown¹*, Greg Wilson²*

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

These authors contributed equally to this work.
* gwilson@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.

Tips 1: Bruk live-programmering

I denne metoden programmerer læreren live foran elevene på storskjerm. Elevene deltar aktivt ved å stille spørsmål og kan eventuelt kode med på egne maskiner samtidig

Fordeler

Ulemper

Tips 1: Bruk live-programmering

I denne metoden programmerer læreren live foran elevene på storskjerm. Elevene deltar aktivt ved å stille spørsmål og kan eventuelt kode med på egne maskiner samtidig

Fordeler

- Man kan justere utifra elevers forståelse og spørsmål
- Det går automatisk mye saktere
- Elever lærerer *programmering* og problemløsning, ikke bare koding

Ulemper

- Overraskende vanskelig for læreren!
- Kan ikke veilede individuelle elever hvor det ikke fungerer
- Kan føles litt som “Monkey see, Monkey do”.

Tips til å hjelpe i liveprogrammeringen:

- **Problem: Overraskende vanskelig for læreren!**
 - **Løsning: Forberedelse. Ha gode notater.**
- **Problem: Kan ikke veilede individuelle elever hvor det ikke fungerer**
 - **Løsning: Løsning: La elever veilede hverandre, ta pauser**
- **Problem: Kan føles litt som “Monkey see, Monkey do”.**
 - **Løsning: Fokuser på forståelse og tanke, la elever stille spørsmål, still spørsmål til elevene**

Tips 2: Orakelbasert oppgaveløsning er den beste måten å utforske problemstillinger, men krever tydelig definerte oppgavesett

I denne metoden jobber elever med utdelte oppgaver. Læreren (orakelet) går rundt å hjelpe der det er behov

Fordeler

Ulemper

Tips 2: Orakelbasert oppgaveløsning er den beste måten å utforske problemstillinger, men krever tydelig definerte oppgavesett

I denne metoden jobber elever med utdelte oppgaver. Læreren (orakelet) går rundt å hjelpe der det er behov

Fordeler

- Programmering læres best ved å gjøre
- Elever får jobbe i eget tempo
- Mengdetrenings

Ulemper

- Kan være vanskelig å rekke rundt til alle
- Kan være frustrende å sette seg fast
- Kan bli for mye fokus på kode, ikke problemløsning

For at orakelbasert undervisning skal bli optimalt bør oppgavesett være tilpasset, og elever bør *jobbe i par*

Et godt oppgavesett

- Mange oppgaver, starter lett og stiger i vanskighetsgrad
- Har tydelig definerte deloppgaver
- Det bør være tydelig om man har klart en oppgave eller ei
- Avslutter gjerne med mer utforskende/åpne oppgaver

Programmering er en aktivitet man helst bør gjøre sammen

Når man jobber sammen tvinges man til å diskutere, og “snakke kode”. Man bli mindre opptatt av detaljer, og fokuserer mer på problemløsningen

Også mindre sannsynlighet for å sette seg ordentlig fast, og unngår frustrasjon

En liten ulempe med gruppearbeid er at de sterkeste kan ta kontroll, og de svakere melder seg ut

En god balanse er *parprogrammering*



Image by [Lisamarie Babik](#) [CC BY 2.0]

I parprogrammering jobber to personer på én maskin

To klare roller: **Sjåfør og navigatør**

Sjåføren: Styrer maskinen, og skriver den faktiske koden.

Navigatøren: Følger med på sjåføren og stiller spørsmål, gir råd, finner feil, foreslår forbedringer



Sjåføren tenker på de små, tekniske detaljene.

Navigatøren har et mer overordnet blikk og er et sikkerhetsnett

Det er viktig å bytte roller ofte!

Minst hvert 30. minutt, men gjerne oftere



Image by [Anders Jensen](#)

Tips 3: Prosjektbasert arbeid kan være en god måte å skape et større opplegg eller utforskende arbeid

I denne metoden jobber elever med utdelt prosjektarbeid/bestilling over lengre tid.
Læreren veileder etter behov

Fordeler

- Kan være meget engasjerende
- Kombinerer lett programmering med eksperiment/teori
- Kan gi meget god mestringsfølelse

Ulemper

- Elevene må kunne nok til å jobbe nogenlunde selvstendig
- Sterke elever kan gjør alt arbeidet mens svake melder seg ut

I prosjektarbeid må ikke alt alltid lages fra bunn av, man kan også gi eksempelkode eller skjelettkode

Eksempelkode:

- Kode som fungerer og kjører. Elever må utvide eller tilpasse koden til sitt behov

Skjelettkode:

- Ukomplett kode. Elevene må fylle inn “skjellettet” for å kunne bruke koden

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

(Dette er spesielt sant om dere har programmering som programfag)

Pass på:

- Ikke kjøp myten om at noen bare ikke kan lære å programmere (“there is no geek gene”)**
- Husk at selv om noen er utrolig flinke vil mange være totalt nybegynnere**

Ikke minst: Bruk sterke elever som en god ressurs i klasserommet

Som ekstra motivasjon og utfordring til de mest interesserte og/eller flinkeste finnes det en trend av spillifisering av koding



Norsk Informatikkolympiade





A dynamic comic book illustration set against a dark, apocalyptic background. In the upper left, a man with red hair and a determined expression looks down at another character who is partially obscured by shadows. The lower right features a man with a beard and a blue hoodie, sitting amidst rubble and holding a laptop displaying a question mark. A massive, jagged hole dominates the center, with shards of light blue energy or metal flying outwards. The overall style is gritty and action-oriented, suggesting a theme of technological conflict or destruction.

CLASH OF CODE