Vad är programmering

Jonas Lindemann

Vad är en dator?

- Består av ett antal delar som samverkar
- Processorn
  - Utför instruktioner (addera, skriv till minne, flytta minne…)
- Minnet
  - Lagrar de instruktioner som skall utföras
  - Lagrar information som används för att visa text grafik eller spela upp ljud

Vad är ett program?

- Ett antal av dessa instruktioner tillsammans utgör ett program eller en applikation
- Program lagras permanent på hårddisken i datorn när det inte används
- Operativsystemet (Windows, Mac OS X eller Linux) också ett program laddar programmet till datorns minne och pekar processorn till rätt position i minnet

Vad är programmering?

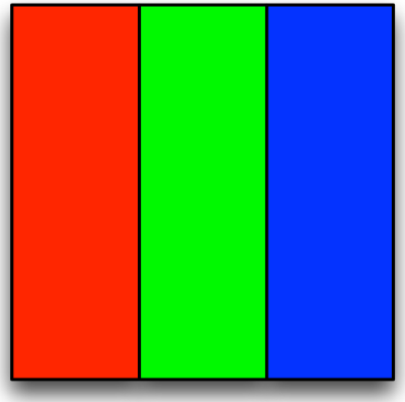
- ”Att gå från tanke till instruktioner i en dator”
- De instruktioner som datorn utför kallas för maskinkod
  - Mycket komplicerat att skriva dessa
  - Behöver hjälp att skapa dessa
- För att göra det enklare använder vi istället ett programspråk
  - Använder instruktioner / funktioner som vi har lättare att förstå
  - Översätts till maskinkod genom ett speciellt program kallat kompilator
- I denna aktivitet kommer vi att använda ett programspråk och kompilator kallad Processing

Processing

- Programspråk, kompilator och utvecklingsmiljö
- Gratis
- Finns för Windows, Mac OS X och Linux
- Hämtas på www.processing.org

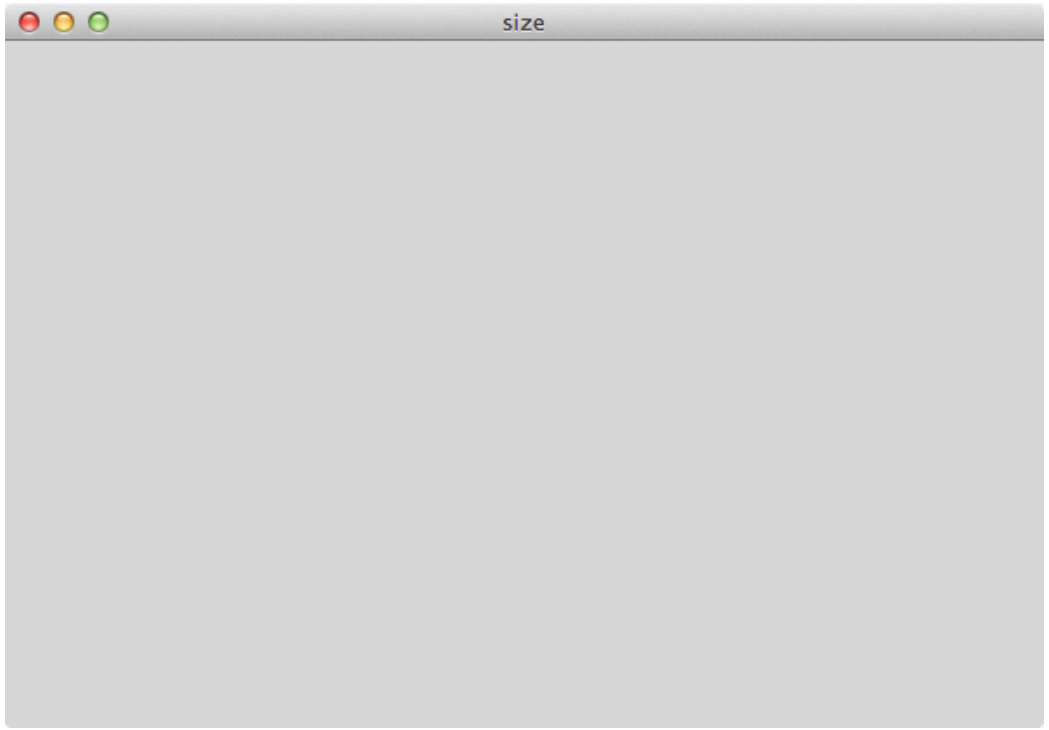
Rita på skärmen

- Bildskärmar är uppbyggda av många små bildpunkter
- Varje bildpunkt består av 3 färger rött, grönt och blått
- Genom att blanda dessa kan man få fram alla färger
- Uppritning genom att anropa funktioner som färglägger bildskärmspunkterna, så att det ser ut som linjer, cirklar och rektanglar.



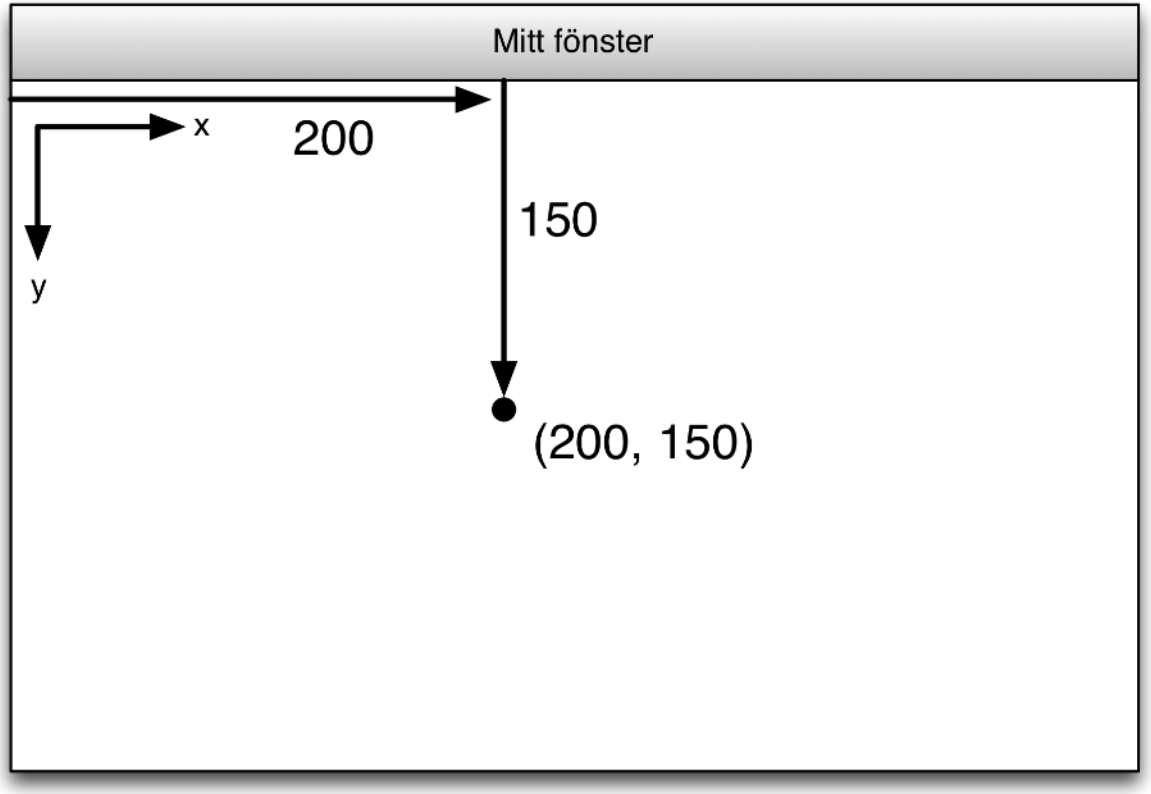
Rityta

- Uppritning i Processing sker till ett fönster med en rityta vi kan använda för uppritning
- Storleken på fönstret sätt med funktionen
  - size(600,400);



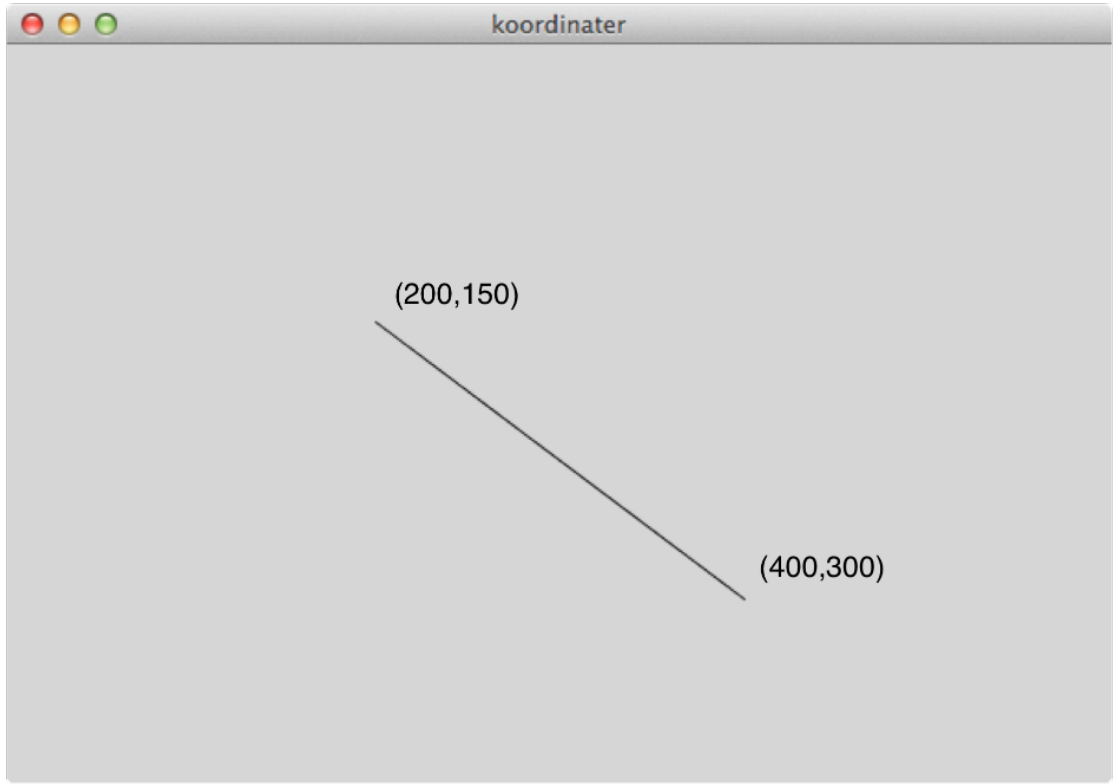
Skärmkoordinatsystem

- För att datorn skall veta var någonstans vi vill rita i fönstret måste vi kunna ange detta på något sätt.
- Processing använder ett s.k. skärmkoordinatsystem för detta.
- Övre högra hörnet är (0,0)



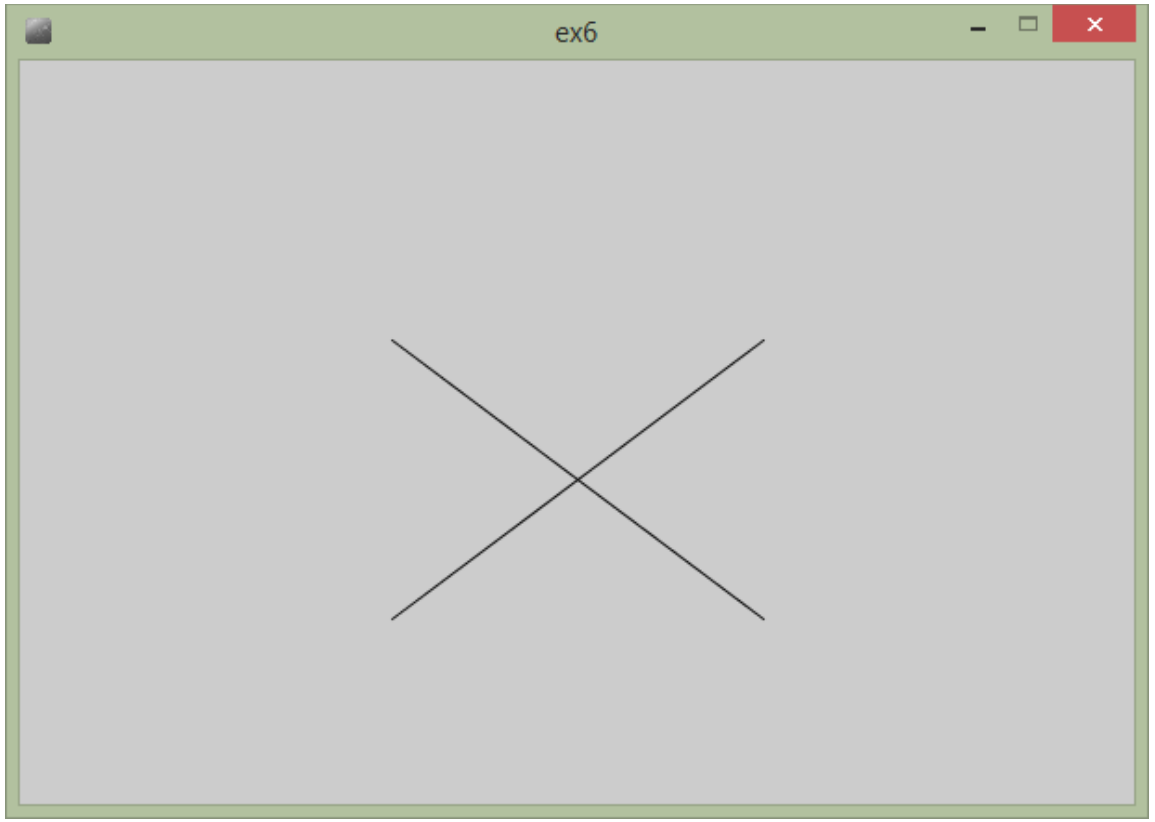
Skärmkoordinatsystem – Exempel 1

```
size(600,400);
line(200,150,400,300);
```



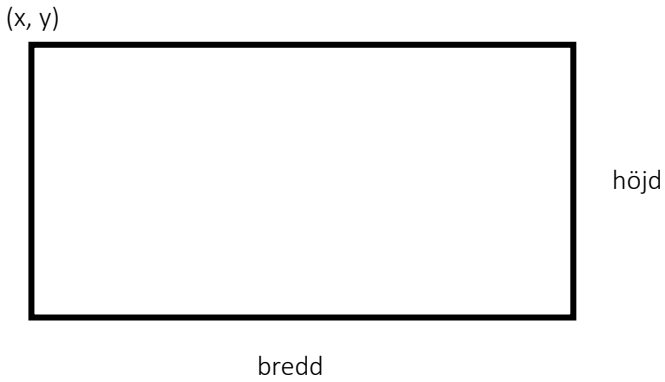
Skärmkoordinater – Exempel 2

```
size(600,400);
line(200,150,400,300);
line(400,150,200,300);
```

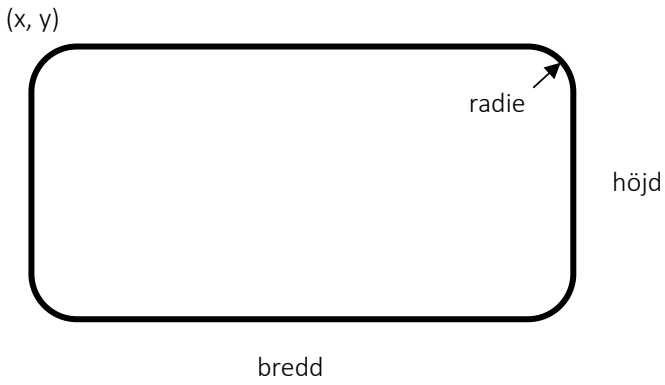


Fler funktioner för uppritning

rect(x, y, bredd, höjd);

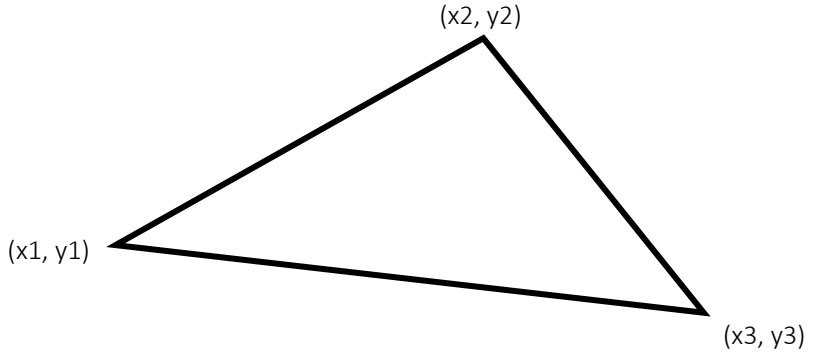


rect(x, y, bredd, höjd, radie);

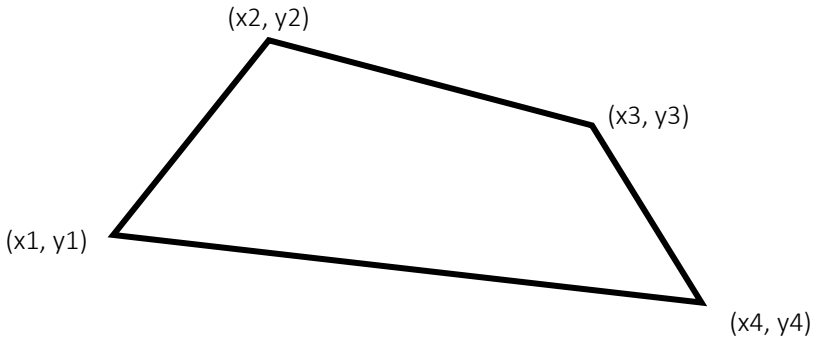


Fler funktioner för uppritning

triangle(x1, y1, x2, y2, x3, y3);

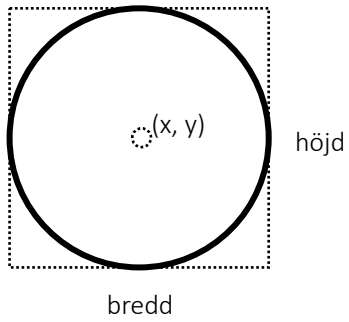


quad(x1, y1, x2, y2, x3, y3, x4, y4);



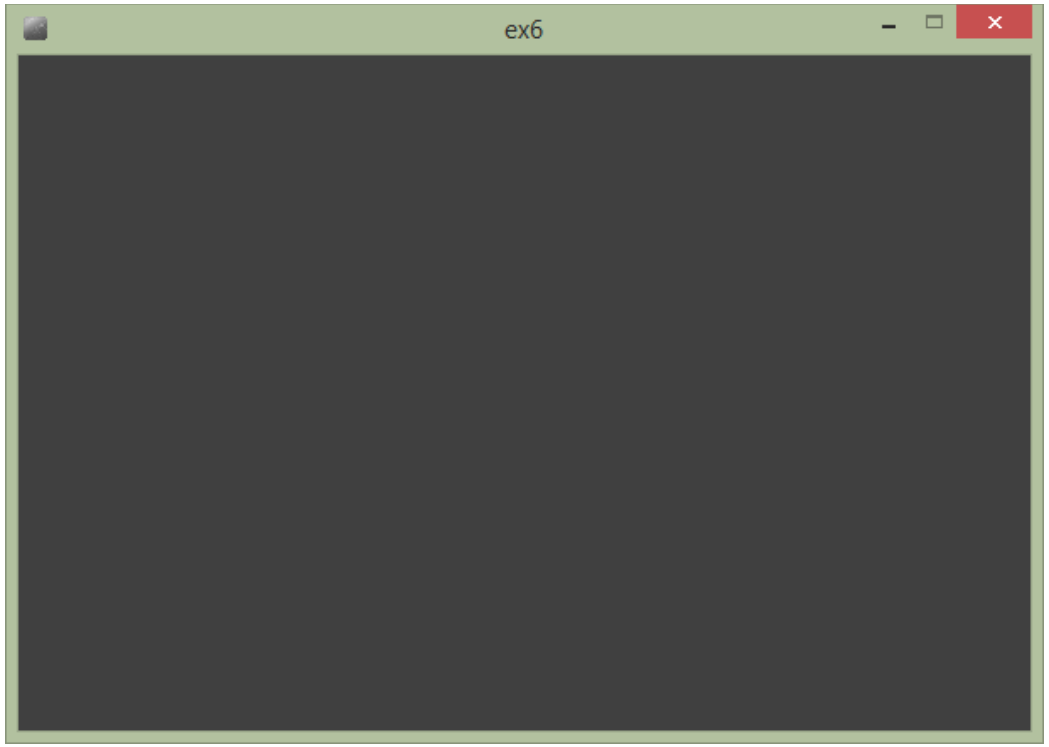
Fler funktioner för uppritning

ellipse(x, y, bredd, höjd);



Bakgrundsfärg

```
size(600,400);
background(64);
```

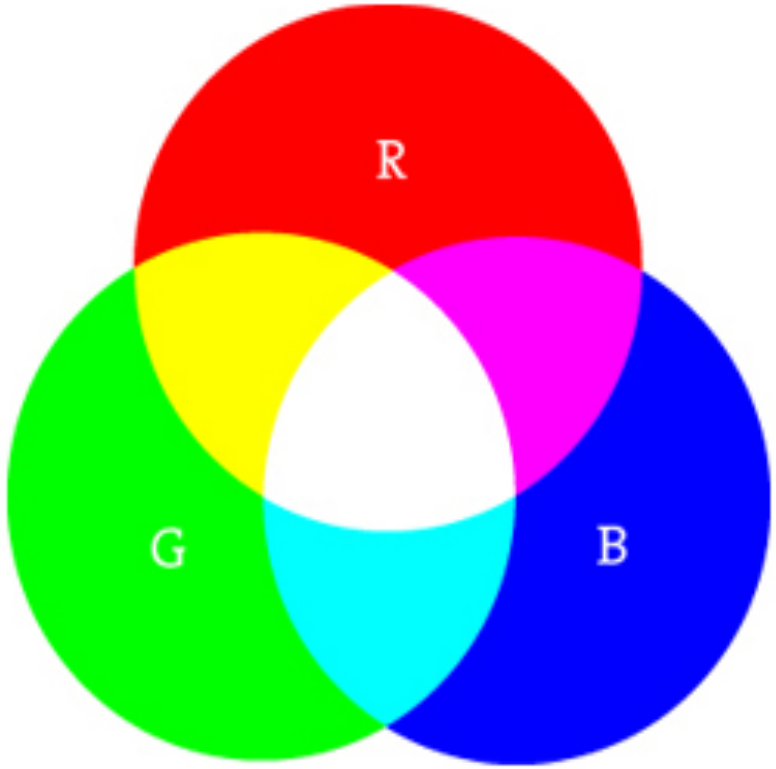


Bakgrundsfärg

```
size(600,400);
background(255);
```

Blanda färger i Processing

- Färger anges med värden för rött, grönt och blått
- Max intensitet är 255
- Minsta intensitet är 0
- Svart = (0,0,0)
- Vitt = (255,255,255)
- Gult = (255,255,0)

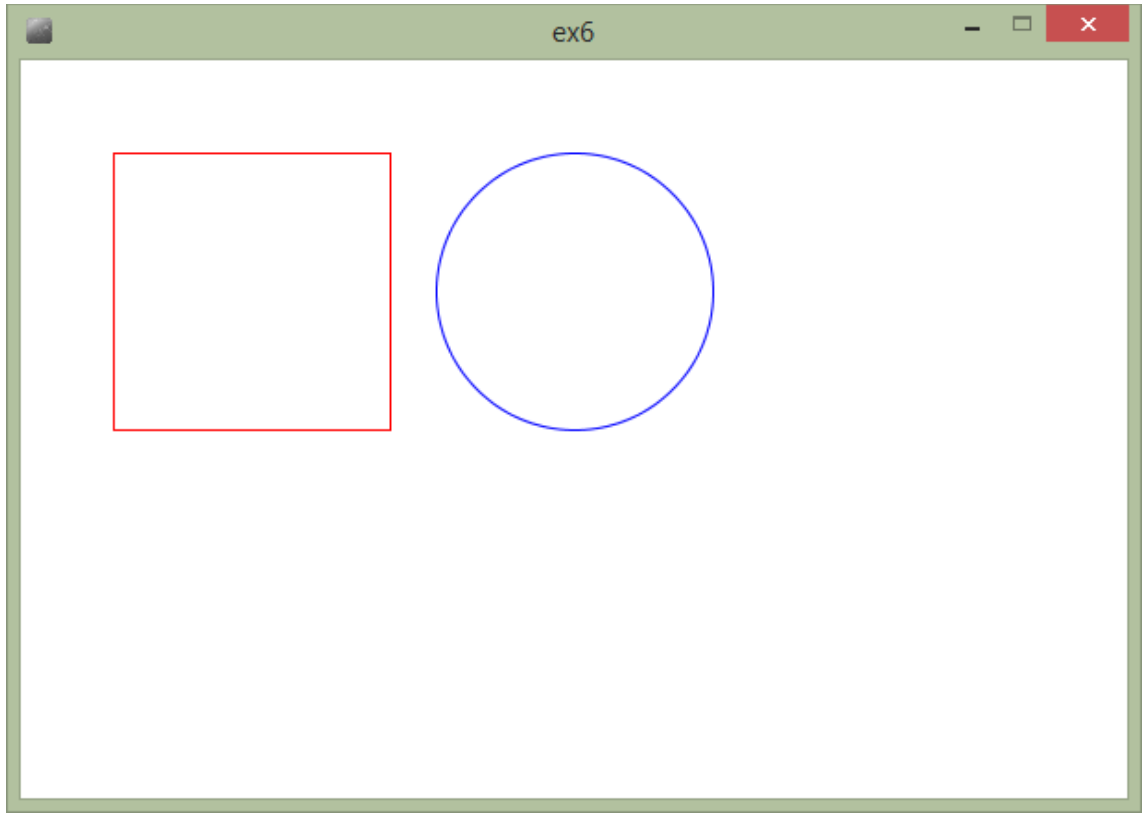


Bakgrundsfärg

```
size(600,400);
background(255,255,0);
```

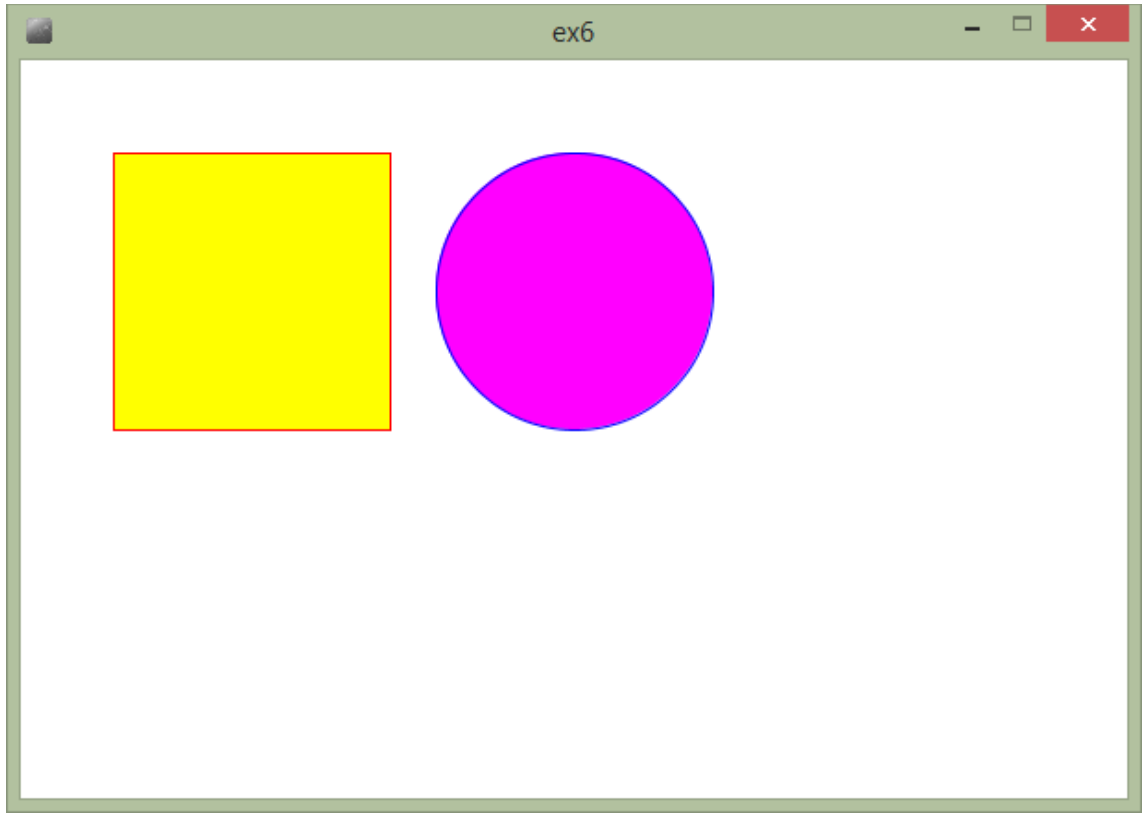
Linjefärg

```
size(600,400);
background(255,255,255);
stroke(255,0,0);
rect(50,50,150,150);
stroke(0,0,255);
ellipse(300,125,150,150);
```



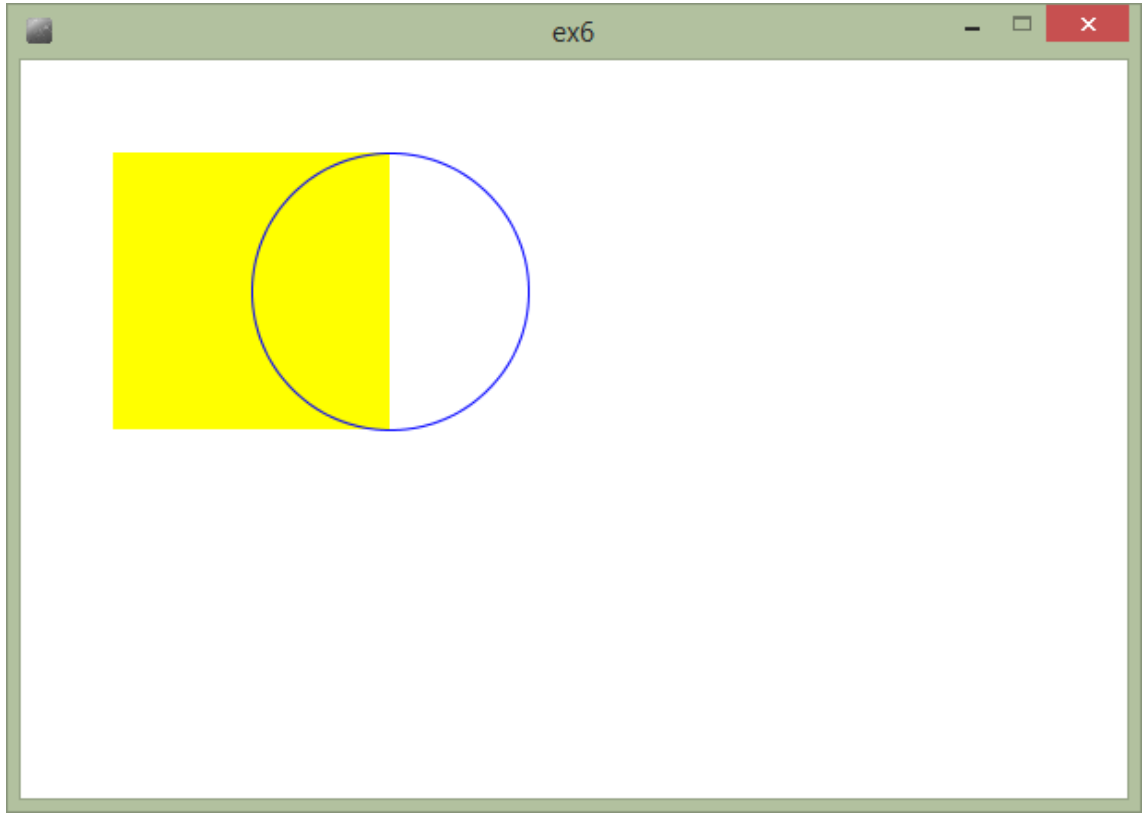
Fyllningsfärg

```
size(600,400);
background(255,255,255);
stroke(255,0,0);
fill(255,255,0);
rect(50,50,150,150);
stroke(0,0,255);
fill(255,0,255);
ellipse(300,125,150,150);
```



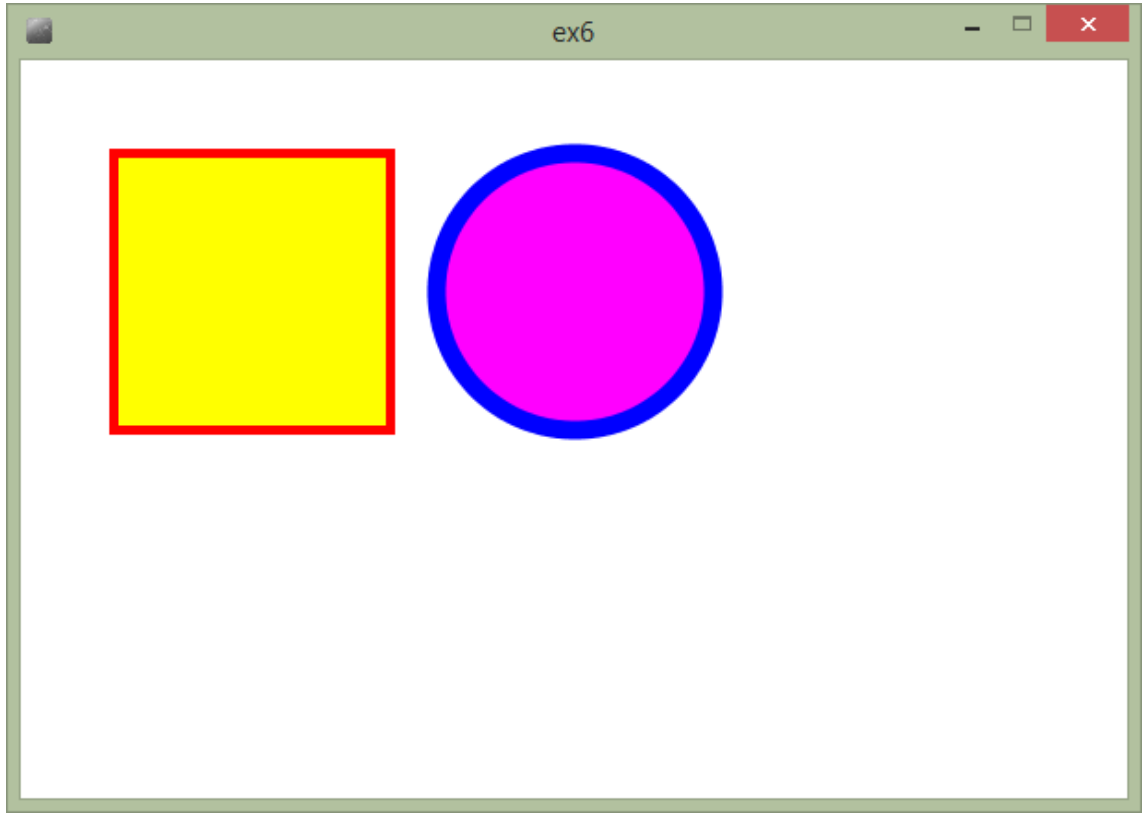
Fyllningsfärg

```
size(600,400);
background(255,255,255);
noStroke();
fill(255,255,0);
rect(50,50,150,150);
stroke(0,0,255);
noFill();
ellipse(200,125,150,150);
```



Linjetjocklek

```
size(600,400);
background(255,255,255);
strokeWeight(5);
stroke(255,0,0);
fill(255,255,0);
rect(50,50,150,150);
strokeWeight(10);
stroke(0,0,255);
fill(255,0,255);
ellipse(300,125,150,150);
```



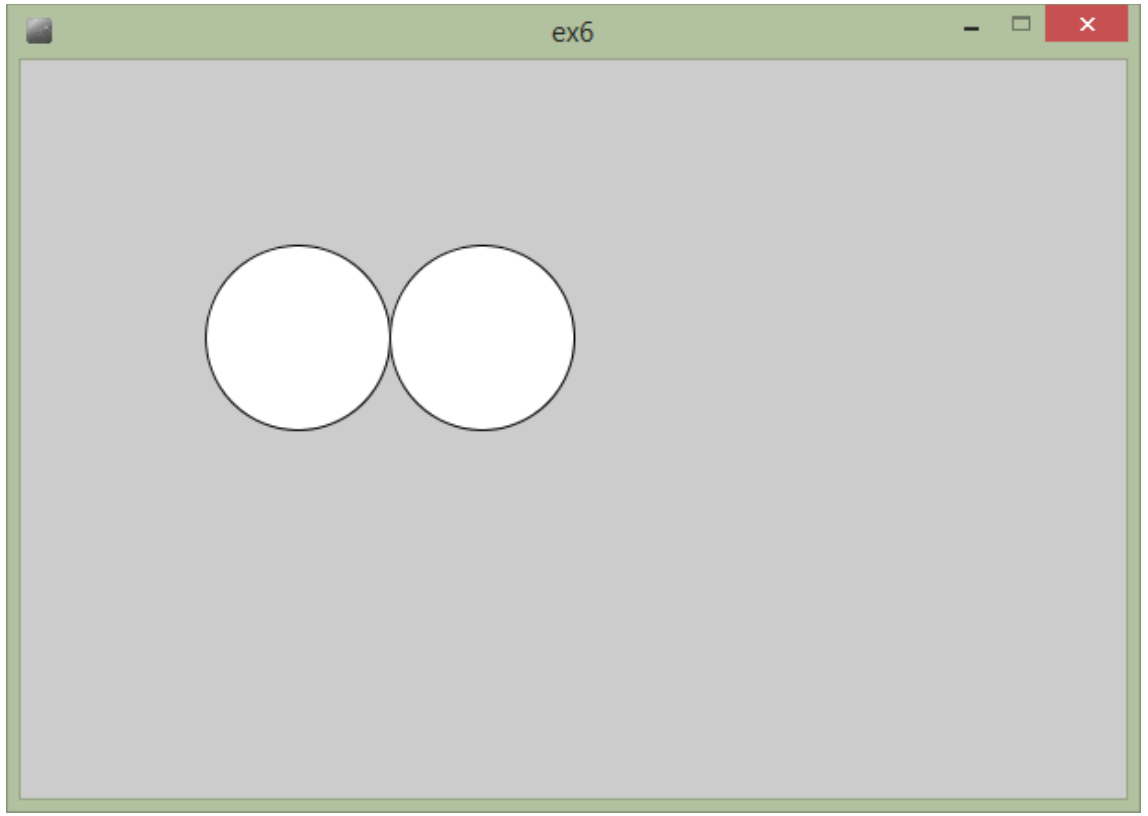
Minne lagring och variabler

- Variabler lagrar siffror och text i datorns minne
- Ersätter siffror som indata i funktioner
- Förenklar hanteringen av information.
- Fungerar som en byrålåda med etikett
  - Lådan innehåller värdet
  - Etiketten gör att vi kan hitta lådan igen.
- 3 huvudsakliga variabler i Processing

Variabeltyp	Exempel
int eller heltal	1, 2, 3, 4, 5
float eller decimaltal	1.5, 43.65, 0.1
String eller text	”Detta är en text”

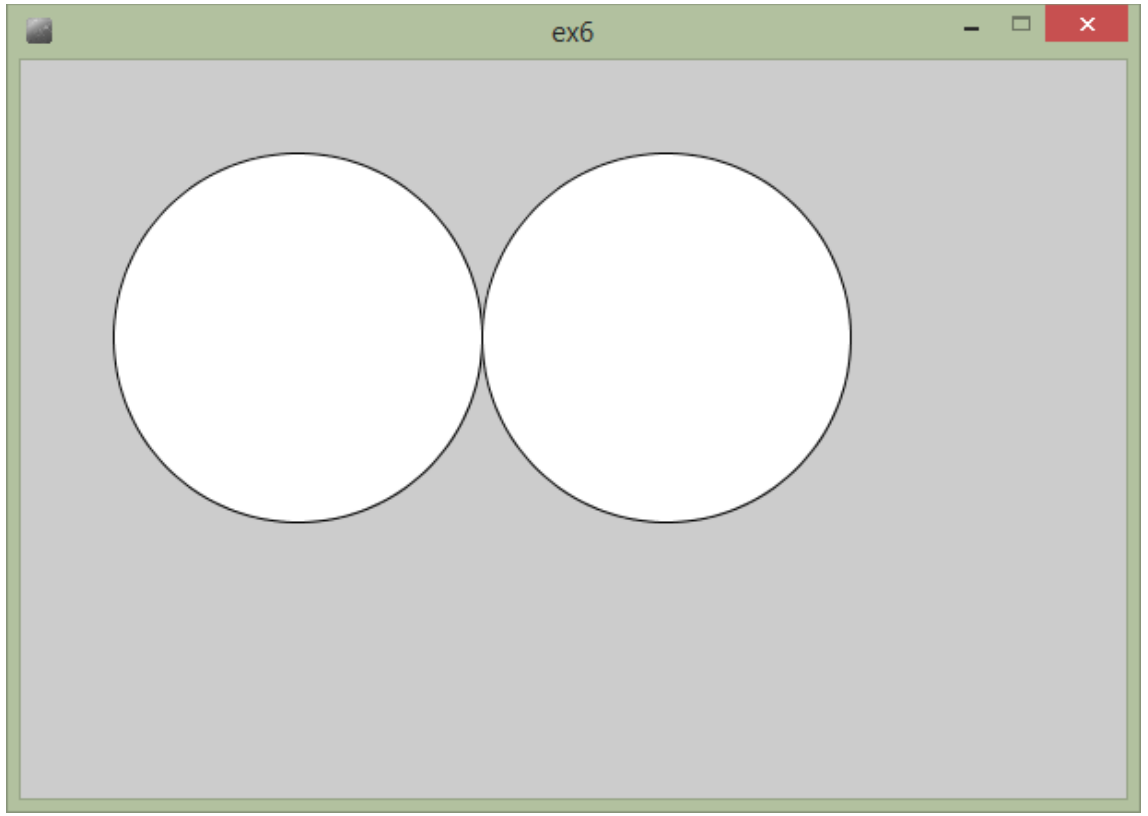
Variabler – Exempel 1

```
int d = 100;
size(600,400);
ellipse(50,50,d,d);
ellipse(50+d,50,d,d);
```

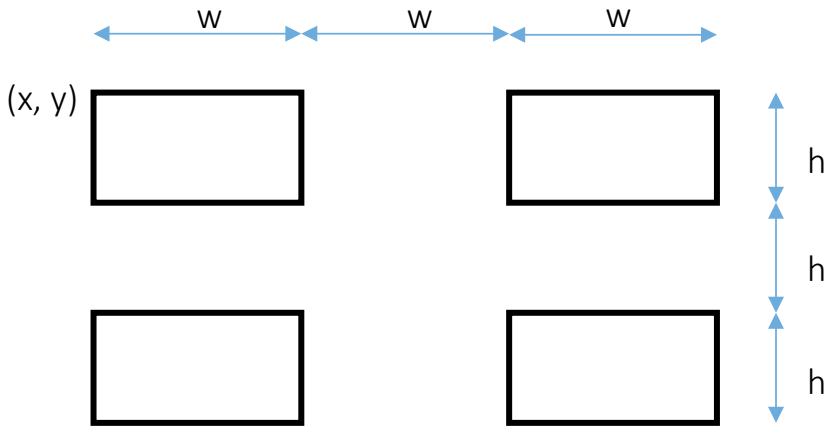


Variabler – Exempel 2

```
int d = 200;
size(600,400);
ellipse(50,50,d,d);
ellipse(50+d,50,d,d);
```

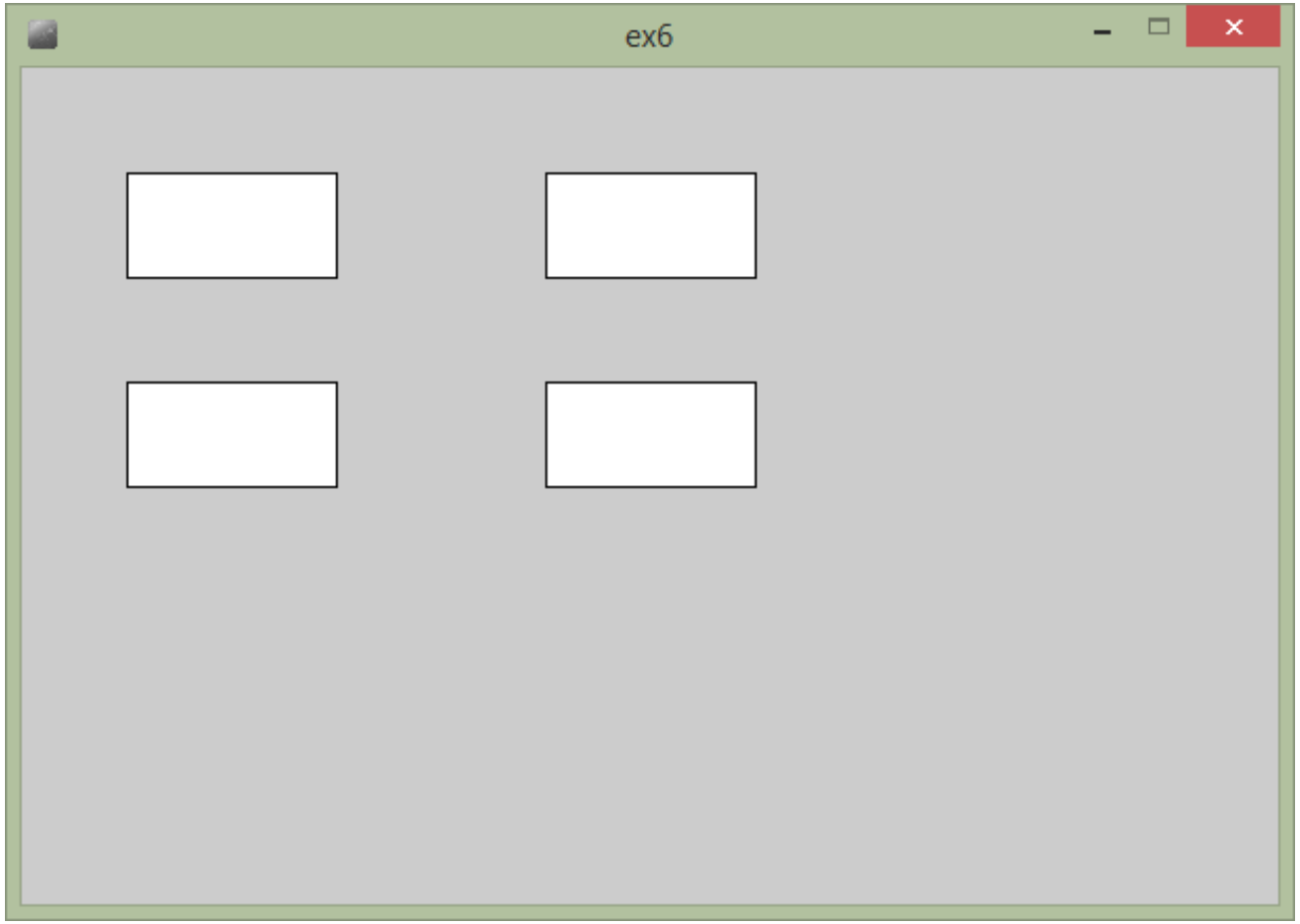


Variabler – Exempel 3



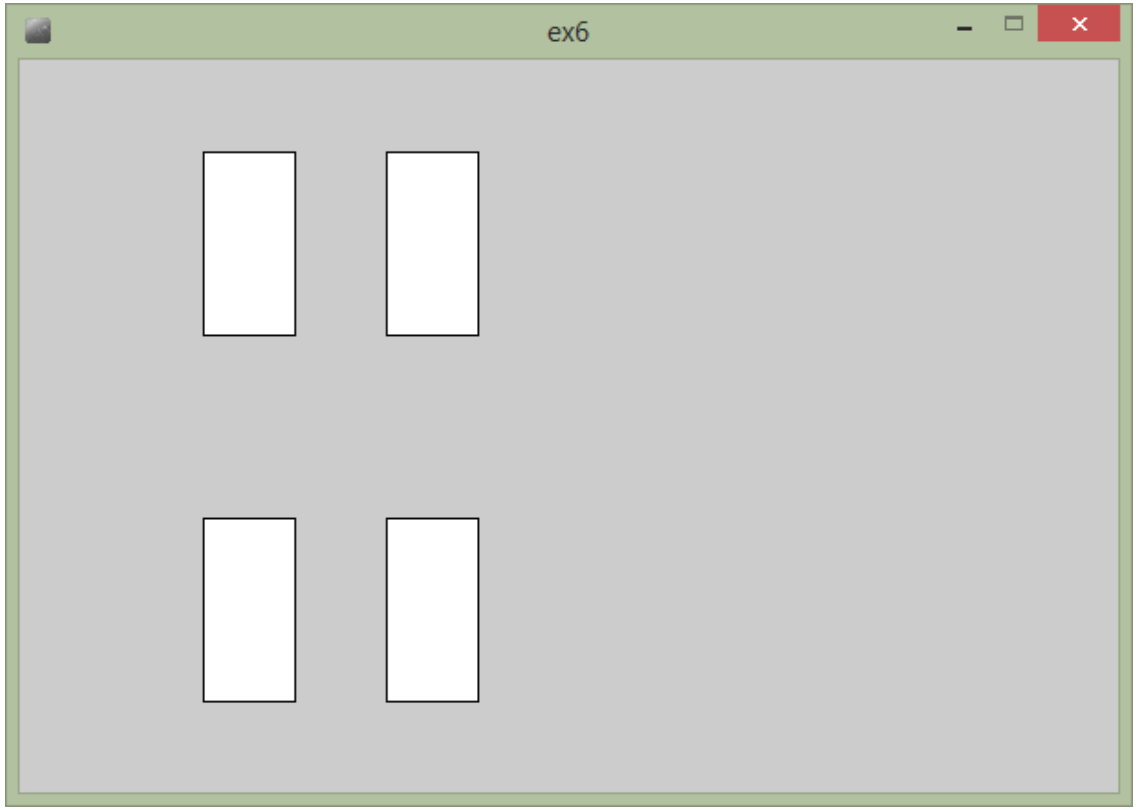

```
size(600,400);

float w = 100;
float h = 50;
float x = 50;
float y = 50;

rect(x, y, w, h);
rect(x + 2*w, y, w, h);
rect(x, y + 2*h, w, h);
rect(x + 2*w, y + 2*h, w, h);
```

Variabler – Exempel 3



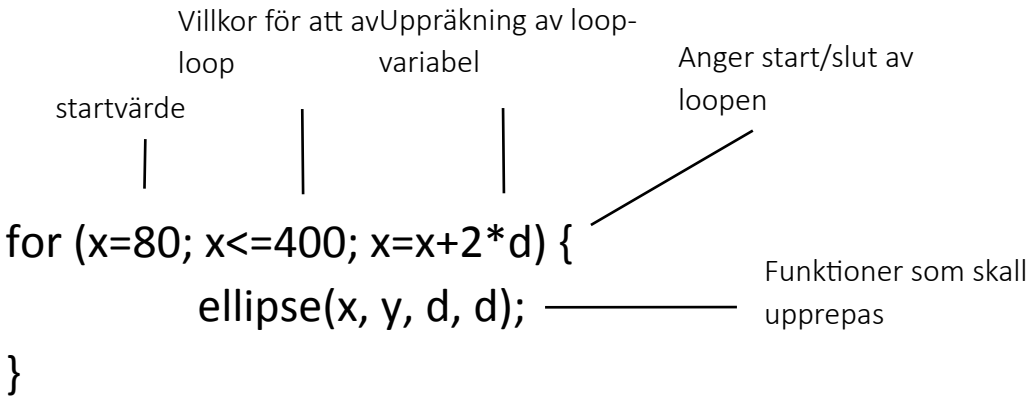
Variabler – Exempel 3

```
size(600,400);

float w = 50;
float h = 100;
float x = 100;
float y = 50;

rect(x, y, w, h);
rect(x + 2*w, y, w, h);
rect(x, y + 2*h, w, h);
rect(x + 2*w, y + 2*h, w, h);
```



Upprepning

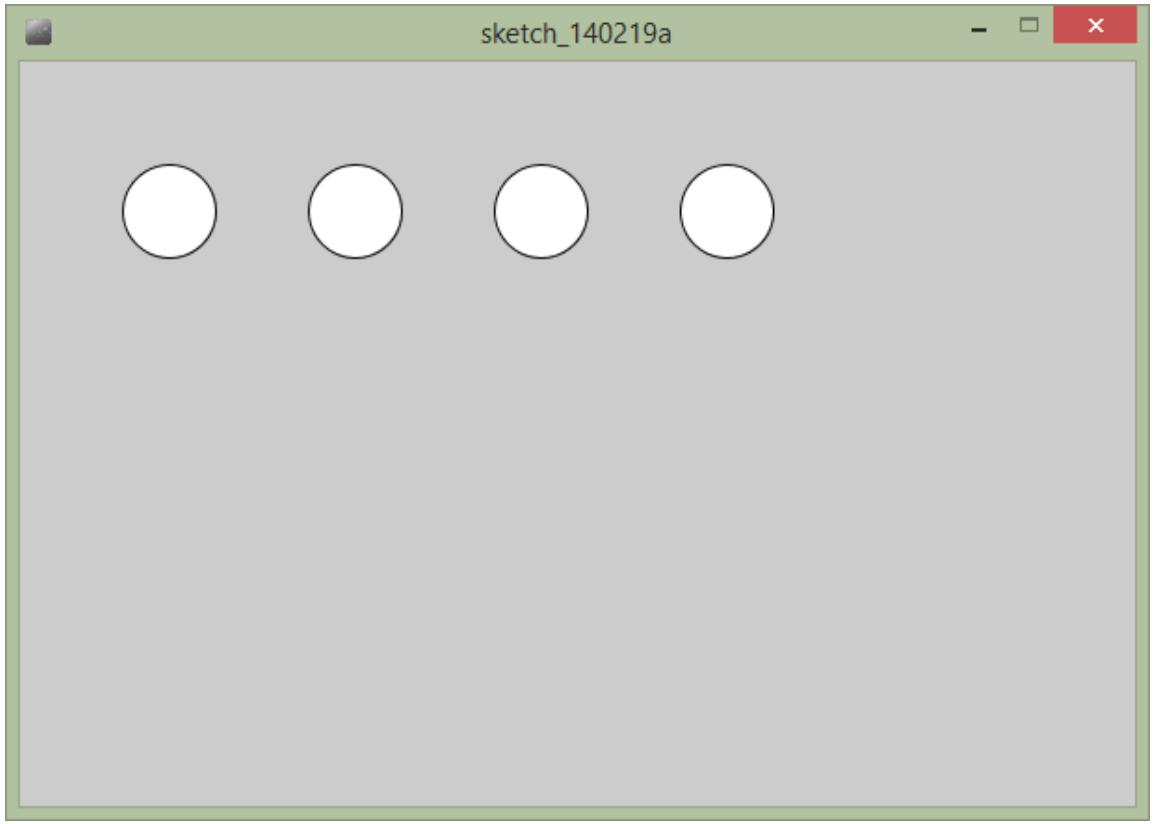
- Ofta måste man upprepa funktioner många gånger i ett program
- För att förenklar detta kan man använda s.k. loopar eller upprepningssatser
- I Processing använder vi oss av for-satsen



x kommer att få värdena *80, 180, 280, 380*

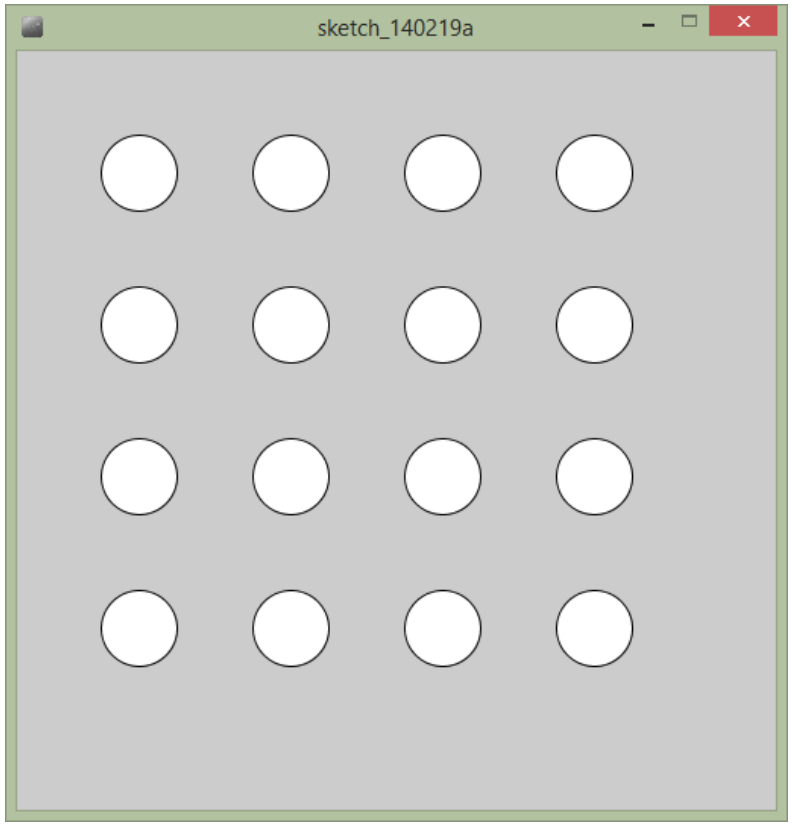
Upprepning – Exempel 1

```
size(600,400);

float d = 50;
float x = 80;
float y = 80;

for (x=80; x<=400; x=x+2*d) {
  ellipse(x, y, d, d);
}
```



Upprepning – Exempel 2

```
size(500,500);

int d = 50;
int x = 80;
int y = 80;

for (x=80; x<=400; x=x+2*d) {
  for (y=80; y<=400; y=y+2*d) {
    ellipse(x, y, d, d);
  }
}
```

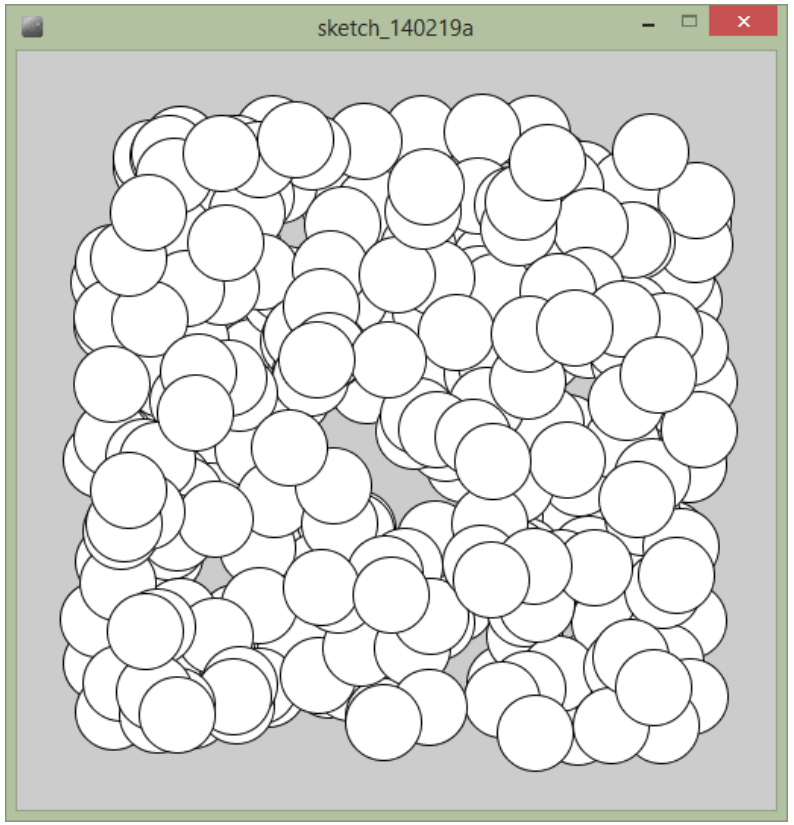


Slumptal

- Något datorn är bra på är faktiskt att kasta tärning
- Resultatet av tärningskasten i datorn kallas för slumptal.
- …placera ut monster i labyrinter eller rita upp intressanta mönster på skärmen…
- Funktionen random(…) kan skapa slumptal

Slumptal exempel

```
size(500,500);

float d = 50;
float x = 80;
float y = 80;

for (int i=1; i<=300; i=i+1) {
  x = random(400);
  y = random(400);
  ellipse(x+50, y+50, d, d);
}
```

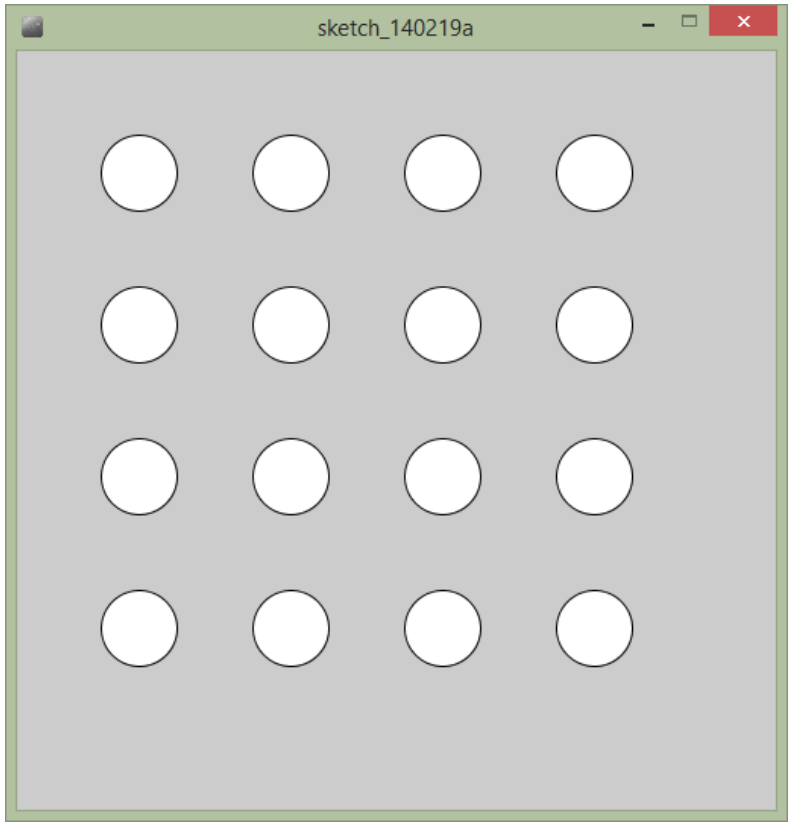


Interaktivitet

- Hittills har programmen inte innehållit någon interaktion med mus eller tangentbord
- Interaktiva program måste implementera egna funktioner
  - setup() – Anropas en gång när programmet startas
  - draw() – Anropas helat tiden under programmets körning

Interaktivitet – Exempel 1

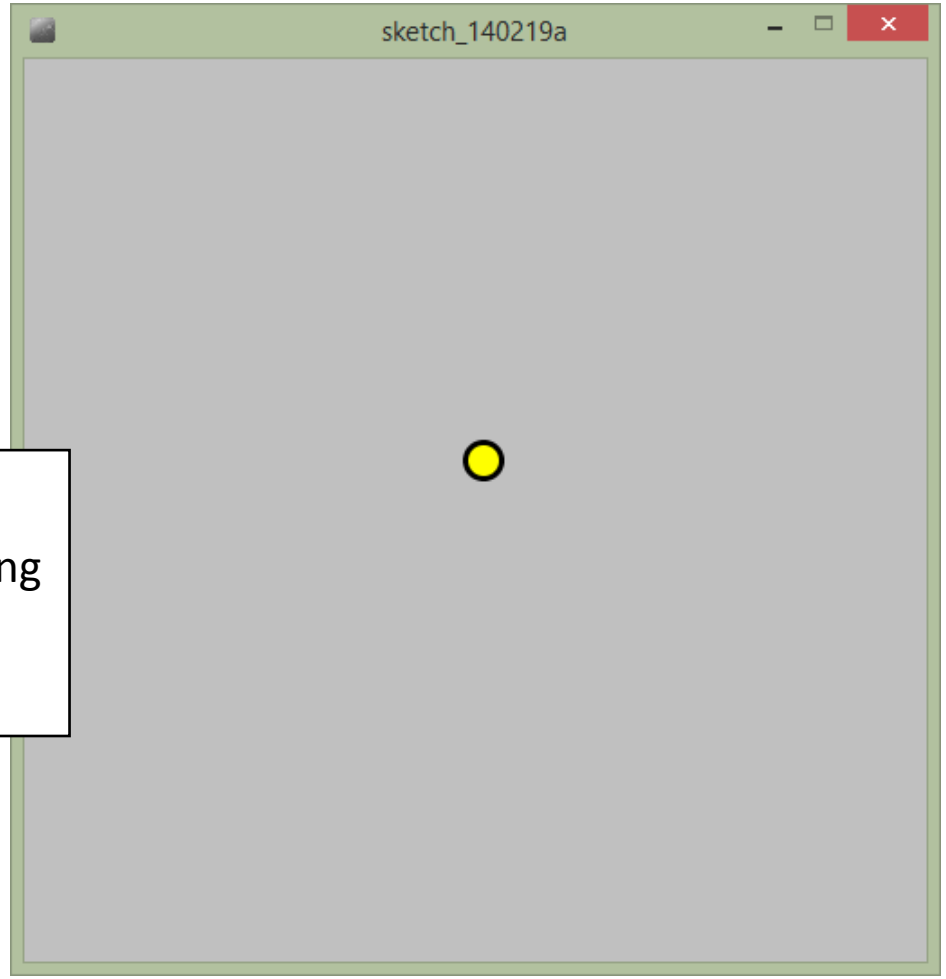
```
// Variabler som skall vara tillgängliga
// för alla funktioner i programmet.

int d = 50;
int x = 80;
int y = 80;

// setup() anropas en gång när programmet
startats.
void setup() {
  size(500,500);
}

// draw() anropas av Processing hela tiden
void draw() {
  background(192);
  for (x=80; x<=400; x=x+2*d) {
    for (y=80; y<=400; y=y+2*d) {
      ellipse(x, y, d, d);
    }
  }
}
```



Använda musen – Exempel 1

```
// Variabler som skall vara tillgängliga
// för alla funktioner i programmet.

int d = 50;
int x = 80;
int y = 80;

// setup() anropas en gång när programmet
startats.
void setup() {
  size(500,500);
}

// draw() anropas av Processing he
void draw() {
  background(192);
  fill(255,255,0);
  strokeWeight(3);
  stroke(0);
  ellipse(mouseX, mouseY, 20, 20);
}
```

mouseX och mouseY är speciall variabler i Processing som innehåller musens aktuella position



Använda musen – Exempel 2

```
// Variabler som skall vara tillgängliga
// för alla funktioner i programmet.

int d = 50;
int x = 80;
int y = 80;

// setup() anropas en gå
startats.
void setup() {
  background(192);
  size(500,500);
}

// draw() anropas av Proc      ng hela tiden
void draw() {
  strokeWeight(3);
  stroke(0);
  line(pmouseX, pmouseY, mouseX, mouseY);
}
```

pmouseX och pmouseY innehåller musens position i föregående uppritning

Villkor eller if-satser

- Viktigt att kunna utföra olika funktioner beroende på värdet på olika variabler eller funktioner
- I Processing görs detta med en if-sats
- Två varianter

Villkor

```
If (a>2) {
        // utförs när a>2
}
```

Funktioner som utförs när villkoret är uppfyllt

Villkor

```
If (a>2) {
        // utförs när a>2
} else {
        // utförs när a>2 inte är
        // uppfyllt
}
```

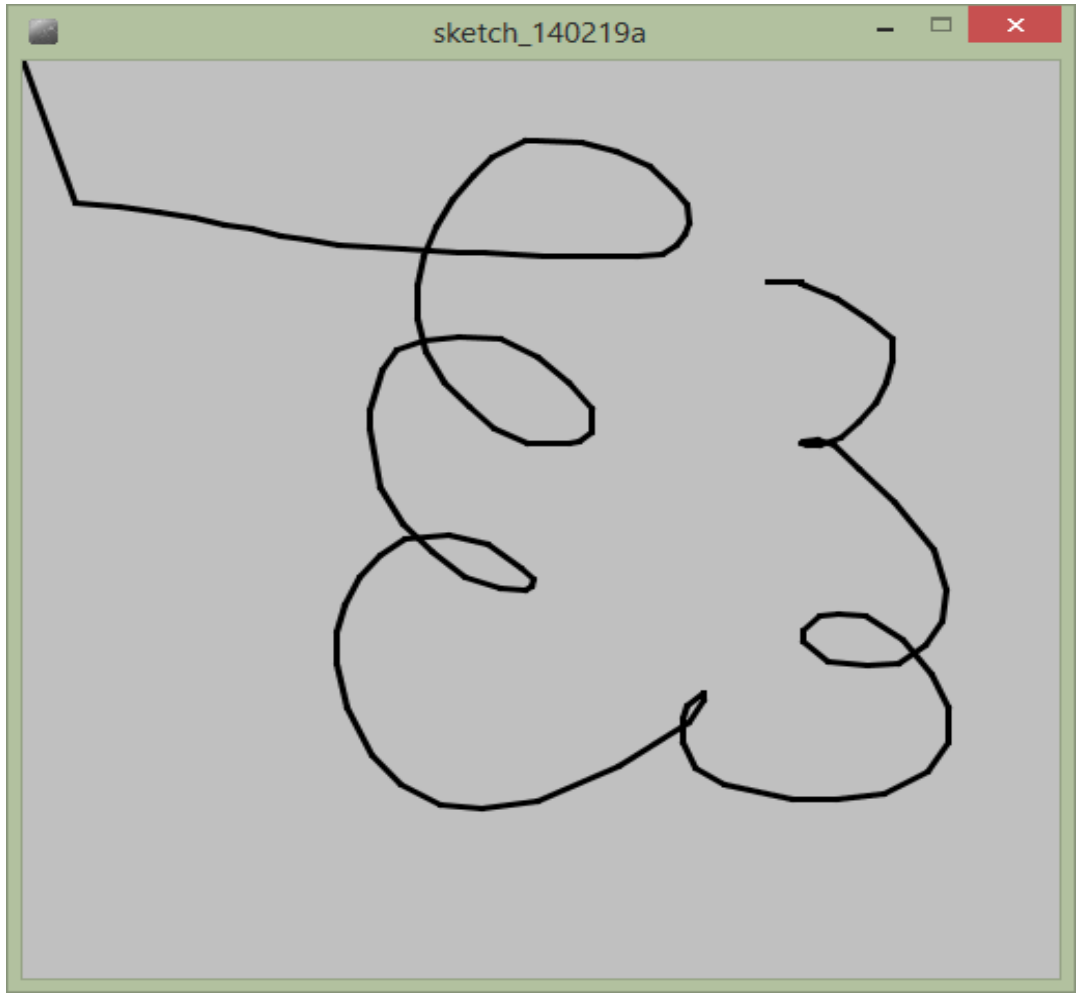
Funktioner som utförs när villkoret är uppfyllt

Använda musen – Exempel 3

```
// Variabler som skall vara tillgängliga
// för alla funktioner i programmet.

int d = 50;
int x = 80;
int y = 80;

// setup() anropas en gång när programmet
startats.
void setup() {
  background(192);
  size(500,500);
}

// draw() anropas av Processing hela tiden
void draw() {
  strokeWeight(3);
  stroke(0);
  if (mousePressed == true) {
    line(pmouseX, pmouseY, mouseX, mouseY);
  }
}
```

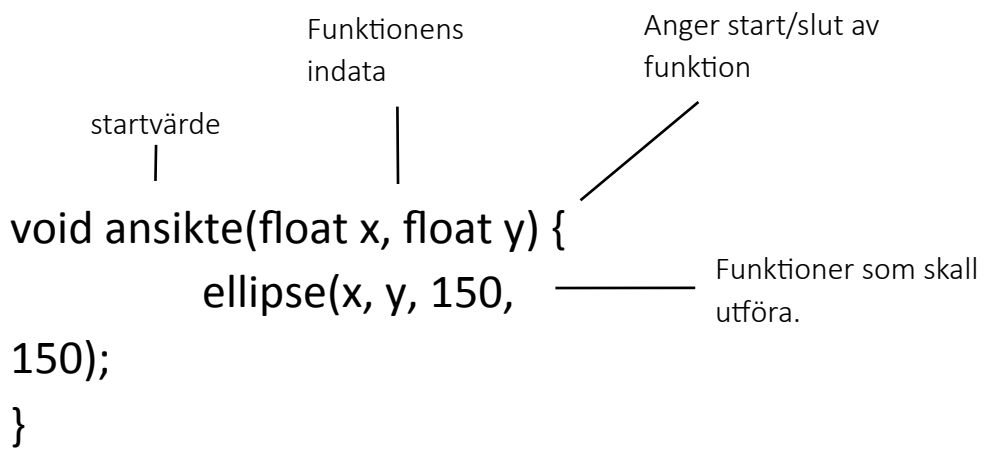
Använda musen – Exempel 4

```
// Variabler som skall vara tillgängliga
// för alla funktioner i programmet.

int d = 50;
int x = 80;
int y = 80;

// setup() anropas en gång när programmet startats.
void setup() {
  background(192);
  size(500,500);
}

// draw() anropas av Processing hela tiden
void draw() {
  strokeWeight(3);
  stroke(0);
  if (mousePressed == true) {
    if (mouseButton == LEFT) {
      line(pmouseX, pmouseY, mouseX, mouseY);
    } else {
      background(192);
    }
  }
}
```



Egna funktioner

- Ibland kan det vara bar att gruppera ihop ett antal funktioner i en egen funktion.
- I processing görs detta på följande sätt:



Egna funktioner – Exempel

```
// Egen funktion för uppritning av ansikte.
void ansikte(float x, float y) {
  strokeWeight(5);
  stroke(0,0,0);
  fill(255,255,0);
  ellipse(x,y,150,150);
  fill(0,0,255);
  ellipse(x-30,y-30,20,20);
  ellipse(x+30,y-30,20,20);
  line(x-30,y+30,x+30,y+30);
}

// setup() anropas en gång när programmet startats.
void setup() {
  background(192);
  size(500,500);
  stroke(0);
}

// draw() anropas av Processing hela tiden
void draw() {
  background(192);
  ansikte(mouseX, mouseY);
}
```