Skilpaddeskolen

Steg 1: Flere firkanter

Sjekkliste

- [ ] Åpne IDLE-editoren, og åpne en ny fil ved å trykke `File > New Window`, og la oss begynne.

  Husk at du skal ha to vinduer åpne. Det ene er 'Python Shell' og det andre er for å skrive kode i.

  Som sist vil den første linjen alltid være `from turtle import *` for å fortelle Python at vi vil tegne ved hjelp av skilpadde-biblioteket!

  ```
  from turtle import *

  for n in range(4):
      forward(100)
      right(90)
  ```

- [ ] Lagre det som en ny fil, og kjør programmet fra menyen ved å trykke `Run > Run Module`.

  Husk at `for n in range(4)` gjentar koden, og at koden må grupperes med mellomrom (innrykk) for å være en del av for-løkken. Bruk 'Tab' (knappen over Caps-Lock) for å flytte kode.

Steg 2: Forskjellige firkanter

La oss bruke variabler for å gjøre programmet vårt lettere å lese og lettere å endre, akkurat som vi gjorde i forrige modul.

Sjekkliste

- [ ] Endre programmet så det ser slik ut:

  ```
  from turtle import *

  sides = 4
  length = 100
  angle = 360/sides

  for n in range(sides):
      forward(length)
      right(angle)
  ```

- [ ] Kjør det ved å trykke `Run > Run Module` fra menyen. Får du den samme firkanten som før? Sjekk at det virker før du går videre.

Dette er et litt langt program, men nå kan vi endre det til å tegne hvilken figur vi vil. Problemet er bare at vi er nødt til å klippe og lime programmet for å få det til. Som tidligere kan vi skrive kode for å slippe å gjenta oss selv (programmerere anstrenger seg gjerne litt slik at de kan være late etterpå!). Denne gangen vil vi definere en ny kommando.

Steg 3: En ny kommando dukker opp

Sjekkliste

- [ ] Vi endrer koden og legger til `def poly():`. Pass på at koden har riktig innrykk (du kan merke den og trykke Tab) og bruk den nye prosedyren. (Å bruke en prosedyre heter å kalle prosedyren).

```
from turtle import *

def poly():
    sides = 4
    length = 100
    angle = 360/sides

    for n in range(sides):
        forward(length)
        right(angle)

pencolor('red')
poly()
right(180)
poly()
```

- [ ] Kjør programmet. Hvis det virker skal to røde firkanter bli tegnet.

Vi sparte litt tid ved å definere en ny prosedyre i Python, og nå kan vi tegne en rød firkant to ganger, uten å skrive hele greia to ganger. Disse nye kommandoene heter prosedyrer i Python, og de er fine for å slippe å skrive så mye.

Steg 4: Hvorfor stoppe med firkanter

Vi er ikke ferdige ennå - hva med å endre prosedyren så den kan tegne hvilken som helst form? Som med `forward` og `right`, kan vi sende verdier inn i prosedyren istedenfor å endre koden hver gang.

Sjekkliste

- [ ] Endre koden så den ser slik ut:

```
from turtle import *

def poly(sides, length):
    angle = 360/sides

    for n in range(sides):
        forward(length)
        right(angle)

pencolor('red')
poly(4, 100)
right(180)
pencolor('blue')
poly(3, 150)
```

- [ ] Kjør den og se hva som skjer.

  La oss ta dette litt sakte, for dette er ganske kule greier. Istedenfor å bestemme variablene i prosedyren, sier vi at prosedyren tar noen verdier som har navn, og så bruker vi verdiene der vi trenger dem.

  Vi flyttet noen verdier ut av prosedyren, og flyttet dem til den delen av koden som bruker dem. Nå kan vi, med en eneste prosedyre, tegne *hvilken som helst* form, med *hvilken som helst farge*. Jeg vet ikke hva du tenker, men dette imponerer meg hver gang jeg tenker på det: Vi kan lære datamaskinen nye triks, og så få den til å gjøre triksene.

Å være i stand til å definere nye prosedyrer som kan oppføre seg forskjellig basert på verdiene vi gir inn er et av de kraftigste verktøyene i programmering.

Tips

Vi skiller mellom prosedyrer og funksjoner når vi snakker om dem, men de er ser helt like ut i Python. Det kommer funksjoner i neste kapittel, men vi nevner det allerede nå så du har hørt om det. Forskjellen er at en prosedyre kan

gjøre noe med omverdenen, for eksempel tegne noe eller skrive noe på skjermen. Både prosedyrer og funksjoner kan returnere en verdi også, det kommer vi tilbake til, men prosedyren kan finne på å gi forskjellig resultat selv om du gir inn samme verdier. For eksempel kan en prosedyre `erDet("fredag")` svare ja på fredager og nei på lørdager. En funksjon derimot må alltid svare det samme hvis den får de samme verdiene.

Steg 5: Skilpaddestreker

Sjekkliste

- [ ] Selv om skilpadden er en liten robot som kan tegne, kan den også flytte seg uten å tegne. Husk at vi kan bruke `penup()` og `pendown()` for å slå av og på at skilpadden setter spor. Åpne en ny Python-fil, og skriv inn koden under:

```
from turtle import *

length = 200
for num in range(8):
    forward(length/16)
    penup()
    forward(length/16)
    pendown()
```

- [ ] Dette programmet tegner en stiplet linje over skjermen. Kjør det og sjekk!

Steg 6: Tegne figurer

Vi kan koble figur-programmet og stiplet-linje-programmet sammen ved å bytte ut kommandoen `forward` med koden vi har for stiplete linjer. Vi bruker koden for å tegne figurer ytterst, og inni der bruker vi koden for å lage stiplete linjer istedenfor hele streker.

Sjekkliste

- [ ] Endre koden så den ser ut som følgende:

```
from turtle import *
speed(11)
shape("turtle")

def dashpoly(sides, length):
    angle = 360/sides

    for n in range(sides):
        for num in range(8):
            forward(length/16)
            penup()
            forward(length/16)
            pendown()
        right(angle)

pencolor('red')
dashpoly(4, 100)
right(180)
pencolor('blue')
dashpoly(3, 150)
```

- [ ] Kjør koden og se hva den gjør.

  Vi har to for-løkker inni hverandre, en ytre og en indre. Den ytre løkken `for n in range(sides)` tegner hver kant av figuren, og hver gang kjører den indre løkken `for num in range(8)` som tegner stiplete linjer.

Den ytre løkken bruker variabelen `n` for å holde styr på hvor mange ganger den har gjentatt seg selv, og den indre løkken bruker variabelen `num` på tilsvarende måte. Du må bruke forskjellige variabelnavn inni løkken, ellers roter du det bare til.

Steg 7: Byggeklosser for figurer

Sjekkliste

- [ ] La oss bruke prosedyrer igjen for å rydde opp i koden. Endre koden fra steg 6 og la oss dele koden i biter.

```
from turtle import *
speed(11)
shape("turtle")

def dashforward(length):
    for num in range(8):
        forward(length/16)
        penup()
        forward(length/16)
        pendown()

def dashpoly(sides, length):
    angle = 360/sides

    for n in range(sides):
        dashforward(length)
        right(angle)

pencolor('red')
dashpoly(4, 100)
right(180)
pencolor('blue')
dashpoly(3, 150)
```

- [ ] Kjør koden og se at den gjør akkurat det samme som før.

Tips

Trikset er at istedenfor å bygge programmer ved å klippe og lime, kan vi definere nye kommandoer og gjenbruke dem. Da blir koden kortere og litt lettere å forstå.

Steg 8: Litt tilfeldigheter

Hva om vi gjør litt tilfeldige sprell rett før vi er ferdige? Vi kan be datamaskinen velge et tall for oss, eller velge en farge for oss, litt som om vi kaster terning. Scratch kan dette også, da brukte vi `pick`.

Sjekkliste

- [ ] I en ny fil, skriv inn følgende:

```
from turtle import *
from random import randrange, choice
colors = ['red', 'blue', 'green']


def poly(sides, length):
    angle = 360/sides

    for n in range(sides):
        forward(length)
        right(angle)

for count in range(10):
    pencolor(choice(colors))
    right(randrange(0,360))
    poly(randrange(3,9), randrange(10,30))
```

- [ ] Lagre og kjør koden

Programmet skal tegne ti figurer i forskjellige farger med forskjellig størrelse. Linjen `from random import randrange, random` gir oss to nye prosedyrer, `randrange()` og `choice()`.

`randrange()` plukker ut et tall mellom det laveste og det høyeste tallet vi gir inn, så `randrange(1, 10)` velger et tall mellom 1 og 9 (Python begynner med 1, og stopper rett før 10).

`choice()` velger en ting fra listen vi gir inn. En liste er en samling av verdier, for eksempel `[1, 2, 3]`. I koden ovenfor bruker vi listen `colors`, som har verdiene `'red'`, `'blue'`, and `'green'`.

Ved å bruke `choice()` og `randrange()` kan vi be datamaskinen om å velge farge, størrelse og form for oss, og det kommer til å bli forskjellig resultat nesten hver eneste gang du kjører programmet.

Flere ting å prøve

Hva med å prøve flere farger, eller å endre tallene? Hva skjer?