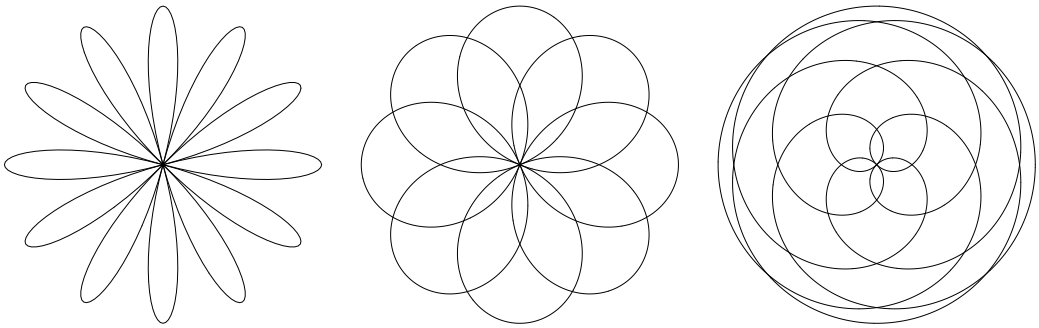
Bonusoppgaver

I denne oppgavesamlingen finner du et knippe med bonusoppgaver for dag 1 dersom du vil ha litt mer å bryne deg på. God koding!

Skilpaddeprogrammering

Oppgave 1 *Matematiske roser*



Bildet over viser tre eksempler på såkalte "rosekurver". Rosekurvene ble først studert av en italiensk matematikker ved navn Guido Grandi. Han syntes disse kurvene minnet om roser og ga dem derfor navnet "rhodone". I denne oppgaven skal vi se på hvordan vi kan tegne "roser" med skilpaddeprogrammering og trigonometri.

For å tegne en rosekurve må vi først se på hvordan vi kan tegne en sirkel ved å løkke oss igjennom vinklene fra `0` til `360`. For å gjøre dette, bruker vi parameterfremstillingen til en sirkel:

$$x(\theta) = rcos(\theta), \tag{1}$$
$$y(\theta) = rsin(\theta), \tag{2}$$

Hvor θ er en vinkel som varierer fra $0°$ til $360°$ og r er radiusen til sirkelen. Under har vi en skjelettkode som du kan bruke for å tegne en sirkel:

```
from math import pi, cos, sin
import turtle

penn = turtle.Turtle()
```

```
5  penn.speed('fastest')
6
7  radius = 50
8
9  # Vi flytter skilpadden dit vi starter sirkelen
10 penn.penup()
11 penn.goto(radius, 0)
12 penn.pendown()
13
14 # Denne løkka tegner sirkelen
15 for grader in range(360):
16     radianer = pi*grader/180
17     x = radius * cos(radianer)
18     y = radius * sin(radianer)
19     penn.goto(x, y)
20
21 turtle.done()
22 turtle.bye()  # Nødvendig hvis vi bruker Spyder
```

a) Kjør koden over for å tegne en sirkel

b) Endre `radius` til `100` og kjør koden. Hva skjer med sirkelen?

Når vi tegner en sirkel, så er radiusen konstant hele tiden, men vi kan også modifisere radiusen inne i løkka.

c) Endre linje 7 og 11 slik at `radius`-variabelen får et nytt navn: `startradius` . Legg så til en ny linje i i løkka hvor du setter radiusen til å være lik `startradius +10*radianer`. Da skal radiusen øke i takt med at vinkelen øker. Hva slags figur tror du at du får? Kjør koden og se om du fikk rett.

d) Når vi tegner en sirkel, så er det ikke noen vits i å bevege seg rundt mer enn en runde rundt origo, men når vi tegner spiraler, så kan vi bevege oss rundt mange ganger! Lag en ny variabel `antall_runder` som du setter lik `2` og endre antallet grader du løkker deg igjennom ved

å endre `for grader in range(360):` til å bli `for grader in range(antall_runder*360):`. Hva tror du skjer når du kjører denne koden? Prøv også med andre verdier for antall runder (f.eks 0.5)).

Guido så på sirkeltegninger sånn som vi gjør nå og stilte seg spørsmålet: hva skjer med figuren om radiusen endrer seg som en cos-kurve?

e) Endre koden din slik at `radius =startradius*cos(k*radianer)`. Velg selv hva `k` skal være. Hva slags figur får du nå?

Guido fant ut at vi kan få mange fine rosekurver ved å endre k til å være forskjellige brøker gitt ved $k = n/d$.

f) Prøv med `n = 4` og `d = 3`. For å få en fullstendig kurve må `antall_runder` være lik `d`.

g) Prøv deg frem med forskjellige verdier av `n` og `d`

h) Wikipedia har en illustrasjon over forskjellige roser du kan få for forskjellige verdier av `n` og `d`. Gå inn på Wikipedia: https://en.wikipedia.org/wiki/Rose_(mathematics) og velg en rose du syntes ser fin ut. Oppdater koden din til å gjenskape denne rosen

Løsning oppgave 1 *Matematiske roser*

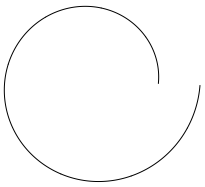
b)

```
from math import pi, cos, sin
import turtle

penn = turtle.Turtle()
penn.speed('fastest')

radius = 100

```

```
# Vi flytter skilpadden dit vi starter sirkelen
penn.penup()
penn.goto(radius, 0)
penn.pendown()

# Denne løkka tegner sirkelen
for grader in range(360):
    radianer = pi*grader/180
    x = radius * cos(radianer)
    y = radius * sin(radianer)
    penn.goto(x, y)

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

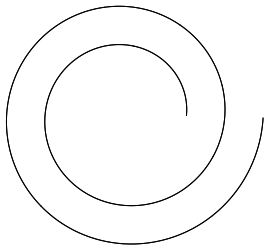
c)

```
from math import pi, cos, sin
import turtle

penn = turtle.Turtle()
penn.speed('fastest')

startradius = 100

# Vi flytter skilpadden dit vi starter sirkelen
penn.penup()
penn.goto(startradius, 0)
penn.pendown()

# Denne løkka tegner sirkelen
for grader in range(360):
    radianer = pi*grader/180
    radius = startradius + 10*radianer
    x = radius * cos(radianer)
    y = radius * sin(radianer)
    penn.goto(x, y)

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



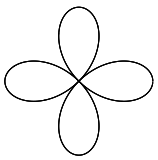
d)

```python
from math import pi, cos, sin
import turtle

penn = turtle.Turtle()
penn.speed('fastest')

antall_runder = 2
startradius = 100

# Vi flytter skilpadden dit vi starter sirkelen
penn.penup()
penn.goto(startradius, 0)
penn.pendown()

# Denne løkka tegner sirkelen
for grader in range(antall_runder*360):
    radianer = pi*grader/180
    radius = startradius + 10*radianer
    x = radius * cos(radianer)
    y = radius * sin(radianer)
    penn.goto(x, y)

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



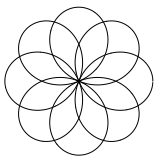
e)

```
from math import pi, cos, sin
import turtle

penn = turtle.Turtle()
penn.speed('fastest')

antall_runder = 1
k = 2
startradius = 100

# Vi flytter skilpadden dit vi starter sirkelen
penn.penup()
penn.goto(startradius, 0)
penn.pendown()

# Denne løkka tegner sirkelen
for grader in range(antall_runder*360):
    radianer = pi*grader/180
    radius = startradius * cos(k*radianer)
    x = radius * cos(radianer)
    y = radius * sin(radianer)
    penn.goto(x, y)

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



f)

```python
from math import pi, cos, sin
import turtle

penn = turtle.Turtle()
penn.speed('fastest')

n = 4
d = 3
k = n/d
antall_runder = d
startradius = 100

# Vi flytter skilpadden dit vi starter sirkelen
penn.penup()
penn.goto(startradius, 0)
penn.pendown()

# Denne løkka tegner sirkelen
for grader in range(antall_runder*360):
    radianer = pi*grader/180
    radius = startradius * cos(k*radianer)
    x = radius * cos(radianer)
    y = radius * sin(radianer)
    penn.goto(x, y)

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



Tilfeldighet

Oppgave 2 *Terningskast*

I denne koden skal du bruke `randint` til å kaste en tilfeldig terning

a) Bruk `randint` til å simulere en sekssidet terning ved å trekke et tilfeldig tall mellom 1 og 6 med `randint`. Skriv resultatet ut til brukeren

b) Modifiser programmet slik at du simulerer å kaste en tyvesidet terning istedenfor en sekssidet terning.

Løsning oppgave 2 *Terningskast*

a)

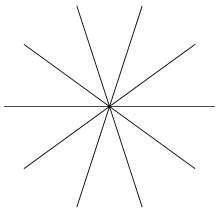
```
1 from random import randint
2
3 terningkast = randint(1, 6)
```

b)

```
1 from random import randint
2
3 terningkast = randint(1, 20)
```

Oppgave 3 *Tegne en stjerne*

Figuren under lager viser en stjerne med 10 stråler med lengde 100



a) Bruk `forward`, `backward` og `right` sammen med en `for`-løkke til å gjenskape stjernen

b) Modifiser koden din slik at stjernen har 8 stråler

c) Modifiser koden din slik at hver stråle får en tilfeldig lengde mellom 10 og 200 (hint: bruk `randint`)

d) Endre koden slik at antall stråler i stjernen blir et tilfeldig tall mellom 10 og 100

Løsning oppgave 3 *Tegne en stjerne*

a)

```
import turtle

skilpadde = turtle.Turtle()

for linje in range(10):
    skilpadde.forward(100)
    skilpadde.backward(100)
    skilpadde.right(360 / 10)

turtle.done()
```

b)

```
import turtle
```

```
2
3  skilpadde = turtle.Turtle()
4
5  for linje in range(8):
6      skilpadde.forward(100)
7      skilpadde.backward(100)
8      skilpadde.right(360 / 8)
9
10 turtle.done()
```

c)

```
1  import turtle
2  from random import randint
3
4  skilpadde = turtle.Turtle()
5
6  for linje in range(10):
7      lengde = randint(10, 200)
8      skilpadde.forward(lengde)
9      skilpadde.backward(lengde)
10     skilpadde.right(360 / 10)
11
12 turtle.done()
```

d)

```
1  import turtle
2
3  skilpadde = turtle.Turtle()
4  from random import randint
5
6  antall_linjer = randint(10, 100)
7  for linje in range(antall_linjer):
8      lengde = randint(10, 200)
9      skilpadde.forward(lengde)
10     skilpadde.backward(lengde)
11     skilpadde.right(360 / antall_linjer)
12
13 turtle.done()
```