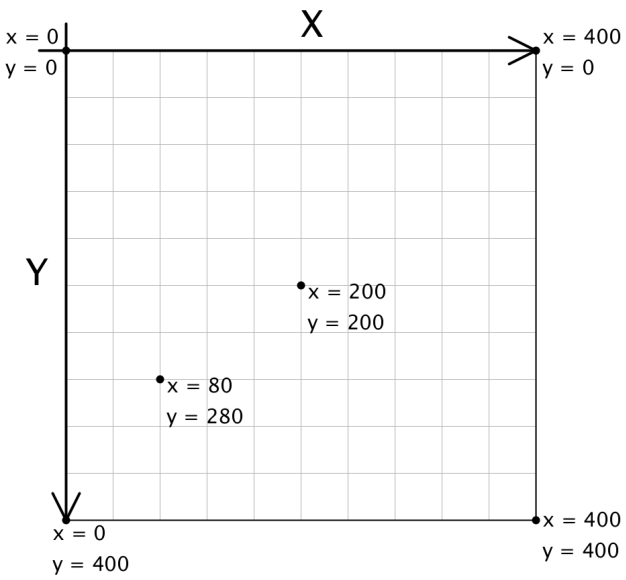
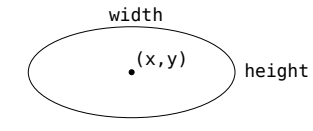
Koordinatsystemet

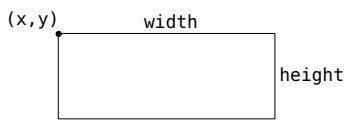
Koordinatsystemet i Processing har punktet (0,0) i øverste venstre hjørne, med x-aksen gående mod højre og y-aksen gående nedad. Det er omvendt hvad vi er vandt til i matematikken hvor y-aksen går opad.

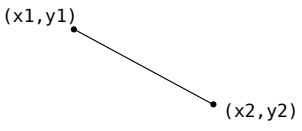


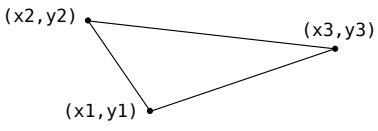
Tegnekommandoer

`ellipse(x, y, width, height)`	Tegn en ellipse. x og y værdier angiver centrum af ellipsen.
`rect(x, y, width, height)`	Tegn et rektangel. x og y værdierne angiver øverste venstre hjørne.
`line(x1, y1, x2, y2)`	Tegn en linje. `x1` og `y1` angiver den ene ende af linjen. `x2` og `y2` angiver den anden ende af linjen.
`triangle(x1, y1, x2, y2, x3, y3)`	Tegn en trekant. De tre sæt xy-koordinater angiver de tre forskellige hjørner af trekanten.
`text("tekst her", x, y)`	Skriv teksten mellem de to sæt gåseøjne. Teksten tegnes ved det angivne koordinat









Farvelægning

Farver angives som tre værdier i intervallet 0-255 der angiver mængden af rød, grøn og blå. Det kaldes RGB-farvesystemet. Søg online efter "rgb color picker" for at finde værktøjer til at finde farver.

`background(r, g, b)`	Slet alt og brug den angivne farve som baggrund
`fill(r, g, b)`	Angiv udfyldningsfarve. Efterfølgende figurer bliver tegnet i den farve.
`stroke(r, g, b)`	Angiv omrids-farve. Efterfølgende figurer bliver tegnet med omrids i den farve.
`noStroke()`	Slå tegning af omrids fra.
`noFill()`	Slå udfyldning af figurer fra, så de er helt gennemsigtige.

Variable

Definition af variabel:

```
x = 4200
```

Nu kan man bruge `x` i sit program i stedet for værdien 4200.

Senere kan variablen ændres:

```
x = 17
```

Variablen kan også bruges på højresiden, hvis man fx vil trække 3 fra:

```
x = x - 3
```

Funktioner

Definition og kald af funktioner:

```
def circleRect(x, y):
    ellipse(x, y, 20, 20)
    rect(x - 10, y - 10, 20, 20)

def scaleAdd(a, b, c):
    temporary = (a * b) + c
    return temporary

# Kald circleRect
# tallene 150 og 100 overføres til funktionen
circleRect(150, 100)

# Kald scaleAdd og aflæs dens returværdi
v = scaleAdd(1000,2,18)
```

Funktioner kan kalde andre funktioner. Funktioner kan, men behøver ikke, returnere et resultat via **`return`** keywordet.

Funktioner med og uden sideeffekter

Funktionen `circleRect` har som *sideeffekt* at tegne en ellipse og et rektangel på skærmen. Den ændrer på noget udenfor funktionen. En anden side effekt kunne være at ændre på en variabel udenfor funktionen (fx en global variabel).

Modsat ændrer `scaleAdd` ikke på noget uden for funktionen, den udfører en beregning og returnerer en værdi. Den har altså ingen *sideeffekter*.

Det er ikke enten eller, man vil ofte også have funktioner i sit program der både har sideeffekter og returnerer værdier.

Lokale variabler

En variabel der er defineret inde i en funktion eller som parametre til en funktion, er lokal for funktionen. Ændrer vi den, påvirker det ikke noget andre steder i programmet. Se Eksempel 2 nedenfor.

Globale variabler

Globale variabler er defineret udenfor funktionerne i programmet. De behøver ikke gives som argument til funktioner. Hvis man vil ændre på dem inde i en funktion, er det til gengæld nødvendigt at skrive **`global`** `x` i starten af funktionsdefinitionen, for at angive at den globale variabel `x` gerne må ændres.

Eksempel 1:

```
foo = 42
def minfunktion():
    global foo
    foo = 20

minfunktion()
# variablen "foo" er nu sat til 20
```

Her deklares det at funktionen `minfunktion` gerne må ændre i den globale variabel foo.

Eksempel 2:

```
foo = 42
def minfunktion():
    foo = 20

minfunktion()
# variablen "foo" er stadig 42
```

Her er der to variabler med samme navn `foo`. En global variabel der har værdien 42, og en lokal variabel der erklæres inde i funktionen, som tildeles værdien 20. Da der ikke er skrevet **`global`** `foo`, vil Python forstå det som om at en ny lokalvariabel skal oprettes og "overskygge" den globale.