

Crash Course i Programmering

HumTek, RUC

Kursus mål

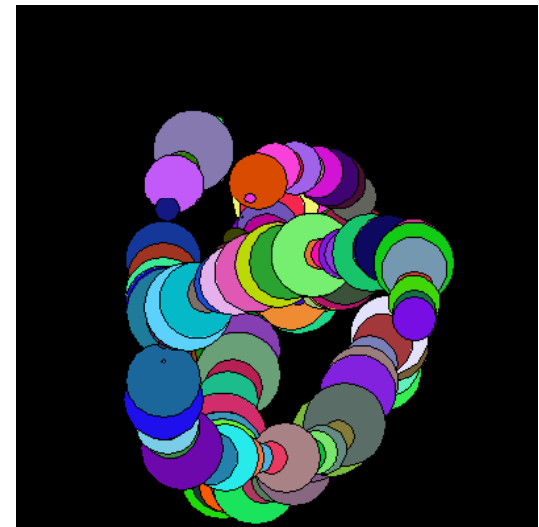
- At give en basal introduktion til programmering i sproget *Processing*
- At give et overblik over sprogets potentiale
- At have det sjovt :-)

Kursus form

- Meget kompakt/intensivt:
- Kun 3 forelæsninger
- Enkelte øvelser indlagt undervejs
- Forberedelse hjemmefra
- Udbytte kræver en indsats(!)

At komme i gang ...

- Download og installer Processing
 - Download processing fra:
 - <http://processing.org/download/index.html>
eller kopier på anden måde
 - unzip til foretrukken folder
fx C:\Program Files
 - opret evt et shortcut i startmenuen
eller andetsteds til processing.exe
 - start Processing (brug shortcut eller
dobbeltklik på processing.exe)



... At komme i gang

- kopier følgende program og sæt det ind i et Processing vindue

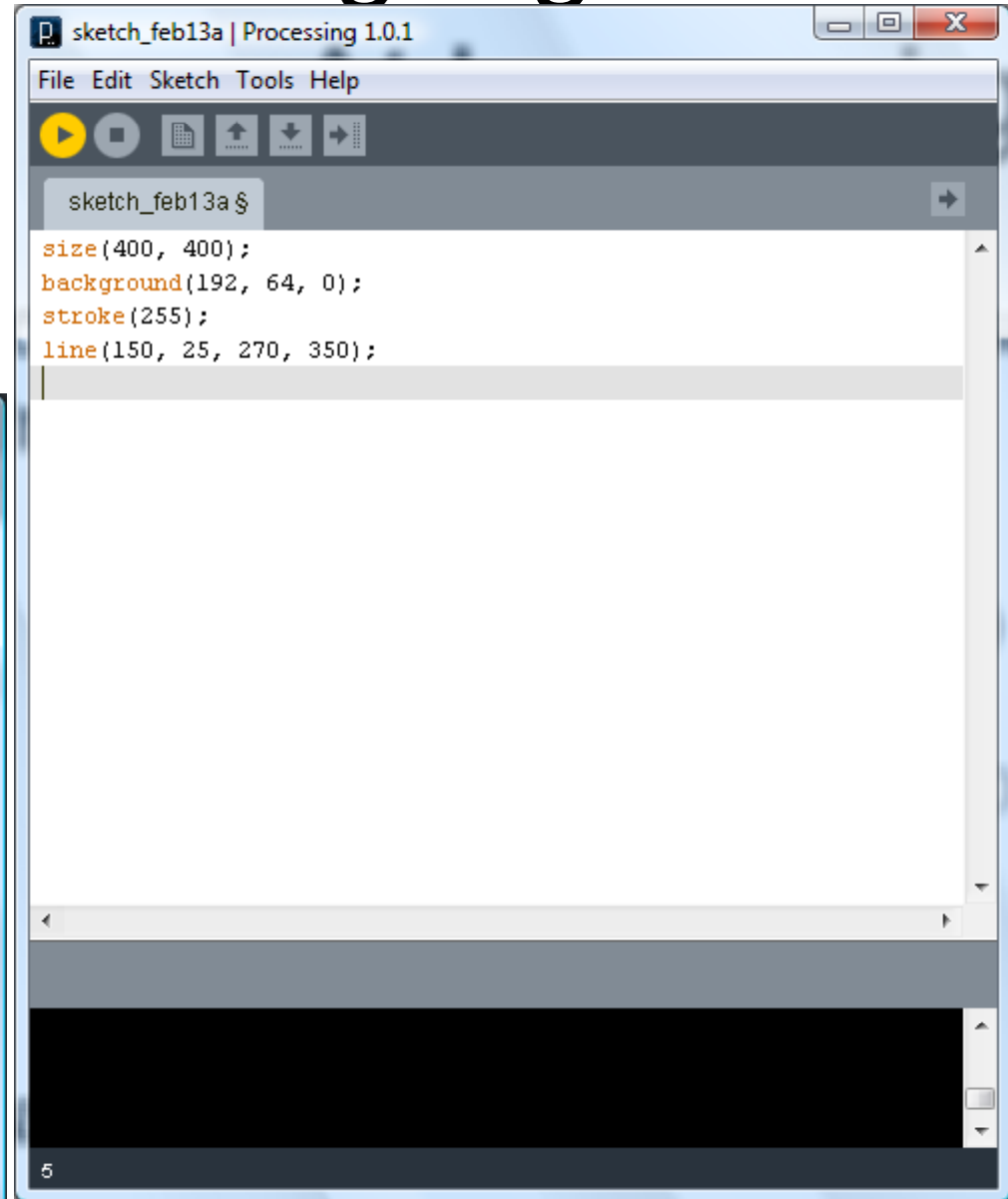
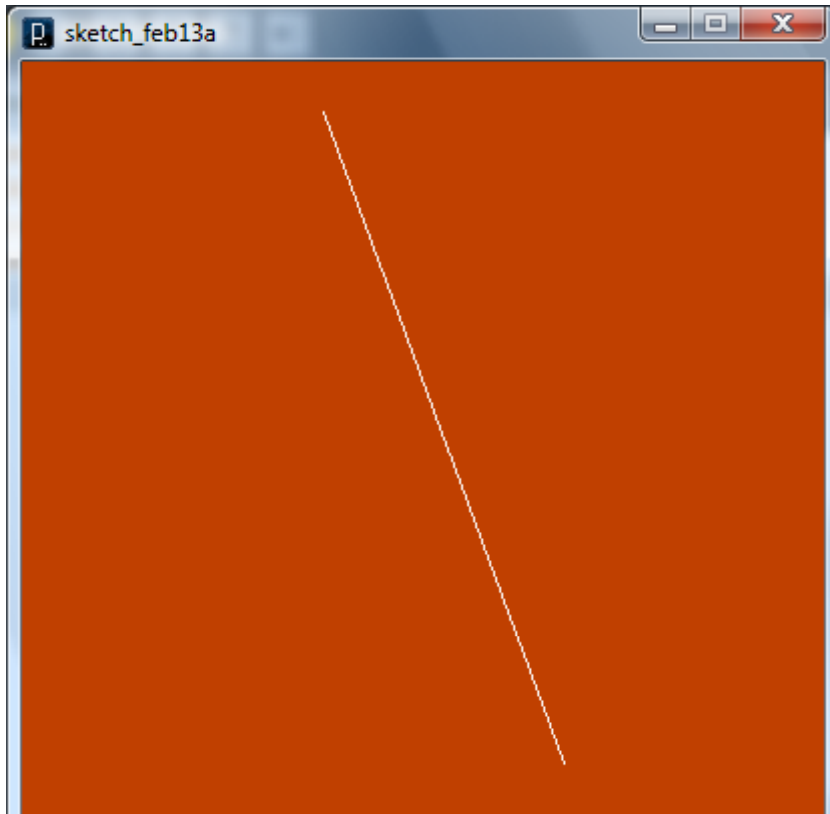
```
size(400, 400);  
background(192, 64, 0);  
stroke(255);  
line(150, 25, 270, 350);
```

- kør programmet



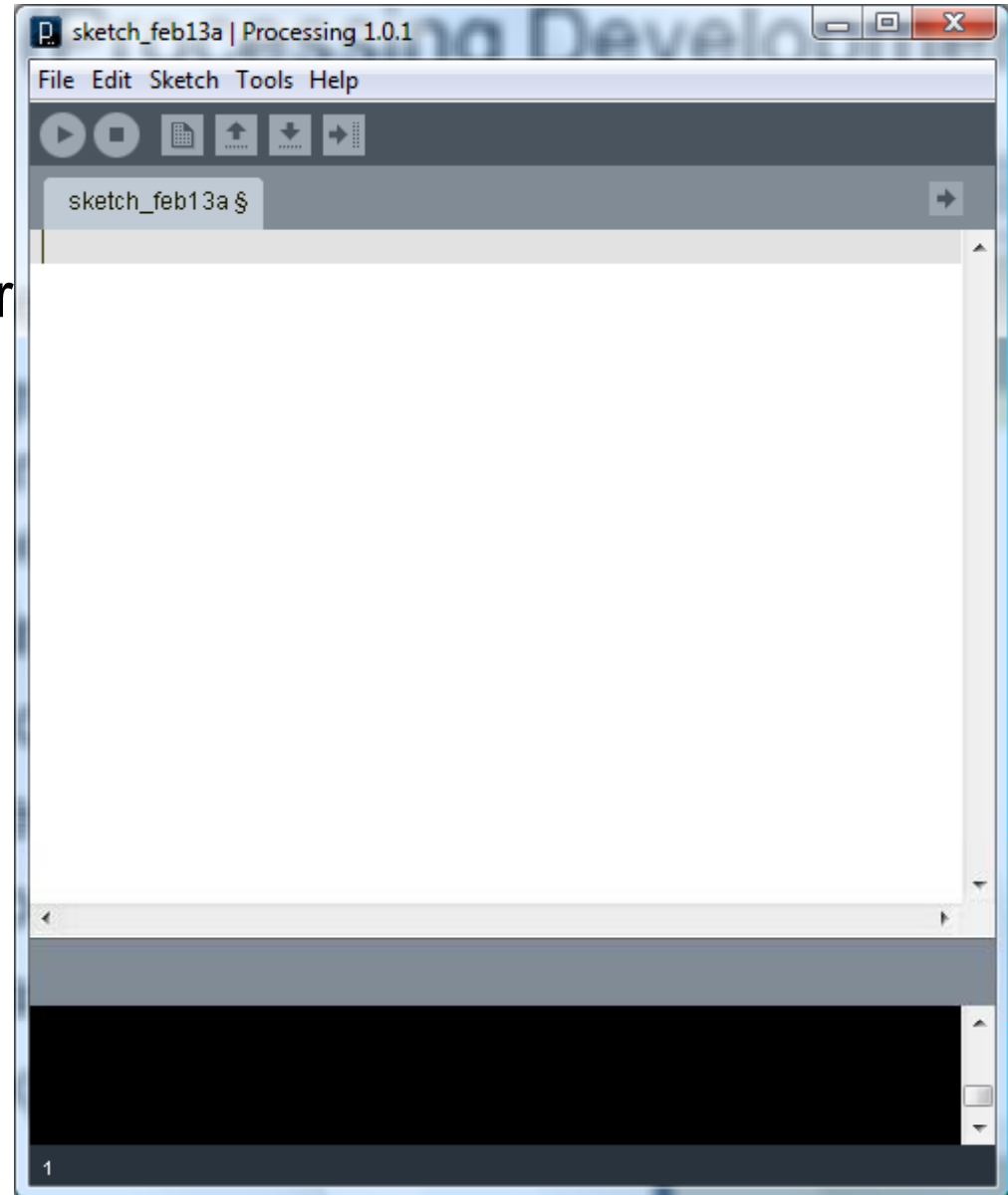
... At komme i gang

- `size(400, 400);`
- `background(192, 64, 0);`
- `stroke(255);`
- `line(150, 25, 270, 350);`



PDE (Processing Development Environment)

- simpel editor
- vigtigste funktionalitet tilgængelig som knapper og menu:
 - run
 - stop
 - new
 - open
 - save
 - export



Processing

- Processing
 - er først og fremmest et programmeringssprog
 - appellerer også som visuelt udtryksmiddel og er
 - særdeles brugbart til data visualisering.

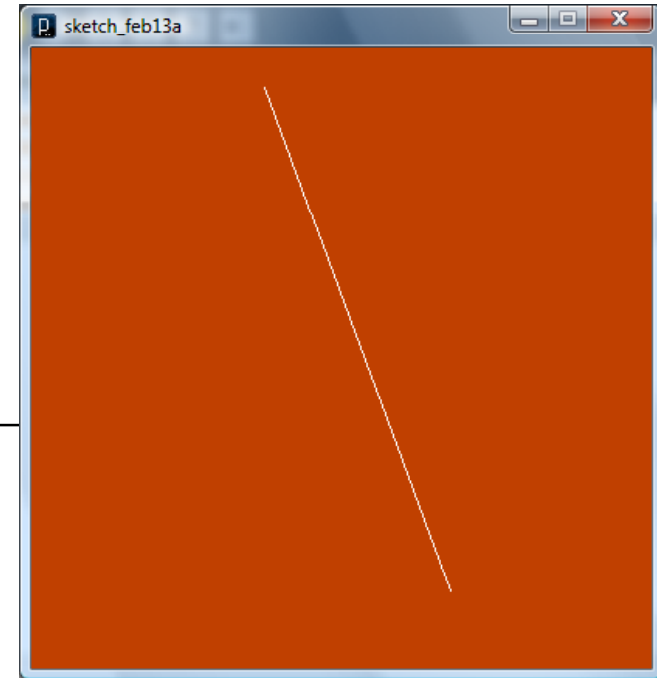
- Processing
 - et selvstændigt sprog, men også
 - en slags skal uden på Java (samme syntaks)
 - giver en intuitiv vej til visuel programmering
 - derfor også velegnet til introduktion til programmering

program = "sketch"

- Programmer producerer tegninger / skitser
- To vigtigste måder et program kan køre på:
 - Basic Mode
 - producerer et enkelt billede
 - opbygges som en liste af sætninger
 - når billedet vises slutter programmet
 - Continuous Mode
 - giver mulighed for animation og interaktion
 - opbygges med standard-funktioner
 - setup() - kaldes een gang (initialisering)
 - draw() - kaldes løbende
 - programmet slutter ikke, men kører indtil det bliver afbrudt

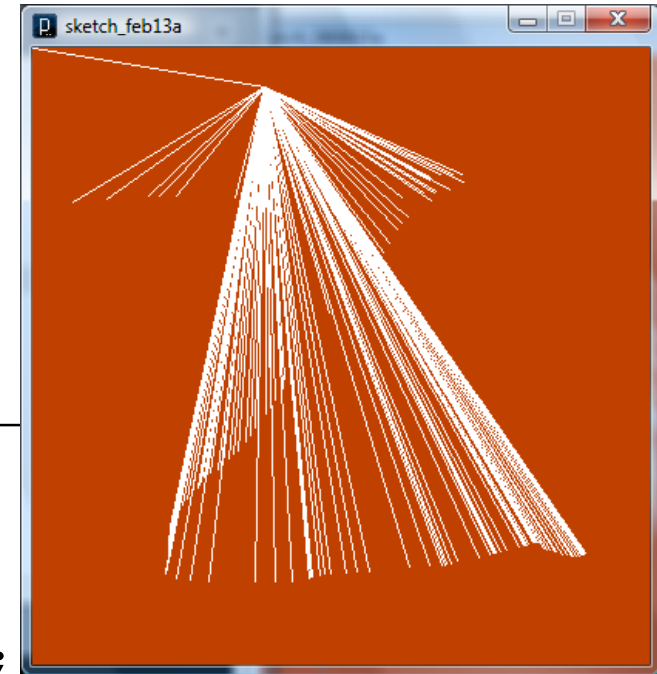
Eksempel, "Basic Mode"

```
size(400, 400);  
background(192, 64, 0);  
stroke(255);  
line(150, 25, 270, 350);
```



Eksempel, "Continuous Mode"

```
void setup( ) {  
  size(400, 400);  
  stroke(255);  
  background(192, 64, 0);  
}  
void draw( ) {  
  line(150, 25, mouseX, mouseY);  
}
```



Kursus materiale

- Processing web-site
 - <http://processing.org>
 - <http://processing.org/reference>
 - <http://processing.org/learning>
- Processing-programmet (PDE'en): Help, Examples, ...
- En god bog (med pdf-smagsprøve):
 - Processing – A Programming Handbook for Visual Designers and Artists, Casey Reas and Ben Fry, MIT press,
- Kursets hjemmeside:
 - <http://akira.ruc.dk/~jmid/crash/>

PDE'en og processing.org

- Brug <http://processing.org> – den er fyldt med information
- Brug PDE'en
 - 'Referencen' er indbygget:
 - Marker et ord
 - Højreklik
 - Vælg 'Find in Reference'

Let the crash begin

Structure 1: Kode-elementer

- Kommentarer - to måder
 - `//` dette er en kommentar på een linie
 - `/*` dette er en kommentar der strækker sig over flere linier `*/`
- Funktioner
 - det meste af sproget er realiseret i funktioner, fx
 - `size()`, `background()`, `point()`, `line()`, `ellipse()`, ...
 - parameter
 - angives for funktion
 - adskilles af kommaer, hvis flere parametre
 - fx
 - `/* her kommer en grå baggrund på en sketch på 200 gang 200 punkter */`
 - `size(200,200);`
 - `background(102);`

Udtryk (expression)

- element i sproget der har en værdi:
 - 5
 - 17.7
 - 3 + 5
 - ((3 + 5) * 7) + 1
 - "en prøve"
 - "dette" + " er " + "en prøve"

 - 6 > 3
 - 6 < 3

Sætning (statement)

- enhed i sproget med bestemt betydning/effekt
- afsluttet med semikolon ";"
- fx funktionskald som:
 - `background(102);`
 - `background((20 - 3) * 7 - 17);`
 - `println("dette" + " er " + "en
prøve");`

Udskrivning - 2 steder

- på sketch'en

- `point()`, `line()`, `ellipse()`, ...

- på konsollen

- `print()`, `println()`

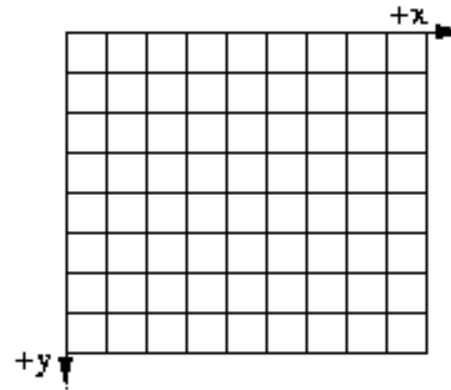
```
println("Processing...");  
println(10+20+30);  
print(10);  
println(20);  
println(30);  
  
println("dette" + " er " + "en prøve");
```

Structure 1 Opgaver

- Opg A
 - Skriv et program der udskriver værdien af udtrykkene
 - $3+4*2$
 - `30 + " kroner"`
 - sæt passende kommentarer ind
- Bogen s 21 opg 2
 - (Skriv et program der danner et $640*480$ vindue med sort baggrund)

Shape 1: Koordinater

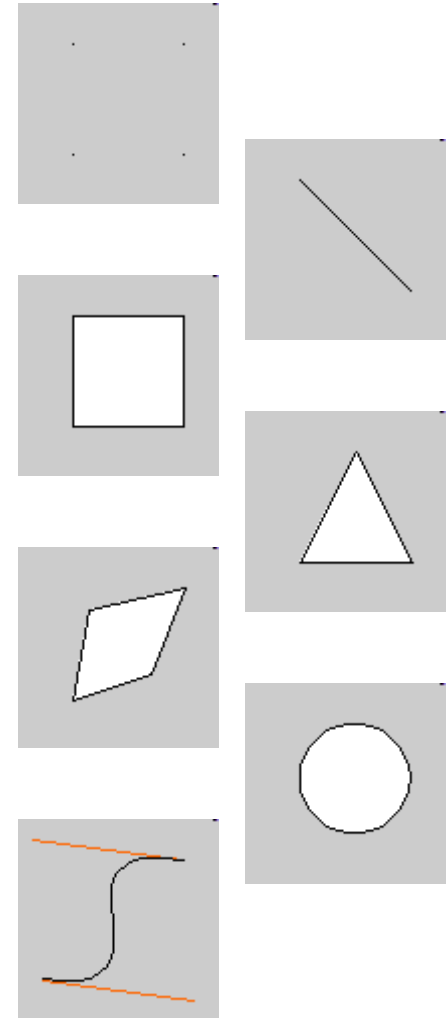
- Koordinater i display-vindue



- Egenskaber ved vindue
 - size(),
 - background(),

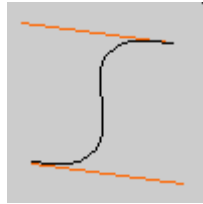
Primitive former

- Punkt og linie
 - `point()`,
 - `line()`,
- Flader
 - `rect()`,
 - `triangle()`,
 - `quad()`,
 - `ellipse()`,
- Kurver
 - `bezier()`



Kurver???

- Kurver
 - bezier()



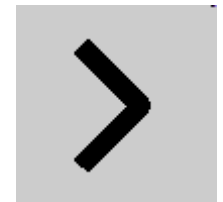
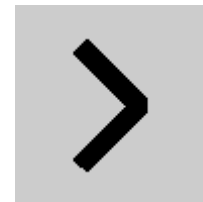
```
noFill();
stroke(255, 102, 0);
line(85, 20, 10, 10);
line(90, 90, 15, 80);
stroke(0, 0, 0);
bezier(85, 20, 10, 10, 90, 90, 15, 80);
```

- Hvad er det?
 - check matematikken
 - eller eksperimenter:

```
void draw(){
  background(200);
  noFill();
  stroke(255, 102, 0);
  line(85, 20, mouseX, mouseY);
  line(90, 90, 15, 80);
  stroke(0, 0, 0);
  bezier(85,20,mouseX, mouseY, 90, 90, 15, 80);
}
```

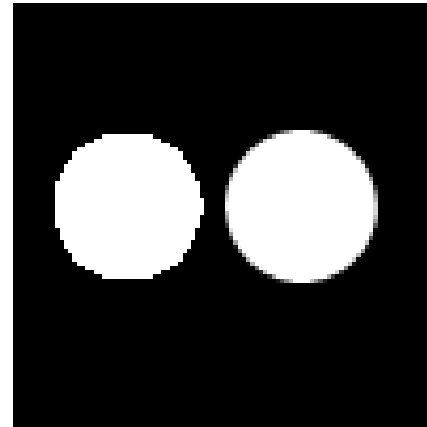
Egenskaber

- Fyld-farve
 - `fill()`
 - `noFill()`
- Streg-farve
 - `stroke()`
 - `noStroke()`
- Streg-tykkelse
 - `strokeWeight()`
- Form på streg-ende
 - `strokeCap()`
 - `ROUND`, `SQUARE`, eller `PROJECT`
- Form på streg-samling
 - `strokeJoin()`
 - `MITER`, `BEVEL`, eller `ROUND`



Egenskaber

- Pæn tegning
 - `smooth()`
 - tegn med "blød" grafik altså uden hakker på kanter og streger
 - (anti-aliased)
 - `noSmooth()`
 - slå smooth fra
 - (mere effektivt)
- Ændring af parameterbetydning
 - `ellipseMode()`
 - `rectMode()`



Shape 1 Opgaver

- Opg A
 - Tegn en tændstiksmænd
- Opg B
 - Giv ham en mave - fx som en trekant
- Opg C
 - Ret tændstiksmænd, så han tegnes med tykke streger med afrundede kanter og samlinger
- Opg D
 - Tegn et stort "S", der fylder det meste af vinduet
- Opg E
 - Tegn på sort baggrund et hvidt øje med sort iris og hvid pupil

Data 1: Data og variable

- Data
 - information på digital form
 - mange forskellige slags
 - numre
 - tekst
 - datoer
 - billeder
 - ...
- Datatype
 - en bestemt slags data
- Datatyper i Processing, bl.a.
 - **int**
 - heltal
 - 17
 - **float**
 - decimaltal
 - 17.24
 - **boolean**
 - sandhedsværdi
 - true,
 - false

Data 1: Data og variable

- **øvrige datatyper(som vi vender tilbage til)**
 - **primitive**
 - `color`
 - `char`
 - `byte`
 - **sammensatte**
 - `String`
 - `Array`
 - `Object`

Data 1: Data og variable

- Variabel
 - en slags pladsholder af given datatype
 - `int X;`
- Tilskrivning (assignment) "="
 - `X = 17;`
- Processing variable
 - `width, height`

```
int x; // Declare the variable x of type int
float y; // Declare the variable y of type float
boolean b; // Declare the variable b of type boolean
x = 50; // Assign the value 50 to x
y = 12.6; // Assign the value 12.6 to f
b = true; // Assign the value true to b
```

```
int x = 50;
float y = 12.6;
boolean b = true;
```

Data 1 Opgaver

- Bogen s 41 opg 1 og 2
- Opg A
 - Hvordan tegner man en streg fra øverste venstre hjørne til nederste højre af vinduet - som virker uanset hvad size er blevet kaldt med?
- Opg B
 - Erklær to variable X og Y og sæt X til den halve bredde på vinduet og Y således at den er 1.2 større X . Tegn en cirkel med radius på $1/4$ af vindues-bredden og centrum i (X,Y)
- Opg C
 - Test Opg B med forskellige vindues-størrelser

Math 1: Aritmetik

- Aritmetiske udtryk kan skrives med +, -, *, /, %, og paranteser
 - + (addition)
 - - (minus)
 - % (modulo)
 - / (divide)
 - (multiply)
 - ()
- Hvad er "operator precedens"?
 - *, /, %
 - +, -
 - =

Aritmetik

- Forkortet skrivemåde ved tilskrivning ++, --, +=, -=, *=, /=
 - ++ (increment)
 - -- (decrement)
 - += (add assign)
 - -= (subtract assign)
 - *= (multiply assign)
 - /= (divide assign)
- Funktioner
 - ceil()
 - floor()
 - round()
 - abs()
 - sq()
 - sqrt()
 - pow()
 - min()
 - max()

Math 1 Opgaver

- Bogen side 50 opg 1 og 2
- Opg A
 - Hvad udskriver følgende program

```
int X=0;
println(X++);
println(X++);
X=X+9;
println(X);
X=X-2;
println(X);
X=X/3;
println(X);
X=X*6;
println(X);
```

Dagens debugging