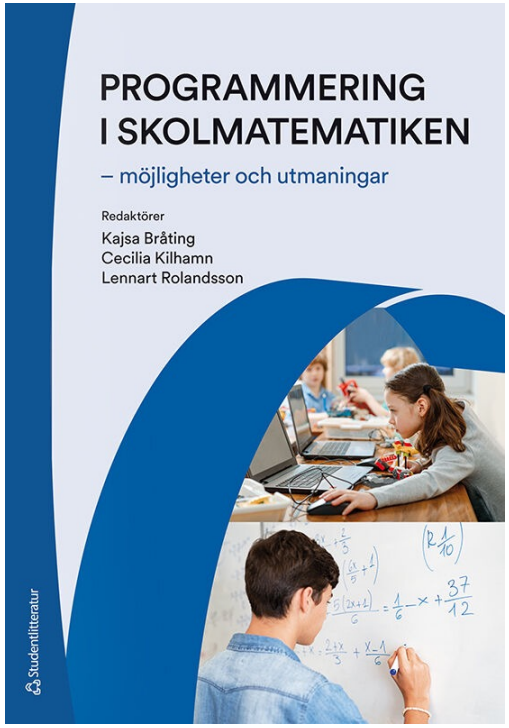
Programmering i grundskolan (i matematik)



Möjligheternas dag, BTH
1 november 2022

Cecilia Kilhamn
Göteborgs universitet
NCM
cecilia.kilhamn@ncm.gu.se

Programming i kursplanen: matematik

Lgr22:

Revidering 2017 för att öka tyngden på **digital kompetens**.

Programmering som en del av kursplanerna i **matematik** och **teknik**.

I matematik står det i syftet:

Vidare ska eleverna genom undervisningen
ges möjligheter att utveckla kunskaper i att
använda digitala verktyg och programmering för att kunna
undersöka problemställningar och matematiska begrepp,
göra beräkningar, samt för att
presentera och tolka data.

Programming i kursplanen: matematik

Centralt innehåll Algebra

- Åk 1-3

Entydiga stegvisa instruktioner och hur de konstru skrivs och följs som grund för programmering.
Hur symboler används vid stegvisa i

(analogt eller med enkl

- Åk 4-6:

Hur algoritmer skapa s vid programmering.
Programmering i visu programmeringsmiljöer

(blockprogramming t.ex. Scratch)

- Åk 7-9:

Hur algoritmer skapas, testas och förbättras vid programmering.
Programmering i visuell och textbaserad programmeringsmiljö.

(Scratch + textprogrammering t.ex.Python eller Jacascript)

VAD ÄR EN ALGORITM?

Programming i kursplanen: teknik

- Åk 1-3

Styrning av föremål med programmering.

- Åk 4-6:

Styrning av egna konstr
programmering

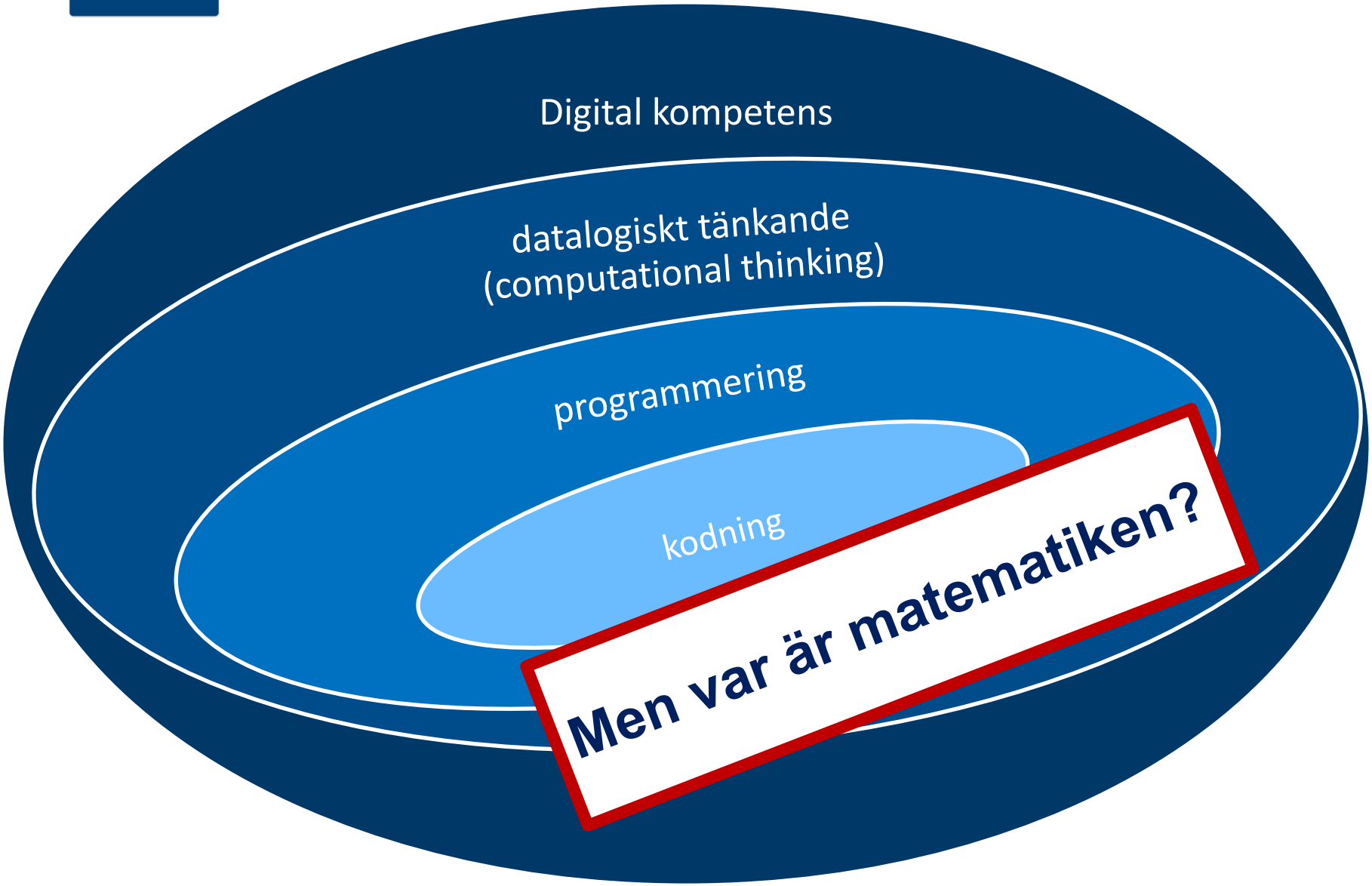
- Åk 7-9:

Hur digitala verkty as i teknikutvecklingsarbete, till exempel för att gör ingar och simuleringar.

Egna konstruktioner där man använder styrning eller reglering med hjälp av programmering.

Varför skilja ut algoritmer från deras tillämpning?

GÖTEBORGS UNIVERSITET



Erfarenheter från InPAC-studien

Analog programmering

- Styra rörelser vanligast (pilar=steg)

Visuella programmeringsmiljöer

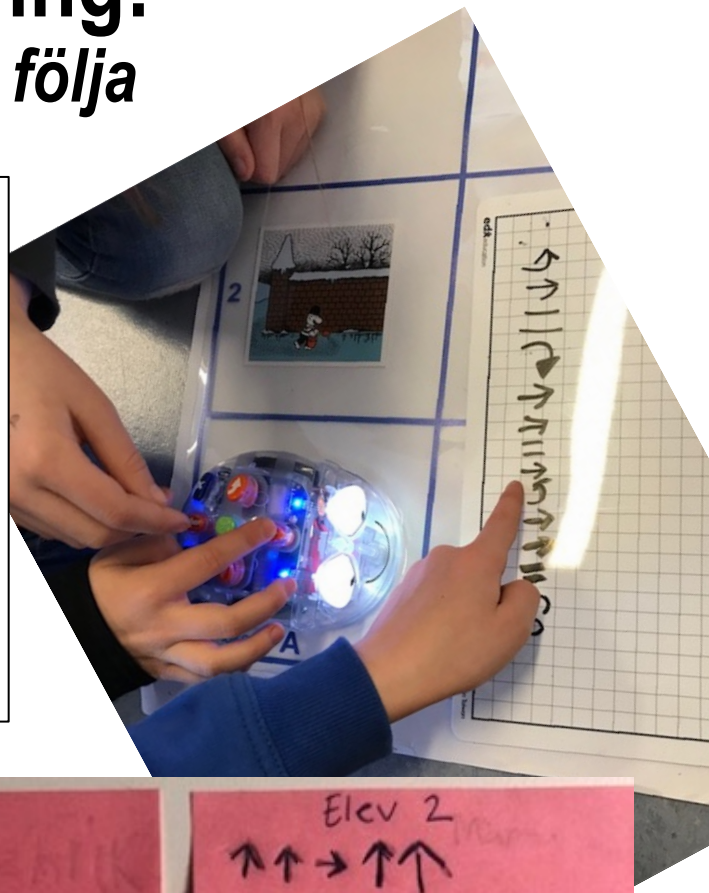
- Kopplar programmering mer till andra ämnen (sv, mu, te)
  animeringar, musik, berättelser med Scratch,
  minirobotar typ Beebots och Microbit
- ”roligt” men innehåller många distraktorer som tar elevernas fokus från matematiken.

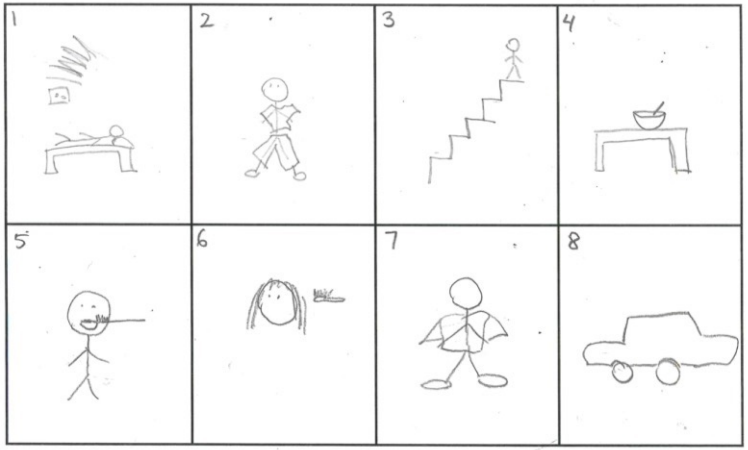
Textbaserad programmeringsmiljöer

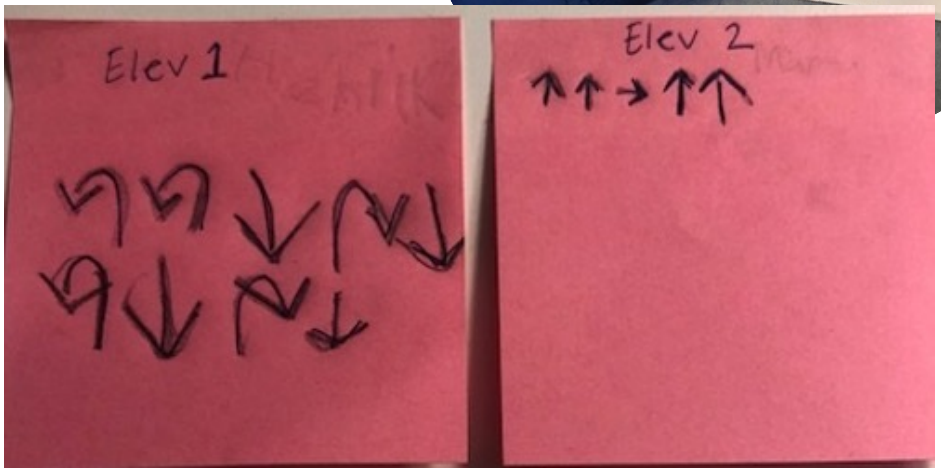
- Mer fokuserade, hela klassen samlade, lättare att hantera
  Python, lite Javascript (även kalkylblad, GeoGebra)
- Mer problem med syntax.
- Mer focus på matematiken.

Analog programmering:
konstruera, beskriva och följa

- Tänk ut en serie stegvisa instruktioner
- Symbolisera dem med bilder
    1. föreställande
    2. ikoniska
    3. symboliska
- Testkör och se om målet nås



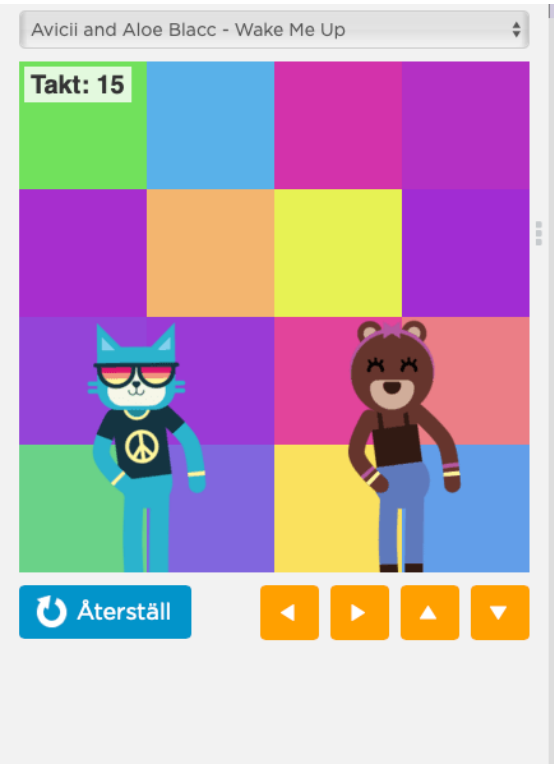


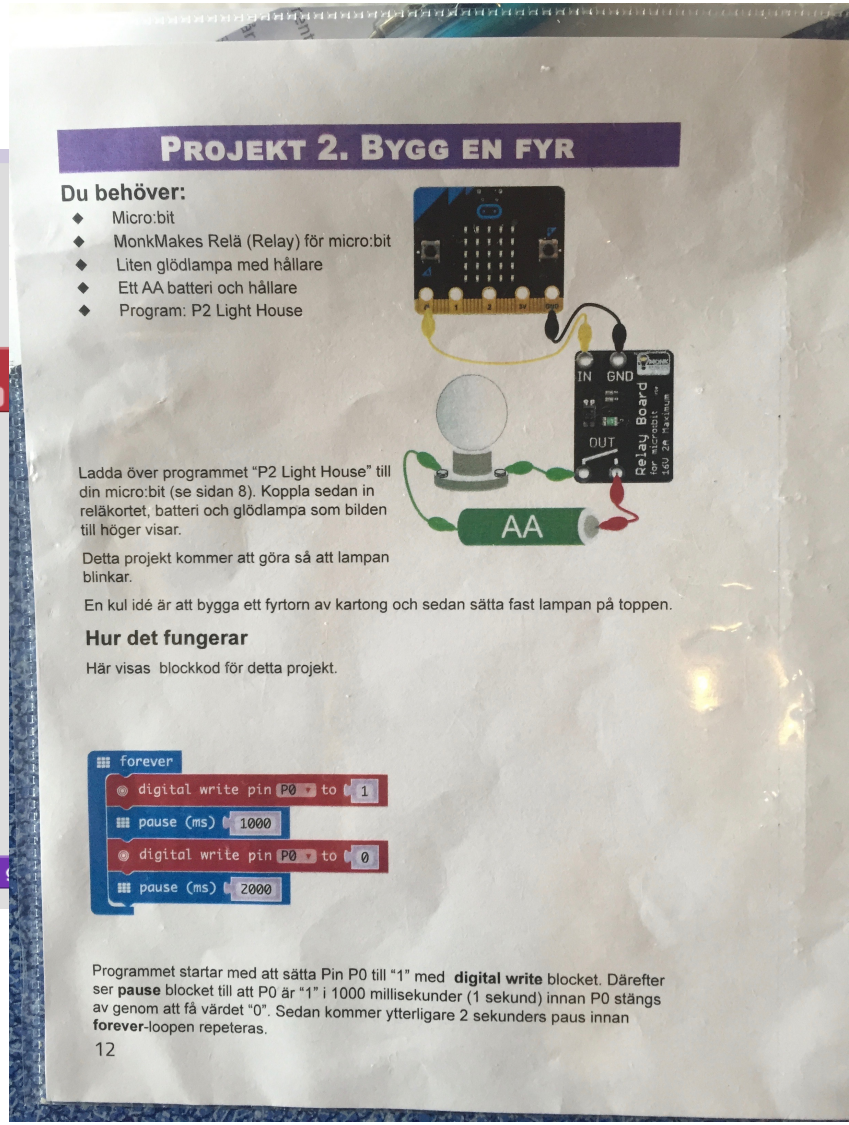


GÖTEBORGS UNIVERSITET

Visuell programmering (block):

konstruera, beskriva och följa





PROJEKT 2. BYGG EN FYR

Du behöver:

- Micro:bit
- MonkMakes Relä (Relay) för micro:bit
- Liten glödlampa med hållare
- Ett AA batteri och hållare
- Program: P2 Light House

Ladda över programmet "P2 Light House" till din micro:bit (se sidan 8). Koppla sedan in reläkortet, batteri och glödlampa som bilden till höger visar.

Detta projekt kommer att göra så att lampan blinkar.

En kul idé är att bygga ett fyrtorn av kartong och sedan sätta fast lampan på toppen.

Hur det fungerar

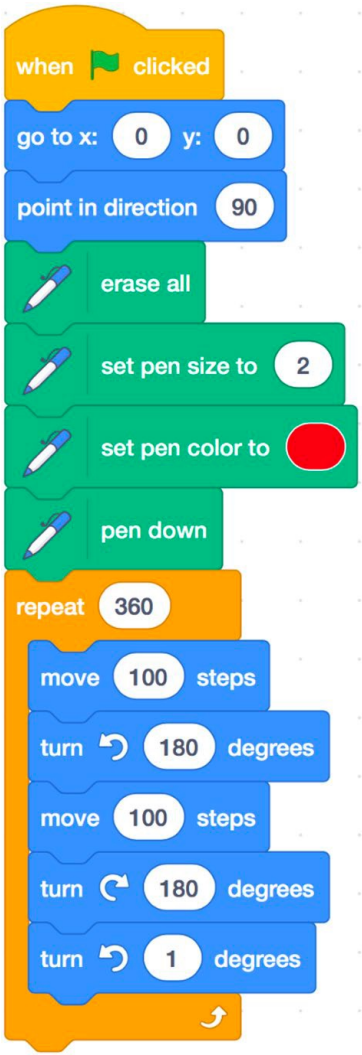
Här visas blockkod för detta projekt.

Programmet startar med att sätta Pin P0 till "1" med **digital write** blocket. Därefter ser **pause** blocket till att P0 är "1" i 1000 millisekunder (1 sekund) innan P0 stängs av genom att få värdet "0". Sedan kommer ytterligare 2 sekunders paus innan **forever**-loopen repeteras.

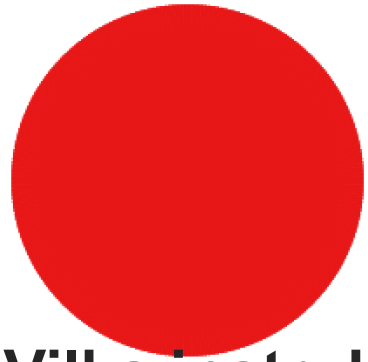
12

GÖTEBORGS UNIVERSITET

Visuell programmering (block):
skapa (och använd), testa och förbättra



https://scratch.mit.edu/projects/625071853/fullscreen/



Vilka instruktioner ska vi ge?

tänk – skissa – gör pseudokod – koda

Vad gör de olika delarna av koden?

Förutse: vad händer om…?

Testa: ändra och se

Förbättra: gör något mer avancerat

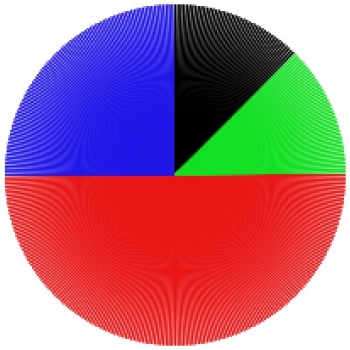
GÖTEBORGS UNIVERSITET

Tinkering – lek med koden

utforska, ändra, förbättra

- Lägg till koordinatsystem.
  Ändra rotation, lägg till variabler

https://scratch.mit.edu/projects/478462569/fullscreen/

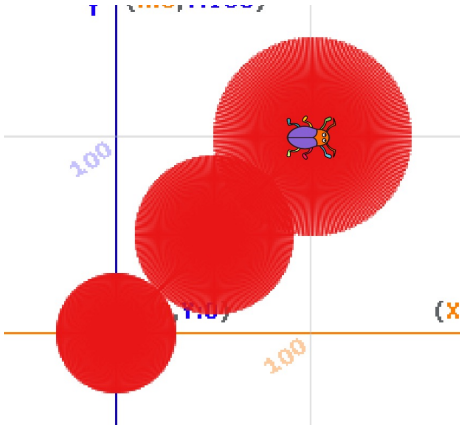


- Ändra gradtal, del av cirkel

https://scratch.mit.edu/projects/478427377/fullscreen/

- Komprimera kod

https://scratch.mit.edu/projects/478587555/fullscreen/

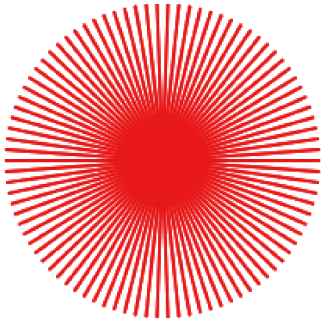


- Ändra placering i x-led

https://scratch.mit.edu/projects/625066884/fullscreen/

- Ändra radie, x-led, y-led

https://scratch.mit.edu/projects/478543202/fullscreen/



- Tillsätt variabler

https://scratch.mit.edu/projects/752671060/fullscreen/

GÖTEBORGS UNIVERSITET

Klassisk Mönsteruppgift.

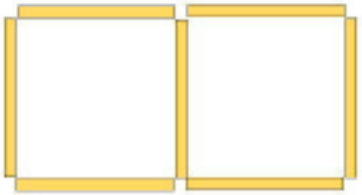
Progression från åk 1 till 9?

Exempel 3: tändsticksmönster

Material: stickor av samma längd.

Figur 1

Figur 2



Figur 3

Nivå 1: Lägg upp mönstret som bilden visar. Fortsätt mönstret. Hur många fler stickor behövs för nästa figur? Och för nästa? Hur vet man det?

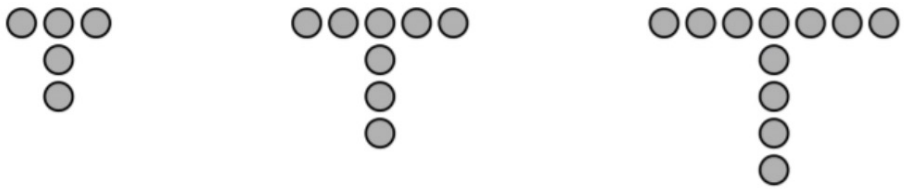
Nivå 2: Hur många stickor behöver man till figur nummer 6? Till figur nummer 10? Till figur nummer 20? Till figur nummer 100? Beskriv hur du ska kunna räkna ut det.

Nivå 3: Formulera ett generellt uttryck som beskriver antalet stickor som behövs för att bilda en figur. Beteckna figurens nummer med n.

GÖTEBORGS UNIVERSITET

Traditionell mönsteruppgift

☹ tråkigt!



Figur 1 Figur 2 Figur 3

Figure nr	Markers
1	5
2	8
3	11
5	17
7	23
10	32
20	62
100	302
n	3*n*+2

Tinkering

Det generella uttrycket som utgångspunkt

☺ Wow – kolla här!!

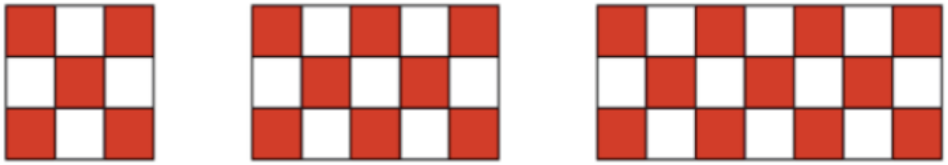
```
1 figure_number = int(input("What figure do you want the number of markers for? "))
2 number_markers = figure_number*3 + 2
3 print("Figure number ", figure_number , "has ", number_markers, "markers")
```

Alternativ mönsteruppgift med programmering (Anders Karlsson)

Gör ett program som kan räkna ut antalet vita kvadrater ur följande mönster. Programmet skall fråga efter vilken figur du vill räkna ut vita kvadrater på och sedan skall programmet visa resultatet.

Du får gärna modifiera programmet så att det blir riktigt bra om du vill!

Bra om du redan nu börjar kommentera / förklara din kod. Sätt ut # och skriv förklaring!



Det känns som om eleverna lättare har förstått vikten av att ta fram formeln för ett mönster för att man använder det i programmet.

Det har också gått lättare att förklara vad figur n är för något.
Nu har vi något att hänga upp det på för att det är där i programmet som det finns.

Visst finns det elever som inte förstått detta nu men det har det också varit tidigare. De kan lägga mönster med klossar och säga hur mönstret upprepar sig men inte på ett matematiskt allmänt sätt.

GÖTEBORGS UNIVERSITET

Algebraiskt eller algoritmiskt tänkande?

Olika sätt att symbolisera en struktur med generella egenskaper och obestämda tal.

v är antalet vita rutor

n är figurens nummer

$v = 3n + 1$

```
while 1:
    figur = input("vilken figur vill du räkna ut antalet vita kvadrater på?")
    try:
        figur_nummer = int(figur)
    except:
        print("du måste skriva in en siffra")
        continue
    antalet_vita_rutor = figur_nummer*3 + 1
    print(antalet_vita_rutor)
```

GÖTEBORGS UNIVERSITET

Textprogrammering: syntaxproblem

Skillnad mellan matematik och programmering

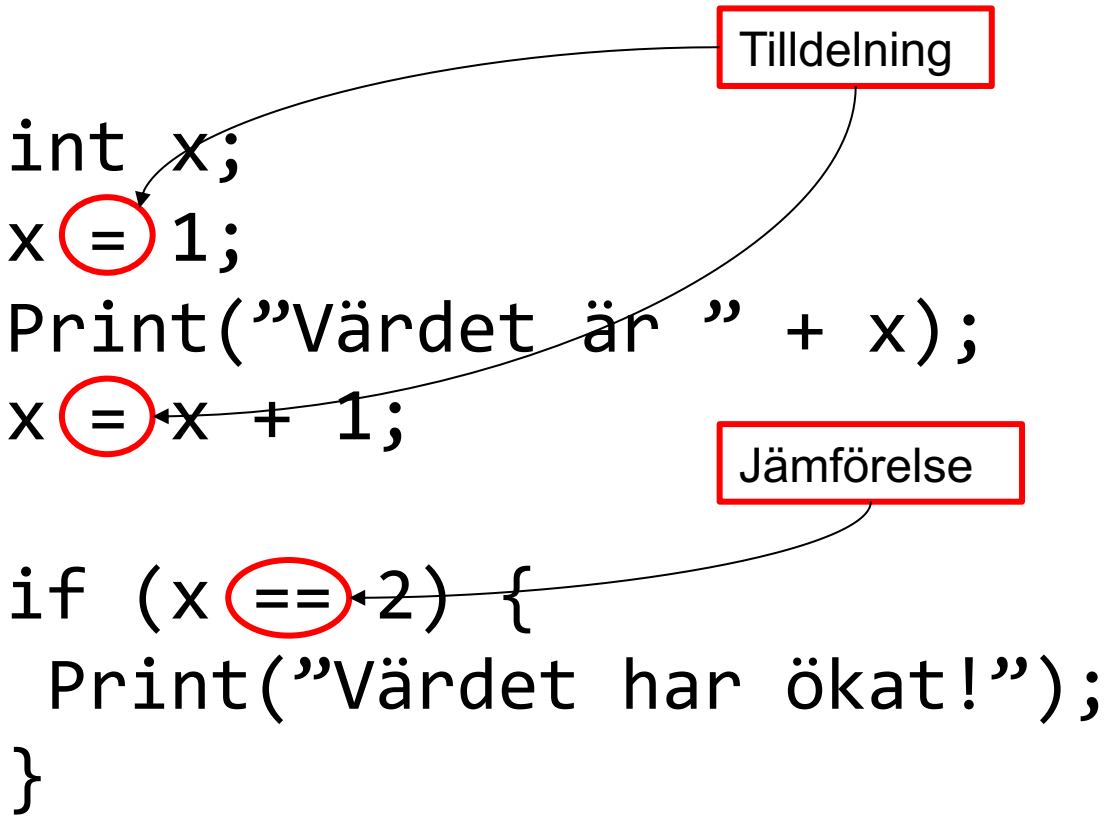
$$x = x + 1$$

Algebra: - Falskt! Likheten satisfieras inte för något x.

Programmering: - Körbart, inga problem!

GÖTEBORGS UNIVERSITET

Likhetstecknets betydelse i programmering



Algebra -------- Programming

= likhet, balance

$x \neq x + 1$

x har samma värde

Variabel:
- Okänt tal
- Generellt tal
- Variabel

Alltid numerisk
i skolalgebra

Uttryck kan manipuleras
utan att variabeln är känd.

= tilldelning

$x = x + 1$

nya gamla värdet

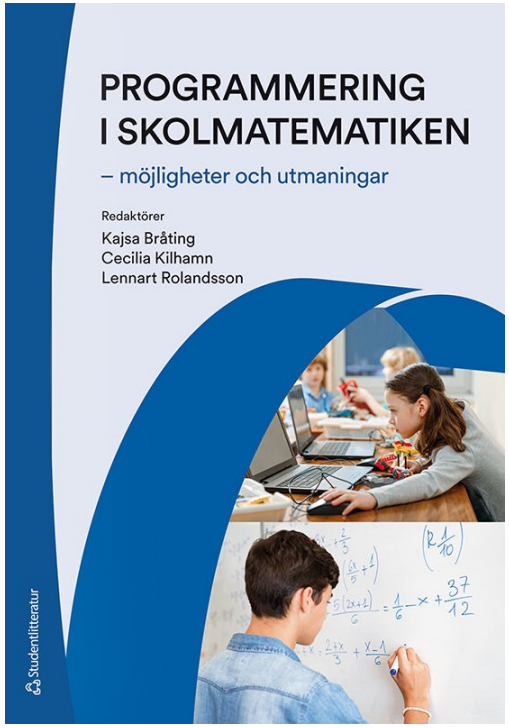
Variabel:
- Minne, Förvaringsplats för data

"Variabler kan ses som lådor
där vi kan stoppa olika världen.
Alla lådor har ett namn, variabelns namn"

Numerisk eller icke-numerisk,
varierande och icke-varierande,
Slumpmässig

Kan bara köras när variabeln fått ett värde

Tack för att ni lyssnat!



Cecilia Kilhamn

Göteborgs universitet

NCM

cecilia.kilhamn@ncm.gu.se