# Pythonprogrammering med Tekna

Fasit for befolkningsvekst og pandemier

kodeskolen simula

kodeskolen@simula.no

**Oppgave 1** *Vekstratefunksjon*

**a)** Lag en funksjon, `vekstrate(antall_kaniner)` som tar inn antall kaniner og returnerer vekstraten for en øy med en bæreevne på 5000 kaniner og en ubegrenset vekstrate på 0.5.

Nå har vi en funksjon som regner ut vekstraten, men hvordan vet vi at den funker? Vi vet at dersom det ikke er noen kaniner på øya, så skal vekstraten være lik den ubegrensede vekstraten, altså $r(0) = 0.5$. Vi vet også at dersom det er 5000 kaniner på øya, så skal vekstraten være 0.

**b)** Sjekk om din `vekstrate` funksjon returnerer `0.5` dersom `antall_kaniner=0` og at den returnerer `0` dersom `antall_kaniner=5000`.

Slik testing du akkurat gjorde er veldig lurt. Det er lett å skrive feil når man koder, og det kan være vanskelig å finne feilen i et stort program! Tenk hvis vi hadde skrevet `1 + antall_kaniner/5000` istedenfor `1 - antall_kaniner/5000`? Det er fort gjort, men hvis vi tester koden underveis kan vi enkelt finne mange av disse feilene.

**Hint:** I Python ser syntaksen til funksjoner slik ut:

```
def funksjonsnavn (inputvariabel):
    # [instruksjonene funksjonen skal utføre]
    return #[det som skal returneres]
```

**Løsning oppgave 1** *Vekstratefunksjon*

**a)**

```
def vekstrate(antall_kaniner):
    return 0.5 * (1 - antall_kaniner/5000)
```

**b)**

```
    print(vekstrate(0))
    print(vekstrate(5000))
```

```
0.5
0
```

**Oppgave 2** *Simulere bærekraftig vekst*

**a)** Nå har vi kode for eksponensiell vekst. Modifiser den koden slik at du simulerer en øy med bærekraftig vekst. Den ubegrensede vekstraten skal være 0.5, bæreevnen skal være 5000 kaniner og simuleringen skal vare i 40 år og du skal starte med 200 kaniner på øya.

**b)** Bruk plottefunksjonaliteten i Pylab for å plotte resultatet fra simuleringen. Ser det mer realistisk ut enn den eksponensielle veksten? Stabiliserer populasjonen seg? i såfall, hva stabiliserer den seg til?

**Løsning oppgave 2** *Simulere bærekraftig vekst*

**a)**

```
def vekstrate(antall_kaniner):
    return 0.5 * (1 - antall_kaniner/5000)

antall_år = 40

k = zeros(antall_år)
k[0] = 200

for t in range(1, antall_år):
    k[t] = k[t-1] + vekstrate(k[t-1])*k[t-1]
```

**b)**

```
plot(range(antall_år), k, '-o')
xlabel('Tid [år]')
ylabel('Antall kaniner')
show()
```

**Oppgave 3** *Å utforske bærekraftig vekst*

**a)** Modifiser bæreevnen til systemet. Hvordan oppfører populasjonen til en øy med en bæreevne på 10 000 kaniner?

Nå skal vi utforske hva som skjer dersom det oppstår en naturkatastrofe på øya. For å gjøre det må vi kunne sette bæreevnen til øya manuelt inne i simuleringsløkka vår.

**b)** Lag en funksjon, `vekstrate(antall_kaniner, ubegrenset_vekstrate, bæreevne)`, som tar inn antall kaniner på øya, den ubegrensede vekst-

rate og øyas bæreevne og returnerer vekstraten for en øy med det gitte antallet kaniner, ubegrensede vekstrate og bæreevne.

**c**) Modifiser koden du skrev i Oppgave 2 slik at den bruker den nye vekstratefunksjonen din. Sett den ubegrensede vekstraten lik 0.5 og bæreevnen lik 5000 kaniner.

Med dette har vi alle puslespillbrikkene vi trenger for å simulere en naturkatastrofe på øya! Det kan vi gjøre ved å endre bæreevnen på øya etter et gitt antall år.

**d**) Bruk en betingelse (`if`) for å sjekke om det har gått mer enn 20 år siden starten av simuleringen (`t > 20`). Hvis det har gått mer enn 20 år, skal bæreevnen være lik 2000, hvis ikke (`else`) skal bæreevnen være lik 5000.

**Hint:** Syntaksen til betingelser ser slik ut:

```
x = 5
if x < 3:
    print("x er mindre enn 3")
else:
    print("x er ikke mindre enn 3")
```

**Løsning oppgave 3** *Å utforske bærekraftig vekst*

**a**)

```
def vekstrate(antall_kaniner):
    return 0.5 * (1 - antall_kaniner/10000)

antall_år = 40

k = zeros(antall_år)
k[0] = 200

for t in range(1, antall_år):
```

```
    k[t] = k[t-1] + vekstrate(k[t-1])*k[t-1]

plot(range(antall_år), k, '-o')
xlabel('Tid [år]')
ylabel('Antall kaniner')
show()
```

**b)**

```
def vekstrate(antall_kaniner, ubegrenset_vekstrate, bæreevne):
    return ubegrenset_vekstrate * (1 - antall_kaniner/bæreevne)
```

**c)**

```
def vekstrate(antall_kaniner, ubegrenset_vekstrate, bæreevne):
    return ubegrenset_vekstrate * (1 - antall_kaniner/bæreevne)

antall_år = 40

k = zeros(antall_år)
k[0] = 200

for t in range(1, antall_år):
    k[t] = k[t-1] + vekstrate(k[t-1], 0.5, 5000)*k[t-1]

plot(range(antall_år), k, '-o')
xlabel('Tid [år]')
ylabel('Antall kaniner')
show()
```

**d)**

```
def vekstrate(antall_kaniner, ubegrenset_vekstrate, bæreevne):
```

```
2       return ubegrenset_vekstrate * (1 -
            antall_kaniner/bæreevne)
3
4   antall_år = 40
5
6   k = zeros(antall_år)
7   k[0] = 200
8
9   for t in range(1, antall_år):
10      if t > 20:
11          bæreevne = 2000
12      else:
13          bæreevne = 5000
14      k[t] = k[t-1] + vekstrate(k[t-1], 0.5, bæ
            reevne)*k[t-1]
15
16  plot(range(antall_år), k, '-o')
17  xlabel('Tid [år]')
18  ylabel('Antall kaniner')
19  show()
```

**Oppgave 4**

**a)** Modifiser koden vi har for å simulere en øy med gauper og kaniner slik at gaupenes dødsrate er lik 0.8 ($d_g = 0.8$). Hvordan påvirker det kaninbestanden på øya?

**b)** Hva skjer om du også starter med 4 gauper istedenfor 2 gauper?

**Løsning oppgave 4**

**a)** Den eneste endringen er på linje 10:

```
1   from pylab import zeros
```

```
2
3  dager_i_år = 365
4  antall_år = 10
5  antall_dager = antall_år*dager_i_år
6
7  f_k = 5/dager_i_år
8  f_g = 0.15/dager_i_år
9  a_g = 1.5/dager_i_år
10 d_g = 0.8/dager_i_år
11
12 k = zeros(antall_dager)
13 k[0] = 10 # start mengde kaniner
14 g = zeros(antall_dager)
15 g[0] = 2
16
17 for t in range(antall_dager - 1):
18     k[t+1] = k[t] + f_k*k[t] - a_g*g[t]*k[t]
19     g[t+1] = g[t] + f_g*g[t]*k[t] -  d_g*g[t]
```

**b**) Maksimalt antall gauper øker og maksimalt antall kaniner synker (sett `g[0] =4`).

### Oppgave 5

Bytt ut kaninvekstraten i koden over slik at vi tar hensyn til bærekraftig vekst. Sett bæreevnen til øya lik 5000 kaniner og den ubegrensede vekstraten til `5 / dager_i_år`.

Du skal altså modifisere kanin-oppdateringsreglene til å bli

$$k_t = k_{t-1} + f_k(k_{t-1})k_{t-1} - a_g g_{t-1} k_{t-1}, \tag{1}$$

hvor kaninfødselsraten er gitt ved

$$f_k(k) = \frac{5}{365}\left(1 - \frac{k}{5000}\right). \tag{2}$$

Hvordan påvirker dette kanin- og gaupepopulasjonen på øya? Var det store endringer?

**Hint:** Du kan f.eks. ta utgangspunkt i funksjonen under

```
def f_k(k):
    return (1 - k/5000) * 5 / 365
```

**Løsning oppgave 5**

Plottet ser nesten helt likt ut som den originale modellen uten bærekraftig vekst.

```
from pylab import zeros, plot, xlabel, ylabel, legend,
    show

dager_i_år = 365
antall_år = 10
antall_dager = antall_år*dager_i_år

def f_k(k):
    return (1 - k/5000) * 5 / 365
f_g = 0.15/dager_i_år
a_g = 1.5/dager_i_år
d_g = 0.8/dager_i_år

k = zeros(antall_dager)
k[0] = 10 # start mengde kaniner
g = zeros(antall_dager)
g[0] = 2

for t in range(antall_dager - 1):
    k[t+1] = k[t] + f_k(k[t])*k[t] - a_g*g[t]*k[t]
    g[t+1] = g[t] + f_g*g[t]*k[t] - d_g*g[t]

plot(range(antall_dager), k, label='Kaniner')
plot(range(antall_dager), g, label='Gauper')

legend()
xlabel('Dager')
ylabel('Antall dyr')
```

```
28 show()
```

### Oppgave 6

Problemløsningsstrategiene vi har brukt her likner faktisk veldig på de man bruker for å simulere pandemier. Den mest kjente pandemisimuleringsmodellen heter SIR-modellen, eller susceptible (smittbar), infected (smittet) og removed (fjernet) modellen. Grunnen til at modellen heter dette er at den deler en befolkning i tre grupper, en gruppe smittbare personer, en gruppe smittede personer og en gruppe med både døde og immune personer. En smittbar person kan bli smittet og en smittet person kan bli fjernet fra simuleringen (altså enten dø eller bli immun).

SIR modellen kan beskrives med den rekursive sammenhengen

$$S_t = S_{t-1} - iI_{t-1}S_{t-1} \tag{3}$$
$$I_t = I_{t-1} + iI_{t-1}S_{t-1} - dI_{t-1} \tag{4}$$
$$R_t = R_{t-1} + dI_{t-1}. \tag{5}$$

$i$ er relatert til smitteraten og $d$ henger sammen med hvor lang tid det tar for en syk person å enten dø eller bli frisk.

Modifiser rovdyr-byttedyr systemet over til å simulere en pandemi med SIR-modellen. Start med 99998 friske personer ($S_0 = 99998$), 2 syke personer ($I_0 = 2$) og 0 fjernede personer ($R_0 = 0$). Sett $i = 0.0000125$ og $d = 0.1$. Simuler systemet i 365 dager.

**Hint:** Hvis du står fast, kan du se på neste del av kompendiet, som går mer i dybden på SIR modellen.

### Løsning oppgave 6

```
1 from pylab import zeros, plot, xlabel, ylabel, legend,
      show
2
3 antall_dager = 365
4
5 i = 0.0000125
```

```
d = 0.1

S = zeros(antall_dager)
S[0] = 99998
I = zeros(antall_dager)
I[0] = 2
R = zeros(antall_dager)

for t in range(antall_dager-1):
    S[t+1] = S[t] - i*S[t]*I[t]
    I[t+1] = I[t] + i*S[t]*I[t] - d*I[t]
    R[t+1] = R[t] + d*I[t]

plot(range(antall_dager), S, label='Friske')
plot(range(antall_dager), I, label='Syke')
plot(range(antall_dager), R, label='Immune og døde')

legend()
xlabel('Dager')
ylabel('Antall personer')
show()
```