Skjelpadder heile vegen ned

Skrevet av: Omsett frå Code Club UK (//codeclub.org.uk)

Oversatt av: Stein Olav Romslo

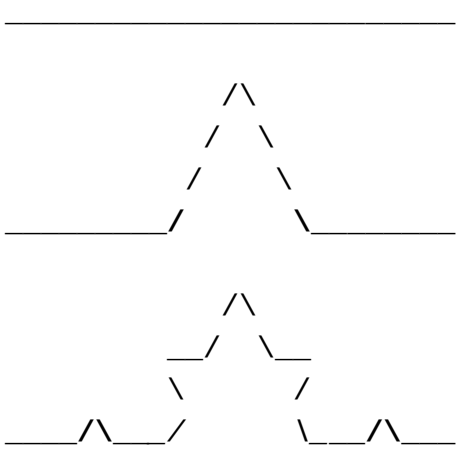
Kurs: Python
Tema: Tekstbasert
Fag: Matematikk, Programmering, Kunst og håndverk
Klassetrinn: 5.-7. klasse, 8.-10. klasse

Introduksjon

Hent fram skjelpaddene dine, åpne IDLE, det er på tide å teikne att.

Steg 1: Teikn eit fjell

Fyrst, la oss sjå på dei følgjande tre figurane. Korleis kan me teikne dei i Python?



Sjekkliste

- [ ] Å teikne den fyrste er lett. Som alltid er den fyrste linja `from turtle import *` så Python veit at me vil teikne. Dette programmet teiknar linja, sjekk at du får det til

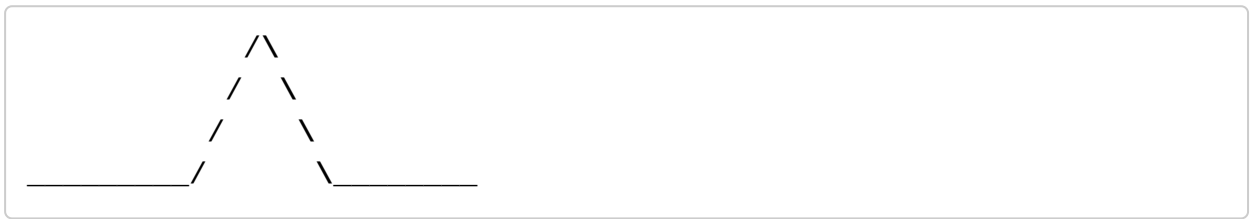
```
from turtle import *

def first():
    forward(30)

first()
```

Her har me sett det inn i ei prosedyre. Me kjem til å lage prosedyrer for alle formene, så la oss leggje dei til i fila me lagar.

- [ ] Legg til litt kode så koden din ser ut som dette:

```
from turtle import *

def first():
    forward(30)

def second():
    forward(30)
    left(60)
    forward(30)
    right(120)
    forward(30)
    left(60)
    forward(30)

second()
```

- [ ] Køyr koden din og sjå kva den gjer. Teiknar den riktig figur? Det skal sjå slik ut:



Steg 2: Eit fjell av fjell

Kva med den tredje figuren? Den er litt komplisert å programmere viss me må skrive alle stega me må ta. Men eigentleg er det berre det fyrste fjellet teikna fire gonger.

Du kan sjå at me teiknar den andre figuren (det enkle fjellet), så snur me og teiknar den att, snur, teiknar att, snur og teiknar ein siste gong.

Sjekkliste

- [ ] I staden for å skrive alle rørslene, la oss teikne den tredje figuren ved å kalle `second`

```
from turtle import *

def second():
    forward(30)
    left(60)
    forward(30)
    right(120)
    forward(30)
    left(60)
    forward(30)

def third():
    second()
    left(60)
    second()
    right(120)
    second()
    left(60)
    second()

third()
```

Funksjonen `third` ser veldig lik ut som `second`, men i staden for å kalle `forward`, kallar me `second`. Viss me ville kunne me ha skrive ein fjerde utgåve der me kallar `third` i staden. Dette høyrest fort ut som mykje å skrive, det må jo vere ein enklare måte å få datamaskina til å forstå dette på?

Desse figurane er spesielle på den måten at me teiknar dei ved å setje saman enklare versjonar av figurane om att og om att. Den tredje er laga av den andre, og den andre er laga av den fyrste. Det me verkeleg vil gjere er å be datamaskina teikne dette om att og om att heilt til det er ferdig.

Steg 3: Igjen og igjen

Me får det til ved å dele problemet i to: det enkle problemet og spesialtilfellet av problemet. Det enkle problemet er enkelt, det er berre `forward(100)`. Spesialtilfellet er litt vanskelegare. Det vil seie: teikn spesialtilfellet, men ein mindre enn i stad, heilt til du kjem til det enkle tilfellet. Det er kanskje enklare å sjå på programmet enn å forklare det.

Sjekkliste

- [ ] Lag ei ny fil med koden under:

```
from turtle import *

def mountain(depth):
    if depth == 1:
            forward(10)
    else:
        newdepth = depth -1
        mountain(newdepth)
        left(60)
        mountain(newdepth)
        right(120)
        mountain(newdepth)
        left(60)
        mountain(newdepth)

mountain(3)
```

Du kan sjå at me har brukt kode som er veldig lik `first`, `second` og `third`. Me brukar ei `if`-setning for å finne ut om me skal teikne det enkle tilfellet eller spesialtilfellet. I det spesielle tilfellet ber me om å teikne eit fjell, akkurat slik som `third` kalla `second`, men me ber den om å teikne ein enklare versjon kvar gong, med ein ny verdi for `depth`, ein mindre enn det me starta med.

- [ ] Køyr det og sjå kva som skjer. Kva skjer viss du prøver `mountain(1)`, `mountain(2)`, eller `mountain(4)`?

Steg 4: Teikn eit snøflak av fjell

Sjekkliste

- [ ] Me skal leggje til ein siste ting i fila frå i stad, slik at den ser ut som under. Me legg til ein ny prosedyre: `snowflake` :

```
from turtle import *

def mountain(depth, length):
    if depth == 1:
            forward(length)
    else:
        newdepth = depth -1
        mountain(newdepth, length)
        left(60)
        mountain(newdepth, length)
        right(120)
        mountain(newdepth, length)
        left(60)
        mountain(newdepth, length)

def snowflake(depth, length):
    mountain(depth, length)
    right(120)
    mountain(depth, length)
    right(120)
    mountain(depth, length)
    right(120)

snowflake(4,5)
```

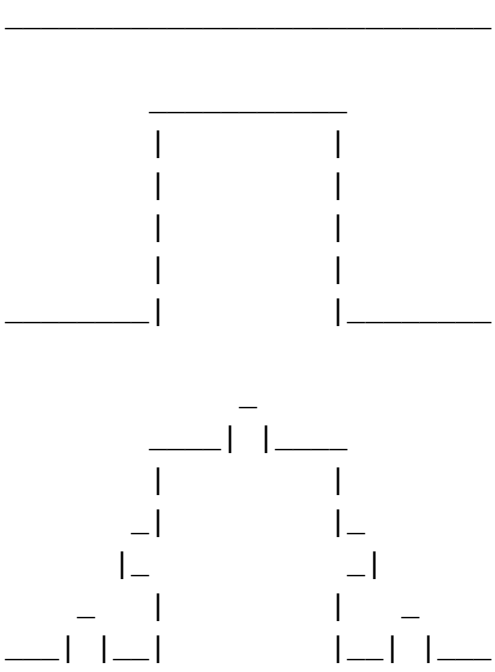
Dette biletet heiter *Kochs snøflak*. Viss du vil kan du prøve å endre på viknklane og sjå kva som skjer.

Dette kallast ein *fraktal*, fordi dei små bileta er laga av små versjonar av seg sjølv.

- [ ] Prøv å køyre `snowflake(1, 50)`, `snowflake(2, 25)`, `snowflake(3, 20)`. Di djupare du går, di lengre tid tek det å teikne, så hugs å setje `speed(11)` så skjelpadda går så fort den kan!

Steg 5: Boksar, fleire boksar, og endå fleire boksar

La oss sjå på ein annan figur, ein som liknar veldig på snøflaket, men med boksar i staden for fjell.



Akkurat som med fjellet er det eit enkelt tilfelle - ei rett linje - og eit spesialtilfelle - ei linje med ein firkanthump på. Me ser at den tredje er akkurat som før, teikna med figur nummer to nokre gonger.

Sjekkliste

- [ ] La oss åpne ei ny fil og prøve å teikne det andre biletet, som er det me skal gjenta:

```
from turtle import *

forward(30)
left(90)
forward(30)
right(90)
forward(30)
right(90)
forward(30)
left(90)
forward(30)
```

- [ ] Køyr det og sjekk at du får denne figuren:

Me har det enkle tilfellet `forward(100)`, og me veit korleis me skal teikne linja med firkanthumpen på seg, så la oss hoppe rett til teikninga!

Steg 6: Klumpete firkantar

Sjekkliste

- [ ] Åpne ei ny fil i IDLE og skriv inn det følgjande:

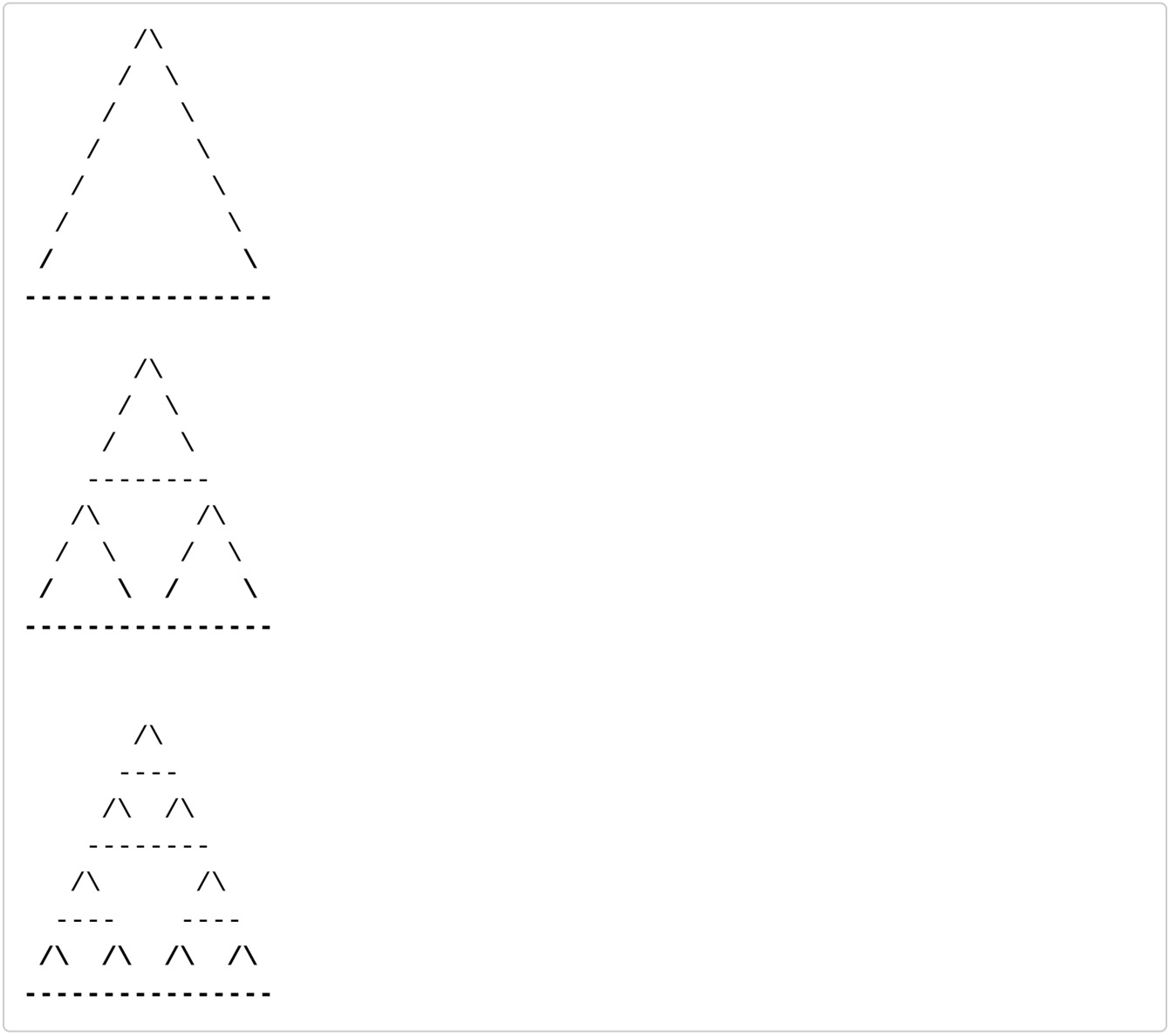
```
from turtle import *

def box(depth, length):
    if depth == 1:
            forward(length)
    else:
        newdepth = depth -1

        # Kva skal me skrive her?
        # Kopier inn koden frå steg 5 hit, men
        # bruk box(newdepth, length) i staden for forward(100)
        # Spør om hjelp viss du ikkje forstår heilt, men prøv
        # deg gjerne fram fyrst.

def xcurve(depth, length):
    box(depth, length)
    left(90)
    box(depth, length)
    left(90)
    box(depth, length)
    left(90)
    box(depth, length)
    left(90)

xcurve(4,5)
```

- [ ] Me har ikkje skrive inn spesialtilfellet, det har me tenkt å la deg gjere. Det er ganske likt som å programmere linja me hadde før, det burde halde å kalle `box(newdepth, length)` for å få det til. Koden burde sjå veldig lik ut som fjellet og snøflaket.

Steg 7: Trekantar att

La oss teikne ein siste fraktal, og som før har me eit enkelt tilfelle og eit spesialtilfelle.

Dei fyrste tre versjonane ser ut som dette. Me teiknar eit triangel, og så teiknar me det som tre trekantar saman.



Sjekkliste

- [ ] Lag ei ny fil og prøv det!

```
from turtle import *

def triforce(depth, length):

    if depth == 0:
        pendown()
        forward(length)
        left(120)
        forward(length)
        left(120)
        forward(length)
        left(120)
        penup()
    else:
        penup()
        newlength = length/2
        newdepth = depth - 1

        triforce(newdepth, newlength)
        forward(newlength)
        triforce(newdepth, newlength)

        left(120)
        forward(newlength)
        right(120)
        triforce(newdepth, newlength)

        right(120)
        forward(newlength)
        left(120)

speed(11)
penup()
setpos(-255,-255)
triforce(7, 512)
```

Du har kanskje lagt merke til at me brukar ein ny kommando `setpos` for å flytte skjelpadda til hjørnet av skjermen.

- [ ] Køyr det og sjå kva som skjer. Me kan sjå at det enkle tilfellet er berre å teikne ein trekant, og at spesialtilfellet er å teikne tre små trekantar.

- [ ] Prøv å endre verdiane me sender til `triforce()`, endre den siste linja til `triforce(5, 300)`, kva gjer den?

Steg 8: Bobler

Sjekkliste

- [ ] Viss du vil kan du teikne med sirklar i staden for trekantar! Åpne ei ny fil og skriv inn følgjande kode:

```
from turtle import *

def bubble(depth, length):
    if depth == 0:
        pendown()
        circle(length/2)
        penup()
    else:
        penup()
        newlength = length/2
        newdepth = depth - 1
        bubble(newdepth, newlength)
        forward(newlength)
        bubble(newdepth, newlength)
        left(120)
        forward(newlength)
        right(120)
        bubble(newdepth, newlength)
        right(120)
        forward(newlength)
        left(120)

speed(11)
penup()
setpos(-255,-255)
bubble(6, 512)
```

- [ ] Kva skjer? Kva ser det ut som? Me har brukt `circle` -kommandoen for å teikne ein sirkel på skjermen, som tek ein radius.

- [ ] Prøv å endre radiusen på sirkelen, bytt ut `circle(length/2)` med `circle(length)`, dette teiknar ein større sirkel i staden for.