Python

Skjelpaddeskulen

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

Oversatt av: Stein Olav Romslo

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

Steg 1: Fleire firkantar

Sjekkliste

- [ ] Åpne IDLE-editoren, og åpne ei ny fil ved å trykkje `File > New File` , og la oss setje i gang.

  Hugs at du skal ha to vindauge oppe. Det eine er 'Python Shell' og det andre er for å skrive kode i.

  Som sist vil den fyrste linja alltid vere `from turtle import *` for å fortelje Python at me vil teikne ved hjelp av skjelpadde-biblioteket!

  ```
  from turtle import *

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

- [ ] Lagre det som ei ny fil, og køyr programmet frå menyen ved å trykkje `Run

  > Run Module`.

  Hugs at `for n in range(4)` gjentek koden, og at koden må grupperast med mellomrom (innrykk) for å vere en del av for-løkka. Bruk *tab* (knappen rett over *caps lock*) for å få innrykk i koden.

Steg 2: Ulike firkantar

La oss bruke variablar for å gjere programmet vårt lettare å lese og lettare å endre, akkurat som me gjorde i førre oppgåve.

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)
```

- [ ] Køyr det ved å trykkje `Run > Run Module` frå menyen. Får du den same firkanten som før? Sjekk at det virkar før du går vidare.

Dette er eit litt langt program, men no kan me endre det til å teikne den figuren me vil. Problemet er berre at me må klyppe og lime programmet for å få det til. Som før kan me skrive kode for å sleppe å gjenta oss sjølv (programmerarar anstrenger seg gjerne litt slik at dei kan vere late etterpå!). Denne gongen skal me lage ein *funksjon*. Ein funksjon er ein enkel måte å gjenbruke ei kodeblokk (eller oppskrift om du vil) mange gonger. Funksjonen får eit namn, og me kan hente den fram att ved å bruke namnet seinare.

Steg 3: Me lagar ein funksjon

Sjekkliste

- [ ] Me endrar koden og legg til `def poly():`. `def` tyder definer, altså å lage. Pass på at koden har riktig innrykk og bruk den nye funksjonen. For å få innrykk på

fleire linjer kan du merke linjene og så trykkje *tab* (knappen rett over *caps lock*). Viss du vil ha mindre innrykk, bruk *shift* + *tab*.

```
from turtle import *

def poly(): # me lagar funksjonen
    sides = 4
    length = 100
    angle = 360/sides

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

pencolor('red')
poly() # me kallar på funksjonen
right(180)
poly()
```

- [ ] Køyr programmet. Viss det virkar skal det bli teikna to raude firkantar.

Me sparte litt tid ved å lage ein ny funksjon i Python, og no kan me teikne ein raud firkant to gonger, utan å skrive heile greia to gonger. Den nye funksjonen `poly()` er fin for å sleppe å skrive så mykje.

Steg 4: Kvifor stoppe med firkantar?

Me er ikkje ferdige endå. Kva med å endre funksjonen så den kan teikne kva form som helst? Som med `forward` og `right` kan me sende verdiar inn i funksjonen i staden for å endre koden kvar gong.

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)
```

- [ ] Køyr den og sjå kva som skjer.

  La oss ta dette litt sakte, for dette er ganske kule greier. I staden for å bestemme variablane i funksjonen seier me at funksjonen tek nokre verdiar som har namn, og så brukar me verdiane der me treng dei.

  Me flytta nokre verdiar ut av funksjonen, og flytta dei til den delen av koden som brukar dei. No kan me, med ein eineste funksjon, teikne *kva form som helst*, med *kva farge som helst*. Dette er veldig imponerande: me kan lære datamaskina nye triks, og så få den til å gjere triksa!

Å vere i stand til å lage nye funksjonar som kan oppføre seg forskjellig basert på verdiane me gir inn er eitt av dei kraftigaste verktøya i programmering.

Tips

I Python fist det funksjonar, og det finst prosedyrer. Desse omgrepa går litt inn i kvarandre, så det er ikkje så farleg om du ikkje ser skilnaden mellom dei. Ein funksjon returnerer noko, og skal helst ikkje gjere noko anna enn å rekne ut returverdien. I tillegg bør ein funksjon alltid returnere det same når den får same input. Prosedyrar liknar veldig på funksjonar, men dei får lov til å returnere ulike ting avhenging av andre ting enn input. I tillegg kan ei prosedyre gjere noko anna enn å returnere noko. Til dømes kan ei prosedyre teikne på skjermen. I Python gjer me ikkje forskjell på funksjonar og prosedyrer, så det er vanleg å berre kalle alle for funksjonar.

Steg 5: Skjelpaddestrekar

Sjekkliste

- [ ] Sjølv om skjelpadda er ein liten robot som kan teikne, kan den òg flytte seg utan å teikne. Hugs at me kan bruke `penup()` og `pendown()` for å slå av og på at skjelpadda set spor. Åpne ei 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 teiknar ei stipla linje over skjermen. Køyr det og sjekk!

Steg 6: Teikne figurar

Me kan kople figur-programmet og stipla linje-programmet saman ved å byte ut funksjonen `forward` med koden me har for stipla linjer. Me brukar koden for å teikne figurar ytst, og inni der brukar me koden for å lage stipla linjer i staden for heile strekar.

Sjekkliste

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

```
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)
```

- [ ] Køyr koden og sjå kva den gjer.

Me har to for-løkker inni kvarandre, ei ytre og ei indre. Den ytre løkka `for n in range(sides)` teiknar kvar kant av figuren, og kvar gong køyrer den indre løkka `for num in range(8)` som teiknar stipla linjer.

Den ytre løkka brukar variabelen `n` for å halde styr på kor mange gonger den har gjenteke seg sjølv, og den indre løkka brukar variabelen `num` på tilsvarande måte. Du må bruke ulike variabelnamn inne i løkka, elles rotar du det berre til.

Steg 7: Byggjeklossar for figurar

Sjekkliste

- [ ] La oss bruke funksjonar att for å rydde opp i koden. Endre koden frå steg 6 og la oss dele koden i bitar.

```
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)
```

- [ ] Køyr koden og sjå at den gjer akkurat det same som før.

Tips

Trikset er at i staden for å byggje program ved å klyppe og lime, kan me lage nye funksjonar og bruke dei att. Då blir koden kortare og litt lettare å forstå.

Steg 8: Litt tilfeldigheiter

Kva viss me gjer litt tilfeldige sprell rett før me er ferdige? Me kan be datamaskina velje eit tal eller ei farge for oss, litt som om me kastar terning. Scratch kan òg gjere det, då brukte me `pick`.

Sjekkliste

Sjekkliste

- [ ] I ei ny fil, skriv inn følgjande:

```
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 køyr koden

Programmet skal teikne ti figurar i ulike fargar med ulik storleik. Linja `from random import randrange, random` hentar inn funksjonane `randrange()` og `choice()`.

Prosedyra `randrange()` plukkar ut eit tall mellom det lågaste og det høgaste talet me gir inn, så `randrange(1, 10)` vel eit tall mellom 1 og 9 (Python startar med 1, og stoppar rett før 10).

Prosedyra `choice()` vel ein ting frå lista me gir inn. Ei liste er ei samling av verdiar, til dømes `[1, 2, 3]`. I koden over brukar me lista `colors`, som har verdiane `'red'`, `'blue'`, og `'green'`.

Ved å bruke `choice()` og `randrange()` kan me be datamaskina om å velje farge, storleik og form for oss, og det kjem til å bli forskjellig resultat nesten kvar einaste gong du køyrer programmet.

Fleire ting å prøve

Kva med å prøve fleire farger, eller å endre tala? Kva skjer?

Lisens: Code Club World Limited Terms of Service
(https://github.com/CodeClub/scratch-curriculum/blob/master/LICENSE.md)