Skilpadder hele veien ned

Erfaren Python

Introduksjon

Hent frem skilpaddene dine, åpne IDLE, det er på tide å tegne igjen.

Steg 1: Tegn et fjell

Men først, la oss se på de følgende tre figurene: Hvordan kan vi tegne dem i Python?

```
____________________

      /\
     / \
    /   \
_______/    \______

      /\
    __/ \__
    \    /
___/\__/    \__/\___
```

Sjekkliste

- [ ] Å tegne den første er lett. Som alltid er den første linjen `from turtle import *` så Python vet at vi vil tegne. Dette programmet tegner linjen, sjekk at du får det til

```
from turtle import *

def first():
    forward(30)

first()
```

Jeg har puttet det inn i en prosedyre fordi jeg liker prosedyrer. Vi kommer til å lage prosedyrer for alle formene, så la oss legge dem til i filen vi lager.

- [ ] Legg til litt kode så koden din blir slik som denne:

```
from turtle import *

def first():
    forward(30)

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

second()
```

- [ ] Kjør koden din og se hva den gjør, tegner den riktig figur? Det skal se slik ut

```
      /\
     / \
    /  \
_______/   \______
```

Steg 2: Et fjell av fjell

Hva med den tredje figuren? Selv om den er litt komplisert å programmere om vi må skrive alle stegene vi må ta, er det egentlig bare det første fjellet tegnet fire ganger.

```
      /\
    __/ \__
    \    /
____/\___/    \___/\___
```

Du kan se at vi tegner den andre figuren (det enkle fjellet), deretter snur vi, tegner den igjen, snur, tegner igjen, snur og tegner den en siste gang.

Sjekkliste

- [ ] Istedenfor å skrive alle bevegelsene, la oss tegne 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()
```

`third` ser veldig lik ut som `second`, men istedenfor å kalle `forward`, kaller vi `second`. Hvis vi ville, kunne vi til og med skrive en fjerde utgave og kalle `third` istedenfor. Dette høres fort ut som mye å skrive, det må da være en måte vi kan få datamaskinen til å forstå dette på?

Disse figurene er spesielle på den måten at vi tegner dem ved å sette sammen enklere versjon av figurene om igjen og om igjen: Den tredje er laget av den andre, og den andre er laget av den første. Det vi virkelig har lyst til er å be datamaskinen om å tegne dette igjen og igjen helt til det er ferdig.

Steg 3: Igjen og igjen

Vi får til dette ved å dele problemet i to: Det enkle problemet og spesialtilfellet av problemet. Det enkle problemet er enkelt: Det er bare `forward(100)`. Spesialtilfellet er litt vanskeligere, det vil si: Tegn spesialtilfellet, men en mindre enn i sted, helt til du kommer til det enkle tilfellet. Det er kanskje enklere å se på programmet enn å forklare.

Sjekkliste

- [ ] Lag en 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 se at vi har brukt kode som er veldig likt `first`, `second` og `third`. Vi bruker en `if`-setning for å finne ut om vi skal tegne det enkle tilfellet eller spesialtilfellet. I det spesielle tilfellet ber vi om å tegne et fjell, akkurat slik som `third` kalte `second`, men vi ber den om å tegne en enklere hver gang, med en ny verdi for `depth`, en mindre enn det vi startet med.

- [ ] Kjør det og se hva som skjer. Hva skjer om du prøver `mountain(1)`, `mountain(2)`, eller `mountain(4)`?

Steg 4: Tegn et snøflak av fjell

Sjekkliste

- [ ] La oss bare legge til en siste ting til filen fra i sted, så den ser ut som under. Vi legger til en 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)
```

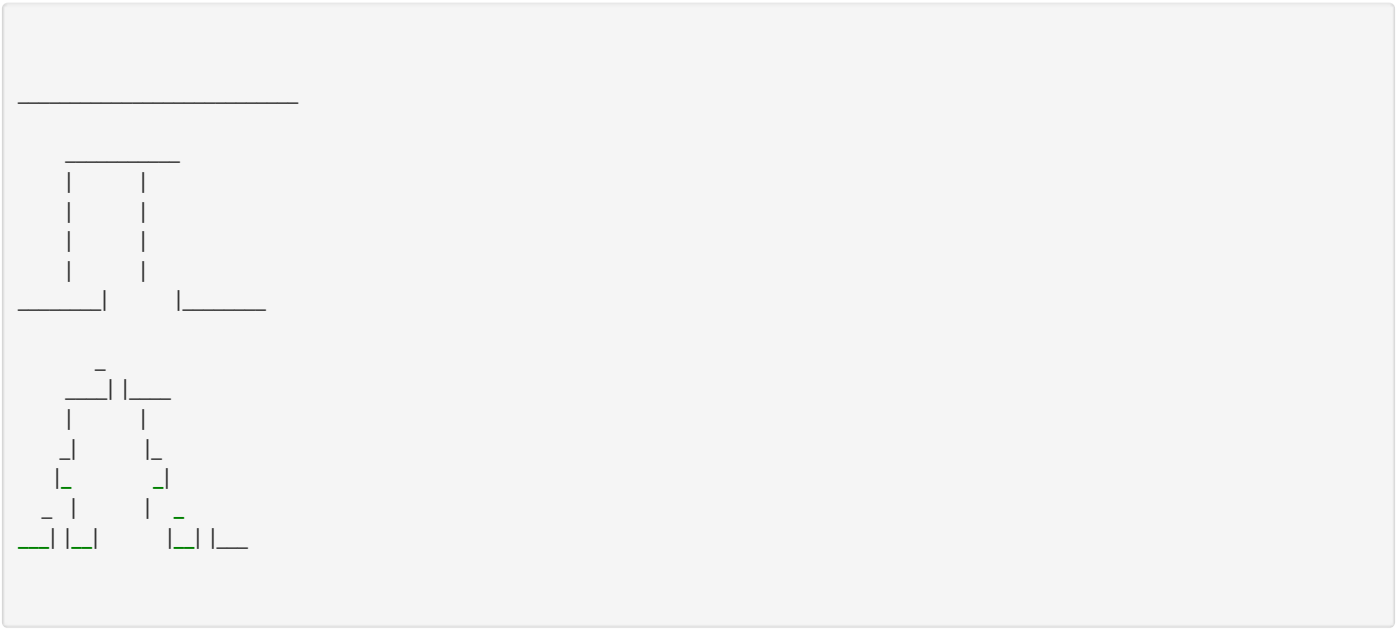
Dette bildet heter Kochs snøflak. Hvis du vil kan du forsøke å endre på vinklene og se hva som skjer.

Dette heter et fraktal, fordi de små bildene er laget av små versjoner av seg selv.

- [ ] Prøv å kjøre `snowflake(1, 50)`, `snowflake(2, 25)`, `snowflake(3, 20)`. Jo større dybde, jo lenger tar det å tegne, så husk å putte inn `speed(11)` så skilpadden går så fort den kan!

Steg 5: Bokser, flere bokser, og enda flere bokser

La oss se på en annen figur, en som ligner veldig på snøflaket, men med bokser istedenfor fjell.



Akkurat som med fjellet er det et enkelt tilfelle: en rett linje, og et spesialtilfelle: Tegn en linje med en firkanthump på. Vi ser også at den tredje er akkurat som før, tegn den andre figuren noen ganger.

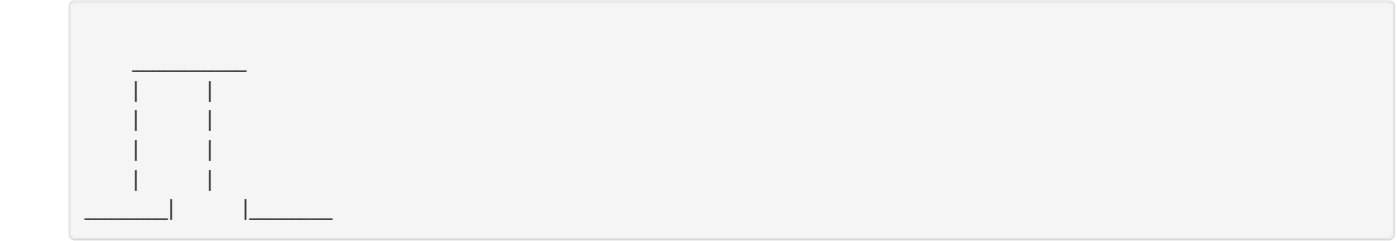
Sjekkliste

- [ ] La oss åpne en ny fil og forsøke å tegne det andre bildet, som er det vi skal gjenta:

```
from turtle import *

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

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



Vi har det enkle tilellet `forward(100)`, og vi vet hvordan vi skal tegne linjen med firkanthumpen på seg, så la oss hoppe rett til tegningen!

Steg 6: Klumpete firkanter

Sjekkliste

- [ ] Åpne en ny fil i IDLE og skriv inn følgende:

```
from turtle import *

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

      # Hva skal vi skrive her?
      # Kopier inn koden fra steg 5 hit, men
      # bruk box(newdepth, length) istedenfor forward(100)
      # Spør om hjelp om du ikke forstår helt, men prøv
      # deg gjerne frem først.

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)
```

- [ ] Vi har ikke skrevet inn spesialtilfellet, det har vi tenkt å la deg gjøre. Det er ganske likt som å programmere linjen vi hadde før, det burde holde å kalle `box(newdepth, length)` for å få det til. Koden burde se veldig lik ut som fjellet og snøflaket.

Steg 7: Trekanter igjen

La oss tegne et siste fraktal, og som før har vi et enkelt tilfelle og et spesialtilfelle.

De første tre versjonene ser ut som dette. Vi tegner et triangel, og så tegner vi det som tre trekanter sammen.

```
    /\
   /  \
  /    \
 /      \
/        \
----------

    /\
   /  \
  /    \
 --------
/\    /\
/  \  /  \
/    \/    \
----------


    /\
   ----
  /\  /\
 --------
/\    /\
----  ----
/\ /\ /\ /\
----------
```

Sjekkliste

- [ ] Lag en 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 vi bruker en ny kommando `setpos` for å flytte skilpadden til hjørnet av skjermen.

- [ ] Kjør det og se hva som skjer. Vi kan se at det enkle tilfellet er bare å tegne en trekant, og at spesialtilfellet er å tegne tre små trekanter.

- [ ] Prøv å endre verdiene vi sender til `triforce()`, endre den siste linjen til `triforce(5, 300)`, hva gjør den?

Steg 8: Bobler

Sjekkliste

- [ ] Hvis du vil kan du tegne med sirkler istedenfor trekanter! Åpne en ny fil og skriv inn følgende 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)
```

- [ ] Hva skjer? Hva ser det ut som? Vi har brukt `circle` kommandoen for å tegne en sirkel på skjermen, som tar en radius.

- [ ] Forsøk å endre sirkelens radius, bytt ut `circle(length/2)` med `circle(length)`, dette tegner en større sirkel istedenfor.

Lisens: Code Club World Limited Terms of Service **Forfatter:** Oversatt fra Code Club UK
Oversetter: Bjørn Einar Bjartnes