Oppgaver

I denne seksjonen finner du oppgaver som hører til dag 1 av Kodeskolens kræsjkurs i programmering.

Tema for første dag er variabler, skilpaddeprogrammering, løkker og betingelser. Dersom du står fast er det bare å spørre. God koding!

Variabler og regning

Oppgave 1 *Finn tre feil!*

Her følger det tre programmer som ikke fungerer. Skriv programmet inn på din egen maskin og kjør det, les feilmeldingen, og prøv å tolke den. Når du skjønner hva som er galt, rett opp feilen og kjør programmet.

a)

```
print(Hva heter du?)
```

b)

```
navn = 'Sahid'
print('hei' navn)
```

c)

```
x = '3'
z = x + 2
print('x + y =', z)
```

Løsning oppgave 1 *Finn tre feil!*

a) Vi må huske hermetegnene (') til strengene våre:

```
print('Hva heter du?')
```

b) Vi må huske , mellom de tingene vi vil skrive ut på skjermen

```
navn = 'Sahid'
print('hei', navn)
```

c) Vi må passe på å bruke tall, ikke strenger hvis vi vil gjøre matematikk

```
x = 3
y = 2
z = x + y
print('x + y =', z)
```

Oppgave 2 *Konvertering av temperatur*

I Norge oppgir vi temperaturer i målestokken *Celsius*, men i USA bruker de ofte målestokken *Fahrenheit*. Hvis du finner en kakeoppskrift fra USA kan det for eksempel stå at du skal bake kaken ved 350 grader. Da mener de altså 350°F. Vi vil nå lage et verktøy som kan konvertere denne temperaturen for oss, sånn at vi vet hva vi skal bake kaken ved i Celsius..

For å regne over fra Fahrenheit til Celsius bruker vi formelen:

$$C = \frac{5}{9}(F - 32).$$

Der F er antall grader i Fahrenheit, og C blir antall grader i celsius.

a) Start med å opprette en variabel, `farenheit`, som du setter lik 350.

b) Lag et program regner ut hvor mange grader Celsius 350 °F tilsvarer. Virker det rimelig å skulle bake en kake ved denne temperaturen?

Programmet du har lagd tar en temperatur i Fahrenheit, og gjør om til Celsius. Men hva om vi ønsker å gå motsatt vei? Om vi ønsker å lage et nytt program som gjør motsatt, så må vi først ha en formel for F.

c) Klarer du å ta uttrykket

$$C = \frac{5}{9}(F - 32).$$

og løse for F?

d) Lag et nytt program som har en variabel, `farenheit`, som du setter lik 220. Regn så ut hvor mange grader Farenheit dette er, og skriv det ut til brukeren.

e) Modifiser programmet ditt til å finne frysepunktet og kokepunktet til vann i Fahrenheit målestokken.

Løsning oppgave 2 *Konvertering av temperatur*

a)

```
fahrenheit = 350
celcius = (5/9)*(fahrenheit-32)

print(f'{fahrenheit} grader fahrenheit
   tilsvarer {celcius:.0f} celcius')
```

b)

```
Fahrenheit: 350
350.0 grader fahrenheit tilsvarer 177 celcius
```

177 ° C virker som en rimelig kakebaketemperatur

c)

$$F = \frac{9}{5}C + 32.$$

d)

```
celcius = 220
fahrenheit = (9/5)*celcius + 32

```

```
4       print(f'{celcius} grader celcius tilsvarer {
            fahrenheit:.0f} fahrenheit')
```

e)

```
Celcius: 0
0.0 grader celcius tilsvarer 32 fahrenheit
```

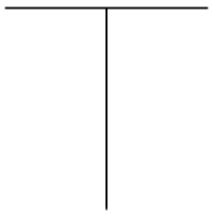
```
Celcius: 100
100.0 grader celcius tilsvarer 212 fahrenheit
```

Skilpaddeprogrammering

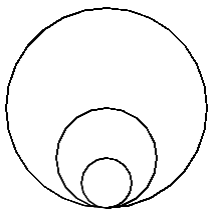
Oppgave 3 *Vår første turtletegning*

Bruk turtle til å gjenskape figurene under. (HINT: du kan finne en liste av alle mulige turtle-kommandoer i dokumentasjonen: https://docs.python.org/3/library/turtle.html)

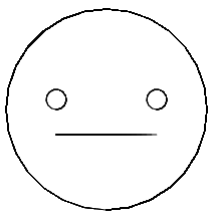
a)



b)



c)



(HINT: For å løfte pennen mellom tegninger trenger du penup og pendown funksjonene)

Løsning oppgave 3 *Vår første turtletegning*

a)

```
## Med stjerneimportering
from turtle import *

forward(100)
backward(50)
right(90)
forward(100)

done()
bye()  # Nødvendig hvis vi bruker Spyder
## Med turtle objekt
import turtle

penn = turtle.Turtle()
penn.forward(100)
penn.backward(50)
penn.right(90)
penn.forward(100)

turtle.done()
turtle.bye()  # Nødvendig hvis vi bruker Spyder
```

b)

```
## Med stjerneimportering
from turtle import *

circle(100)
circle(50)
circle(25)
done()
```

```
bye()  # Nødvendig hvis vi bruker Spyder
## Med turtle objekt
import turtle

penn = turtle.Turtle()
penn.circle(100)
penn.circle(50)
penn.circle(25)
turtle.done()

turtle.bye()  # Nødvendig hvis vi bruker Spyder
```

c)

```
## Med stjerneimportering
from turtle import *

circle(100)
penup()
left(90)
forward(100)
right(90)
forward(50)
pendown()
circle(10)
penup()
backward(100)
pendown()
circle(10)
right(90)
penup()
forward(25)
pendown()
left(90)
forward(100)

done()
bye()  # Nødvendig hvis vi bruker Spyder
## Med turtle objekt
import turtle
```

```
27
28  penn = turtle.Turtle()
29
30  penn.circle(100)
31  penn.penup()
32  penn.left(90)
33  penn.forward(100)
34  penn.right(90)
35  penn.forward(50)
36  penn.pendown()
37  penn.circle(10)
38  penn.penup()
39  penn.backward(100)
40  penn.pendown()
41  penn.circle(10)
42  penn.right(90)
43  penn.penup()
44  penn.forward(25)
45  penn.pendown()
46  penn.left(90)
47  penn.forward(100)
48  turtle.done()
49
50  turtle.bye()  # Nødvendig hvis vi bruker Spyder
```

Oppgave 4 *Vi slår opp i dokumentasjonen*

Under er en kodesnutt som tegner en tegning:

```
1   from turtle import *
2
3   left(60)
4   forward(50)
5   right(40)
6   forward(30)
7   home()
8   left(120)
9   forward(50)
10  left(40)
```

```
11 forward(30)
12 home()
13
14 done()
15 bye()  # Nødvendig hvis vi bruker Spyder
```

a) Kjør koden. Hva tror du `home()` gjør?

b) Se i dokumentasjonen (https://docs.python.org/3/library/turtle.html). Hvordan beskrives `home`-funksjonen? Hadde du rett?

Løsning oppgave 4 *Vi slår opp i dokumentasjonen*

a) `home`-funksjonen flytter skilpadden tilbake til start (koordinatene $(0, 0)$).

Lister og løkker

Oppgave 5 *Fruktliste*

I denne oppgaven skal vi øve å opprette lister og hente ut elementer med indeksering

a) Opprett en liste, `frukt`, som inneholder elementene `'eple'`, `'honningmelon'`, `'kiwi'` og `'appelsin'`

b) Hvilken indeks har `'honningmelon'`? Bruk indeksering til å hente ut dette elementet fra lista og skriv det ut til terminalen

c) Hvilket element får du dersom du indekserer med `[-1]`? Skriv ut elementet med indeks `-1` og se om du hadde rett.

Løsning oppgave 5 *Fruktliste*

a)

```
frukt = ['eple', 'honningmelon', 'kiwi', 'appelsin
    ']
```

b)

```
print(frukt[1])
```

c)

```
print(frukt[-1])
```

```
appelsin
```

Oppgave 6 *For-løkker*

Syntaksen for å løkke over en liste med en for-løkke i Python er slik:

```
for [løkkevariabel] in [liste]:
    [det som skal gjentas, f.eks print(løkkevariabel)]
```

a) Opprett en liste, navneliste, som skal inneholde fem navn. (Du kan velge selv hvilke navn)

b) Bruk en for-løkke til å gå igjennom lista med navn og skrive ut en hilsen til hvert navn

Løsning oppgave 6 *For-løkker*

a)

```
navneliste = ["Olivia", "Henrik", "Emma", "Eline",
    "Liam"]
```

b)

```python
for navn in navneliste:
  print(f"Heisann, {navn}!")
```

Oppgave 7 *Trekantblomst*

Les koden under

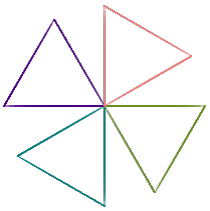
```python
from turtle import *

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

done()
bye()  # Nødvendig hvis vi bruker Spyder
```

a) Hva tror du den tegner? Kjør koden og se om du hadde rett.

b) Gjør om koden slik at den bruker en `for`-løkke til å tegne den samme figuren uten å skrive `forward` eller `right` mer en en gang hver.

c) Opprett en en liste `farge_liste` som inneholder fargene `'olivedrab'`, `'teal'`, `'indigo'` og `'lightcoral'`

d) Bruk en `for`-løkke og `color`-kommandoen sammen med `farge_liste` og koden fra oppgave b) til å gjenskape blomsten under.



Løsning oppgave 7 *Trekantblomst*

a) Koden tegner en trekant

b)

```
from turtle import *

for side in range(3):
    forward(100)
    right(120)

done()
bye()  # Nødvendig hvis vi bruker Spyder
```

c)

```
farge_liste = ["olivedrab", "teal", "indigo", "
   lightcoral"]
```

d)

```
from turtle import *

farge_liste = ["olivedrab", "teal", "indigo", "
   lightcoral"]
for farge in farge_liste:
    color(farge)
    for side in range(3):
        forward(100)
        right(120)
    right(90)

done()
bye()  # Nødvendig hvis vi bruker Spyder
```

Betingelser

Oppgave 8 *Vinkeltyper*

Vi kan dele vinkler inn i tre forskjellige typer:

1. En vinkel som er mindre enn 90 ° kalles en *spiss* vinkel.
2. En vinkel som er større enn 90 ° kalles en *stump* eller *butt* vinkel.
3. En vinkel som er akkurat 90 ° kalles en *rett* vinkel

a) Opprett en variabel `vinkel` som du gir en verdi mellom 0 og 360. Bruk så `if`, `elif` og `else` til printe ut om vinkelen er er *spiss*, *stump* eller *rett*

b) Modifiser koden slik at du bruker en for-løkke, med en tellevariabel, `vinkel`, som varierer mellom 0 og 360 grader. La det gå 10 grader mellom hver vinkel og print ut ut om hver av disse 36 vinklene er spiss, stump eller rett. (hint, du kan bruke `range(0, 360, 10)` for å generere vinkler mellom 0 ° og 360 ° med 10 ° mellom hver vinkel)

Løsning oppgave 8 *Vinkeltyper*

a)

```
vinkel = 36

if vinkel < 90:
    print(f'{vinkel} er en spiss vinkel')
elif vinkel > 90:
    print(f'{vinkel} er en butt vinkel')
else:
    print(f'{vinkel} er en rett vinkel')
```

b)

```
for vinkel in range(0, 360, 10):
    if vinkel < 90:
        print(f'{vinkel} er en spiss vinkel')
    elif vinkel > 90:
```

```
5           print(f'{vinkel} er en butt vinkel')
6       else:
7           print(f'{vinkel} er en rett vinkel')
```

Oppgave 9 *Mønstre med løkker og betingelser*

I denne oppgaven skal vi se på hvordan vi kan kombinere løkker og betingelser for å lage fine mønstre.

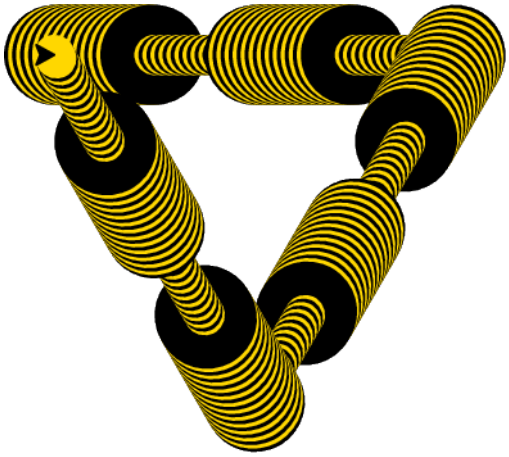
a) Bruk `pensize`-kommandoen for å sette pennestørrelsen lik 50.

b) Lag en for-løkke som du itererer over 100 ganger. For hver iterasjon skal du bevege skilpadden to steg fremover

c) Bruk en betingelse for å skjekke om tellevariabelen er et partall eller et oddetall. Dersom den er et partall skal du sette pennefargen til svart (`'black'`) og dersom det er et oddetall skal du sette pennefargen til gull (`'gold'`). (Hint: du kan bruke `tall % 2== 0` for å sjekke om `tall` er delelig på 2)

d) Nå skal vi også endre pennestørrelsen inne i løkka for å lage et interessant mønster. På starten av løkka skal du bruke `if` og `elif` for å sette pennestørrelsen til 20 på iterasjon 25 og iterasjon 75, og 50 på iterasjon 0 og iterasjon 50.

e) Nå har vi kode for å lage en interessant strek. Det neste vi skal gjøre er å bruke denne koden for å lage en kul trekant. For å gjøre det må du plassere løkka som lager en rett strek inni en ny løkke som du itererer over tre ganger. Etter at hver strek har blitt tegnet bruker du `right`-funksjonen for å snu skilpadda 120°. Nedenfor ser du hvordan sluttfiguren din kan se ut.



Løsning oppgave 9 *Mønstre med løkker og betingelser*

a)

```
from turtle import *

pensize(50)

done()
bye()  # Nødvendig hvis vi bruker Spyder
```

b)

```
from turtle import *

pensize(50)

for tellevariabel in range(100):
    forward(2)


done()
bye()  # Nødvendig hvis vi bruker Spyder
```

c)

```python
from turtle import *

pensize(50)

for tellevariabel in range(100):
    if tellevariabel % 2 == 0:
        color('black')
    else:
        color('gold')
    forward(2)


done()
bye()  # Nødvendig hvis vi bruker Spyder
```

d)

```python
from turtle import *

for tellevariabel in range(100):
    if tellevariabel == 0:
        pensize(50)
    elif tellevariabel == 25:
        pensize(20)
    elif tellevariabel == 50:
        pensize(50)
    elif tellevariabel == 75:
        pensize(20)
    if tellevariabel % 2 == 0:
        color('black')
    else:
        color('gold')
    forward(2)


done()
bye()  # Nødvendig hvis vi bruker Spyder
```

e)

```python
from turtle import *

for side in range(3):
    for tellevariabel in range(100):
        if tellevariabel == 0:
            pensize(50)
        elif tellevariabel == 25:
            pensize(20)
        elif tellevariabel == 50:
            pensize(50)
        elif tellevariabel == 75:
            pensize(20)

        if tellevariabel % 2 == 0:
            color('black')
        else:
            color('gold')
        forward(2)
    right(120)


done()
bye()  # Nødvendig hvis vi bruker Spyder
```