Python

Skjelpadder

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:

Her skal me lære eit programmeringsspråk som heiter Python. Personen som laga det kalla det opp etter favorittprogrammet sitt på TV: Monthy Python sitt Flygande Cirkus. Python blir brukt av mange programmerarar til mange ulike ting. Python blir brukt av YouTube, NASA, CERN og mange andre. Viss kodeklubben din har ein Raspberry Pi kan du bruke Python til å programmere den. Mange likar Python fordi det er enkelt å lese (i motsetnad til språk dei synest er vanskelege å lese). Meiningane er mange, og etter kvart som du lærer fleire språk får du nok sterke kjensler for fleire av dei. Å vere i stand til å lese kode er viktig for ein programmerar, kanskje like viktig som å kunne skrive den.

Steg 1: Hei, Skjelpadde!

No skal me ha det moro med skjelpadder. Ei skjelpadde er ein liten robot som teiknar seg sjølv på skjermen din. Me kan få den til å bevege seg rundt med Python-kommandoar.

Sjekkliste

- [ ] Åpne eit nytt kodevindauge og skriv dette:

```
from turtle import *

forward(100)
```

- [ ] Lagre programmet ditt som myturtle.py og vel `Run -> Run Module`. Ser du korleis skjelpadda bevega seg 100 punkt framover på skjermen? Skjelpadda har

ein penn festa til seg, så den teiknar linjer når den beveger seg rundt.

Tips

Python-filer skal alltid ha filnamn som sluttar med `.py`.

Sjekkliste

- [ ] La oss få skjelpadda til å bevege seg rundt på skjermen! Prøv å bruke `backward(distance)` i tillegg til å snu den ved å bruke `right(angle)` og `left(angle)`. Instruksjonen `backward(20)` fortel til dømes skjelpadda at den skal bevege seg bakover 20 pikslar, og `right(90)` fortel at den skal snu seg 90 gradar til høgre. Du kan gi den meir enn ein instruksjon i gongen, og dei blir utført i rekkefølgje.

```
from turtle import *

speed(11)
shape("turtle")

forward(100)
right(120)
forward(100)
left(90)
backward(100)
left(90)
forward(50)
```

Vinklar og gradar

Leik deg litt med å lage dine eigne figurar ved å bruke `forward`, `backward`, `left`, `right`. Hugs at `forward` og `backward` brukar pikslar, medan `left` og `right` brukar gradar. La oss undersøke ei skjelpadde som beveger seg til høgre.

```
        Nord
         0
         |
Vest     |     Aust
270 -----|----- 90
         |
         |
        180
        Sør
```

Når skjelpadda ser nordover og du ber den snu seg 90 gradar til høgre, ser den austover. Snur du den 180 gradar frå nord ser den sørover, og snur du den 270 gradar frå nord ser den vestover. Snur du 360 gradar stoppar den der den starta. Kanskje er det lettere å tenke på dette som snowboard-triks?

Kva med å snu mot venstre?

```
        Nord
          0
          |
Vest      |      Aust
  90 -----|----- 270
          |
          |
         180
         Sør
```

Når skjelpadda ser nordover og du ber den snu seg 90 gradar mot venstre, ser den vestover. Når skjelpadda ser nordover og du ber den snu seg 180 gradar mot venstre ser den sørover, og om den ser nordover og du ber den snu seg 270 gradar mot venstre ser den austover. Snur du 360 gradar er du attende der du starta, 360 gradar er alltid heilt rundt.

Kva gjer koden på starten av programmet vårt?

- Koden `from turtle import *` fortel Python at me vil bruke skjelpadde-biblioteket (turtle), ei samling av kode me kan bruke for å teikne på skjermen. Å bruke eit ferdig bibliotek gjer at me kan spare tid og gjenbruke andre sitt arbeid.

- Funksjonen `speed()` bestemmer farta til skjelpadda. Me må gi inn ein verdi mellom 1 og 11, der 11 er det raskaste og 1 er det treigaste.

- Funksjonen `shape()` set forma på teiknefiguren vår. Me bruker forma (shape) `"turtle"` (skjelpadde), men me kan òg seie at me vil at figuren skal sjå ut som `"arrow"` (pil), `"circle"` (sirkel), `"square"` (kvadrat), `"triangle"` (trekant) eller `"classic"` (klassisk).

Me kjem til å bruke desse instruksjonane på toppen av alle programma våre i denne oppgåva. Viss du vil kan du prøve å gi skjelpadda ei av dei andre formene, til dømes pil, og få den til å gå så fort eller sakte som du vil.

Steg 2: Teikne figurar!

La oss lage eit kvadrat ved å fortelje skjelpadda korleis den skal bevege seg rundt.

Sjekkliste

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

```
from turtle import *

speed(11)
shape("turtle")

forward(100)
right(90)
forward(100)
right(90)
forward(100)
right(90)
forward(100)
right(90)
```

Lagre programmet ditt og vel `Run -> Run Module`. Ser du ein firkant? Skjelpadda snur seg 90 gradar fire gonger, og ender opp med å sjå den same retninga som den starta. Å snu 90, 90, 90 og så 90 gradar igjen snur skjelpadda totalt 360 gradar.

Kva med ein trekant? Ein trekant har tre hjørne, så me må snu tre gonger. Viss me vil ende opp i same retning må me snu 360 gradar, akkurat som med firkanten.

Difor snur me 120 gradar tre gonger.

- [ ] Endre koden din til å sjå ut som koden under for å få den til å teikne ein trekant:

```
from turtle import *

speed(11)
shape("turtle")

forward(100)
right(120)
forward(100)
right(120)
forward(100)
right(120)
```

- [ ] Køyr koden. Ser du ein trekant?

Vel ei farge

Kva er yndlingsfarga di? Du kan endra farga på linjene ved å bruke funksjonen `pencolor`. Du har kanskje lært at farge på engelsk skrivast "colour" med u. Det er britisk engelsk, men Python brukar amerikansk engelsk, og amerikanarane stavar det "color" utan u. Du kan òg endre storleiken på pennen ved å bruke `pensize`:

Sjekkliste

- [ ] Endre koden frå dømet over til å sjå ut som det neste dømet ved å leggje til desse nye kommandoane:

```
from turtle import *

speed(11)
shape("turtle")

pensize(10)
pencolor("red")
forward(100)
right(120)
pencolor("blue")
forward(100)
right(120)
pencolor("green")
forward(100)
right(120)
```

- [ ] Køyr koden din. Kva teiknar den på skjermen? Denne koden teiknar ein tjukk trekant i tre ulike farger.

- [ ] Prøv å endre fargene i koden din, køyr den og sjå kva som skjer. Skjelpadda kan mange hundre ulike fargar, ikkje berre blå, raud og grøn. Prøv med yndlingsfarga di! Du kan òg bruke fargar i **hex**, som du kanskje har gjort med CSS før. I staden for å bruke `pencolor("red")` kan du bruke hex `pencolor("#FF0000")`. Kva farge er `#FF4F00`?

Steg 3: Gjenta deg sjølv (med ei for-løkke)

Det siste programmet var dei same kommandoane igjen og igjen. I staden for å skrive dei ned kan me be datamaskina om å gjenta dei for oss. Du har vore borti *iterasjon* i Scratch ved å bruke `For alltid` - og `Gjenta` / `Gjenta til` -blokker. I Python brukar me **for-løkker** når me har kode du vil gjenta `n` gonger. I dette dømet skal me gjenta koden (som er rykka inn) 4 gonger fordi ein firkant har 4 sider.

Sjekkliste

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

```
from turtle import *

speed(11)
shape("turtle")

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

- [ ] Lagre programmet og vel: `Run -> Run module`.

  Legg merke til at koden er skyve inn, *indentert*, eller dytta til høgre under for-løkka. Python brukar mellomrom for å vite kva kommandoar som skal bli gjenteke. Du kan bruke tabulatortasten for å få IDLE til å *indentere*, eller bruke `shift + tab` til å ta det bort.

- [ ] La oss sjå kva som skjer viss me berre indenterer (skyv inn) `forward`. Gjer om programmet ditt så det ser ut som dette:

```
from turtle import *

speed(11)
shape("turtle")

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

- [ ] Legg merke til at `forward` er indentert og `right` ikkje er det. Kva trur du dette programmet gjer? Prøv å køyre det og finn det ut.

  Fekk du ei rett linje? Python vil gjenta `forward` fire gonger, og så snu til høgre. Python brukar mellomrom for å gruppere kommandoar saman, akkurat som Scratch brukar blokker. Python klagar til deg viss du ikkje har fått mellomromma riktig.

- [ ] La oss endre programmet tilbake slik at det lagar ein firkant att, men i staden for å bruke tal i koden skal me gi tala namn. Det gjer det lettare å sjå kva programmet gjer, og gjer at me slepp å gjenta oss sjølv.

  Endre fila så den ser slik ut:

```
from turtle import *

speed(11)
shape("turtle")

sides = 4
length = 100
angle = 90
for count in range(sides):
    forward(length)
    right(angle)
```

- [ ] Lagre programmet og vel: `Run -> Run module`.

Oppgåve: Teikn dei andre formene

Kan du teikne nokre av figurane under berre ved å endre verdiane?

- [ ] Ein trekant? (tre sider)
- [ ] Eit pentagram? (fem sider)
- [ ] Eit heksagram? (seks sider)
- [ ] Eit oktagram? (åtte sider)

Hugs at ein trekant har tre sider, så me snur 120 grader i kvart av dei tre hjørna for at det skal bli 360 gradar til saman. For ein firkant må me snu 90 gradar 4 gonger, som òg blir 360 gradar.

Viss du snur seks gonger, kor mange gonger må du snu for at det skal bli 360 gradar? Prøv med ulike tal og sjå kva som skjer.

Steg 4: Snu, snu, snu

I staden for å rekne ut vinklane sjølv kan me få datamaskina til å gjere det for oss. Python let deg leggje til, trekke frå, gange og dele. Me kan skrive `sides = 4 + 1` i staden for 5, eller `sides = 4 - 1` i staden for 3. For multiplikasjon brukar Python `*`, og for divisjon skriv me `/`. Viss me må snu 360 gradar til saman kan me rekne ut vinkelen me treng. For ein firkant får me `360 / 4` som er lik 90, for trekanten får me `360 / 3` som er lik 120.

Sjekkliste

- [ ] Endre programmet ditt til å rekne ut vinkelen.

```
from turtle import *

speed(11)
shape("turtle")

sides = 4
length = 20

angle = 360/sides
for count in range(sides):
    forward(length)
    right(angle)
```

- [ ] No kan du endre talet på sider. Klarar Python å gjere jobben rett? Prøv med så mange kantar du vil!

Steg 5: Fylte figurar

Sjekkliste

- [ ] Me kan be skjelpadda om å fylle figurane med ei farge ved å bruke `begin_fill()` og `end_fill()`. Endre koden din til å bruke desse kommandoane:

```
from turtle import *

speed(11)
shape("turtle")

sides = 4
length = 20

fillcolor('red')
pencolor('red')
begin_fill()

angle = 360/sides
for count in range(sides):
    forward(length)
    right(angle)
end_fill()
```

Akkurat som med `pencolor` vel `fillcolor` farga skjelpadda skal bruke for å fylle inn figurane du teiknar. Denne koden teiknar ein raud firkant med ein raud strek rundt.

Du kan bruke `begin_fill()` for å fortelje skjelpadda at den skal fargeleggje figuren du teiknar, og seie `end_fill()` for å seie at du er ferdig.

- [ ] Prøv å endre fargane, sidene og lengdene og sjå kva figurar du kan teikne!

Steg 6: Pennen går opp, pennen går ned

Viss du vil flytte skjelpadda utan at den skal setje spor etter seg, så kan du bruke `penup()` og `pendown()` for å slå av og på at skjelpadda skal teikne.

Sjekkliste

- [ ] Prøv dette i ei ny fil:

```
from turtle import *

speed(11)
shape("turtle")

pencolor('red')

for count in range(20):
    penup()
    forward(10)
    pendown()
    forward(20)
```

- [ ] Dette burde teikne ein stipla strek over skjermen din. Køyr koden og sjå!

Heim, kjære heim på skjermen

Eit par triks på slutten: `home()` får skjelpadda til å gå heim dit den starta, `clear()` tørkar alle spora av skjermen, og `reset()` flyttar skjelpadda *og* reinsar skjermen.

Steg 7: Gjer kva du vil!

Du kan `forward()`, `backward()`, `left()`, `right()`, du kan gjenta ting med `for count in range(4)`, endre farger, endre fart og til og med fylle figurar!

Kan du teikne eit hus? Ein fugl? Ein slange? Ein katt? Ein hund? Ei løve? Du kan kombinere figurar og sjå kva du kan lage. Kan du teikne ein robot?