# Skilpadder

## Introduksjon:

I denne modulen skal vi lære et programmeringsspråk som heter Python. Personen som laget det kalte det opp etter sitt favorittprogrammet på TV: Monthy Pythons Flyvende Cirkus. Python brukes av mange forskjellige programmerere til mange forskjellige ting. Python brukes av YouTube, NASA, CERN og andre. Hvis din Kodeklubb har en Raspperry Pi kan du bruke Python til å programmere den. Mange elsker Python fordi de synes det er lett å lese (i motsetning til språk de synes er vanskeligere å lese). Her er meningene mange, når du lærer flere språk får du sikkert også sterke meninger om dem. Å være i stand til å lese kode er viktig for en programmerer, kanskje like viktig som å kunne skrive den.

# Steg 0: Åpne Python-editoren

Hvis Python allerede er installert på maskinen din er det bare å begynne.

- [ ] På Windows, finn IDLE i start-menyen
- [ ] På Mac OS X, åpne Terminal.app og skriv `idle` og trykk enter.
- [ ] På Linux, åpne opp en Terminal, skrive `idle` og trykk enter.

Hvis du ikke har Python installert fra før - ikke få panikk! Du kan laste ned siste versjon av Python fra http://www.python.org/. Nøyaktig hvilken versjon det er spiller liten rolle, men vi bruker Python 3, så versjonsnummeret bør begynne med 3 (og ikke 2).

Når IDLE starter ser du et Output-vindu som heter `Python Shell`. Vi må åpne et nytt vindu for å skrive kode i. Trykk `File -> New Window` så er du klar for steg 1. Pass på at du har begge vinduene synlige.

# Steg 1: Hello, World!

`Hello, World!` betyr `Hei, Verden!` og brukes ofte for å vise frem et nytt programmeringsspråk. Alt man skal få til er å få datamaskinen til å skrive akkurat disse ordene på skjermen.

## Sjekkliste

- [ ] Åpne IDLE, editoren som følger med Python. Vi kommer til å skrive all koden vår i denne editoren. Når du åpner den ser du et Output-vindu. I dette vinduet kommer feilmeldinger og resultater til å dukke opp.
- [ ] Hvis du ikke har gjort det ennå, velg `File -> New Window`. Et tomt vindu dukker opp, med '*Untitled*' i tittelfeltet. Du skal ha to vinduer åpne nå, et for å skrive programmet ditt i, og et annet for å vise resultater i. Pass på at du skriver i det riktige!
- [ ] Skrive følgende kode inn i det nye vinduet:

```
print("Hello, World!")
```

- [ ] Dobbeltsjekk at du ikke er i Output-vinduet og lagre koden din.

Du kan lagre ved å velge `File -> Save`. Når du blir bedt om å skrive inn et filnavn, skriv hello.py, og lagre filen på skrivebordet ditt. Så kan du velge `Run -> Run Module`.

Gratulerer med ditt første Python-program :D (PS! Du kan be det skrive hva du vil, kanskje kan du prøve å få det til å si hei til deg selv?)

## Tips

- [ ] På **Windows** og **Ubuntu**, bruk Ctrl-N for å lage nye shell-vinduer, bruk Ctrl-S for å lagre og bruk F5 for å kjøre

programmet ditt. På noen maskiner må du kanskje trykke fn-knappen også.

- [ ] På **Mac OS X**, cmd-N for å lage nye shell vinduer, Command-S for å lagre, og hold nede fn-knappen og trykk F5 for å kjøre programmet ditt.

# Steg 2: Hei, Skilpadde!

Nå skal vi ha det litt gøy med skilpadder. En skilpadde er en liten robot som tegner seg selv på skjermen din, vi kan få den til å bevege seg rundt med Python-kommandoer.

## Sjekkliste

- [ ] Åpne et nytt kodevindu (Fra `File` menyen) og skriv dette:

```
from turtle import *

forward(100)
```

- [ ] Lagre programmet ditt som myturtle.py og velg `Run -> Run Module`. Ser du hvordan skilpadden beveget seg 100 punkter fremover på skjermen? Skilpadden har en penn festet til seg, så den tegner linjer når den beveger seg rundt.

### Tips

Python-filer skal alltid ha filnavn som slutter med `.py`.

## Sjekkliste

- [ ] La oss få skilpadden til å bevege seg rundt på skjermen! Forsøk å bruke `backward(distance)` i tillegg til å snu den ved å bruke `right(angle)` og `left(angle)`. Instruksjonen `backward(20)` forteller for eksempel skilpadden at den skal bevege seg bakover 20 pixler, og `right(90)` forteller den at den skal snu seg 90 grader til høyre. Du kan gi den mer enn en instruksjon om gangen, de blir utført i rekkefølge.

```
from turtle import *

speed(11)
shape("turtle")

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

### Vinkler og grader

Lek deg litt med å lage dine egne figurer ved å bruke `forward`, `backward`, `left`, `right`. Husk at `forward` og `backward` bruker pixler, mens `left` og `right` bruker grader. La oss undersøke en skilpadde som bever seg til høyre.

```
      Nord
       0
       |
Vest   |   Øst
 270 -----+----- 90
       |
       |
      180
      Syd
```

Når skilpadden ser nordover og du ber den snu seg 90 grader til høyre, ser den østover. Snur du den 180 grader fra nord, ser den sydover, og snur du den 270 grader fra nord ser den vestover. Snur du 360 grader stopper den der den begynte. Kanskje er det lettere å tenke på som snowboard-triks?

Hva med å snu mot venstre?

```
      Nord
       0
       |
Vest   |   Øst
 90 -----+----- 270
       |
       |
      180
      Syd
```

Når skilpadden ser nordover og du ber den snu seg 90 grader mot venstre, ser den vestover. Når skilpadden ser nordover og du ber den snu seg 180 grader mot venstre ser den sydover, og om den ser nordover og du ber den snu seg 270 grader ser den østover. Snur du 360 grader er du tilbake der du starter, 360 grader er alltid helt rundt.

### Hva gjør koden på starten av programmet vårt?

- [ ] `from turtle import *` forteller Python at vi vil bruke skilpadde-biblioteket (turtle), en samling av kode vi kan bruke for å tegne på skjermen. Å bruke et ferdig bibliotek gjør at vi kan spare tid og gjenbruke andres arbeid.
- [ ] `speed()` bestemmer farten til skilpadden. Vi må gi inn en verdi mellom 1 og 11, der 11 er det raskeste og 1 er det treigeste.
- [ ] `shape()` Vi bruker formen(shape) "turtle" (skilpadde), men vi kan også si at vi vil at figuren skal se ut som "arrow" (pil), "circle" (sirkel), "square" (kvadrat), "triangle" (trekant) eller "classic" (klassisk).

Vi kommer til å bruke disse instruksjonene på toppen av alle programmene våre i denne leksjonen. Hvis du vil kan du forsøke å gi skilpadden en av de andre formene, som pil, og få den til å gå så fort eller sakte som du vil.

# Steg 3: Tegne figurer!

La oss lage et kvadrat ved å fortelle skilpadden hvordan den skal bevege seg rundt.

## Sjekkliste

- [ ] Åpne en ny fil i IDLE og skriv inn følgende 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 velg `Run -> New Module`. Ser du en firkant? Skilpadden snur seg 90 grader fire ganger, og ender opp med å se den samme retningen som den starten. å snu 90, 90, 90 og så 90 grader igjen snur skilpadden totalt 360 grader.

Hva med en trekant? En trekant har tre hjørner, så vi må snu tre ganger. Hvis vi vil ende opp i samme retning, må vi snu 360 grader, akkurat som med firkanten. Derfor snur vi 120 grader, deretter 120 grader og så en gang til.

- [ ] Endre koden din til å se ut som koden under for å få den til å tegne en trekant:

```
from turtle import *

speed(11)
shape("turtle")

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

- [ ] Kjør koden. Ser du en trekant?

## Velg en farge

Hva er yndlingsfargen din? Du kan endre fargen på linjene ved å bruke funksjonen `pencolor` (Python staver på amerikansk, og amerikanerne staver colour uten u.). Du kan også endre størrelsen på pennen ved å bruke `pensize`:

## Sjekkliste

- [ ] Endre koden fra eksemplet over til å se ut som det neste eksemplet, ved å legge til disse nye kommandoene:

```
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)
```

- [ ] Kjør koden din, hva tegner den på skjermen? Denne koden tegner en tykk trekant i tre forskjellige farver.

- [ ] Forsøk å endre fargene i koden din, kjør den og se hva som skjer. Skilpadden kan mange hundre forskjellige farger, ikke bare blå, rød og grønn. Forsøk med din yndlingsfarge! Du kan også bruke farger i **hex**, som du kanskje har gjort med CSS før. Istedenfor å bruke `pencolor("red")` kan du bruke hex `pencolor("#FF0000")`. Hvilken farge er #FF4F00?

# Steg 4: Gjenta deg selv (med en for-løkke)

Det siste programmet var de samme kommandoene igjen og igjen. Istedenfor å skrive dem ned, la oss be maskinen om å gjenta dem for oss. Du har vært borti *iterasjon* i Scratch ved å bruke `Forever` og `Repeat` / `Repeat until` blokker. I Python brukes **for-løkker** når du har kode som du vil gjenta n ganger. I dette eksemplet vil vi gjenta koden (den som er skjøvet inn) 4 ganger fordi en firkant har 4 sider.

## Sjekkliste

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

```
from turtle import *

speed(11)
shape("turtle")

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

- [ ] Lagre programmet og velg: Run -> Run module.

  Legg merke til at koden er skjøvet inn, *indentert*, eller dyttet til høyre under for-løkken. Python bruker mellomrom for å vite hvilke kommandoer som skal gjentas. Du kan bruke Tab-tasten for å få IDLE til å *indentere*, eller bruke Shift-Tab til å ta dem bort.

- [ ] La oss se hva som skjer hvis vi bare indenterer (skyver inn) `forward`. Gjør 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 indendert og `right` ikke er det. Hva tror du dette programmet gjør? Forsøk å kjøre det og finn det ut.

  Fikk du en rett linje? Python vil gjenta `forward` fire ganger, og deretter snu til høyre. Python bruker mellomrom for å gruppere kommandoer sammen, akkurat som Scratch bruker blokker. Python klager til deg om du ikke har fått mellomrommene riktig.

- [ ] La oss endre programmet tilbake slik at det lager en firkant igjen, men istedenfor å bruke tall i koden skal vi gi tallene navn. Dette gjør det lettere å se hva programmet gjør, og gjør at vi slipper å gjenta oss selv.

  Endre filen 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 velg: `Run -> Run module`.

## Oppgave: Tegn de andre formene

Kan du tegne noen av figurene under bare ved å endre verdiene?

- [ ] En trekant? (tre sider)
- [ ] Et pentagram? (fem sider)
- [ ] Et hexagram? (seks sider)
- [ ] Et oktagram? (åtte sider)

Husk, en trekant har tre sider og vi må derfor snu 120 grader i hvert av de tre hjørnene for at det skal bli 360 grader tilsammen. For en firkant må vi snu 90 grader i hvert hjørne, som også blir 360 grader.

Hvis du snur seks ganger, hvor mange ganger må du snu for at det skal bli 360 grader? Prøv med forskjellige tall og se hva som skjer.

# Steg 5: Snu, snu, snu

Istedenfor å regne ut vinklene, kan vi ikke heller få datamaskinen til å gjøre det for oss? Python lar deg plusse, trekke fra, gange og dele. Vi kan skrive `sides = 4 + 1` istedenfor 5, eller `sides = 4 - 1` istedenfor 3. For multiplikasjon bruker Python `*`, og for divisjon skriver vi `/`. Hvis vi må snu 360 grader tilsammen, kan vi regne ut vinkelen vi trenger. For en firkant er `360 / 4` lik 90, for trekanten er `360 / 3` lik 120.

## Sjekkliste

- [ ] Endre programmet ditt til å regne 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)
```

- [ ] Nå kan du endre antall sider, klarer Python å gjøre jobben riktig? Prøv med så mange kanter du vil!

# Steg 6: Fylte figurer

## Sjekkliste

- [ ] Vi kan be skilpadden om å fylle figurene med en farge ved å bruke `begin_fill()` og `end_fill()`. Endre koden din til å bruke disse kommandoene:

```
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` velger `fillcolor` fargen skilpadden skal bruke for å fylle inn figurene du tegner. Denne koden tegner en rød firkant med en rød strek rundt.

Du kan bruke `begin_fill()` for å fortelle skilpadden at den skal fargelegge figuren du tegner, og si `end_fill()` for å si at du er ferdig.

- [ ] Forsøk å endre fargene, sidene og lengdene og se hvilke figurer du kan tegne!

# Steg 7: Pennen går opp, pennen går ned

Hvis du vil flytte skilpadden uten at den skal sette spor etter seg, kan du bruke `penup()` og `pendown()` for å slå av og på at skilpadden skal tegne.

## Sjekkliste

- [ ] Forsøk dette i en ny fil:

```
from turtle import *

speed(11)
shape("turtle")

pencolor('red')

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

- [ ] Dette burde tegne en stiplet strek over skjermen din. Kjør det og se!

## Hjem, kjære hjem på skjermen

Et par triks på slutten: `home()` får skilpadden til å gå hjem dit den begynte, `clear()` tørker alle sporene av skjermen, og `reset()` flytter skilpadden og renser opp skjermen.

# Steg 8: Gjør hva 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 figurer!

Kan du tegne et hus, en fugl? En slange? En katt? En hund? En løve? Du kan kombinere figurer og se hva du kan lage. Kan du tegne en robot?