Oppgaver

Her finner du oppgaver som hører til Kodeskolens introduksjonkurs i programmering, og dette dekker begge dagene. Tema for første dag er *introduksjon til Python*, *variabler*, *input*, *betingelser* og *løkker*. Tema for andre dag er *funksjoner* og *plotting*.

Dersom du står fast er det bare å spørre. I tillegg anbefaler vi å lese i kompendiet hvis det er noen temaer du synes er spesielt vanskelige. Oppgaver markert som bonusoppgaver er litt mer utfordrende og du velger selv om du har lyst til å prøve deg på dem.

God koding!

Innhold

1 Variabler og regning

Oppgave 1 *Printing*

a) Lag et program som skriver ut teksten «Hei, verden!», til skjermen.

b) Lag et program der du først lagrer navnet ditt i en variabel, og så få programmet ditt til å skrive ut en hilsen direkte til deg.

c) Hvis du bruker kommandoen `len()` på variabelen din får du ut antall bokstaver i navnet ditt. Endre programmet ditt så det også skriver ut denne informasjonen.

d) Du kan bruke funksjonen `input()` til å stille brukeren et spørsmål. Bruk dette til å lage et program som spør brukeren om navnet deres, og deretter skriver ut en beskjed som bruker navnet de har oppgitt. Skriv også ut hvor mange bokstaver det er i navnet til vedkommende.

Løsning oppgave 1 *Printing*

a)

```
1 print("Hei, verden!")
```

b)

```
1 navn = "Maria"
2 print(f"Hei, {navn}!")
```

c)

```
1 navn = "Maria"
2 print(f"'{navn}' har {len(navn)} bokstaver i seg."
    )
```

d)

```python
navn = input("Hva heter du?\n")
print(f"Hei, {navn}! Navnet ditt har {len(navn)}
    bokstaver i seg.")
```

Oppgave 2 *Kvadrattall*

Kvadrattall er heltall som er blitt kvadrert, altså ganget med seg selv, eller opphøyet i annen. I Python kan du regne ut kvadratet av et tall `n` enten ved å skrive `n*n` eller `n**2`.

a) Skriv et program som spør brukeren om et tall, og deretter skriver ut kvadratet av tallet brukeren ga.

Pass på at når vi spør om et tall må vi gjøre om svaret til fra brukeren til en tallvariabel ved å skrive `int` på følgende måte: `int(input())`. Dette er fordi «int» står for «integer», som er engelsk for heltall.

b) Bruk programmet ditt og prøve-feile metoden til å finne det minste tallet som har et kvadrat på over 1000.

Løsning oppgave 2 *Kvadrattall*

a)

```python
base = float(input("Skriv inn et tall:"))
kvadrat = base*base
print(f"Kvadratet av {base} er {kvadrat}")
```

b) Ved å bruke programmet og prøve-feile-metoden kan vi finne ut at det største tallet som har kvadrattall mindre enn 1000 er 31.

Oppgave 3 *Finn fire feil!*

Her følger det fire programmer som har blitt skrevet feil. Finn feilen i hver

programsnutt. Du kan godt kjøre programmet inn på din egen maskin, og kjøre det, da kan kanskje feilmeldingen hjelpe deg å skjønne hva som er galt.

Når du tror du skjønner hva som er galt, rett feilen på din egen maskin, og kjør programmet for å sjekke at det fungerer som det skal.

a)

```
print(Hei, Verden!)
```

b)

```
Print("Hei, Verden!")
```

c)

```
person = input("Hva heter du?")
print("Hei på deg, {navn}")
```

d)

```
pi = 3,14
radius = 4
areal = radius*pi**2
```

Løsning oppgave 3 *Finn fire feil!*

a) Vi må huske på fnuttene våre

```
print("Hei, Verden!")
```

b) Vi må bruke liten p i print:

```
print("Hei, Verden!")
```

c) Vi må passe på at vi bruker riktig variabel:

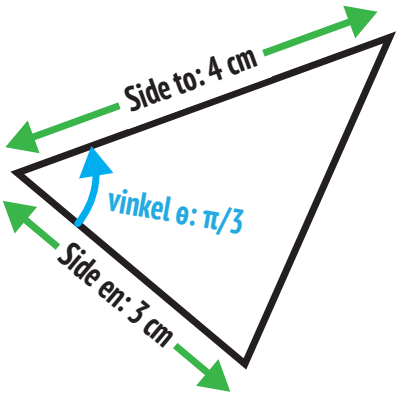
```
navn = input("Hva heter du?")
print("Hei på deg, {navn}")
```

d) Vi må bruke punktum som desimaltegn, ikke komma:

```
pi = 3.14
radius = 4
areal = radius*pi**2
```

Oppgave 4 *Arealsetningen*

Vi kjenner til to sider og vinkelen mellom dem i en trekant:



Arealsetningen sier at arealet er gitt ved:

$$\frac{1}{2} \times \text{side1} \times \text{side2} \times \sin\theta$$

a) Vi ser at vi kommer til å trenge `sin`-funksjonen. Importer `sin` og `pi` fra `math` biblioteket.

b) Lag variablene `side1`, `side2` og `vinkel` og gi dem verdiene 3, 4 og $\frac{\pi}{3}$ (**OBS:** vinkelen er gitt i radianer)

c) Regn ut arealet til trekanten og skriv resultatet ut til terminalen

d) Modifiser programmet ditt til å først spørre brukeren om verdiene til `side1`, `side2` og `vinkel` med `float` og `input` og så skrive ut arealet til terminalen.

Løsning oppgave 4 *Arealsetningen*

a)

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

b)

```python
side1 = 3
side2 = 4
vinkel = pi/3
```

c)

```python
areal = (1/2)*side1*side2*sin(vinkel)
print(f'Arealet er {areal}')
```

d)

```python
from math import sin, pi
side1 = float(input('Hva er lengden til side en?')
    )
side2 = float(input('Hva er lengden til side to?')
    )
vinkel = float(input('Hva er vinkelen mellom de?')
    )
areal = (1/2)*side1*side2*sin(vinkel)
print(f'Arealet er {areal}')
```

2 Betingelser

Oppgave 5 *Finn størst tall*

I denne oppgaven skal vi øve på `if`-tester ved å finne det største av to tall.

a) Skriv et program som ber brukeren om å skrive inn to tall og lagre dem i to variabler, `tall1` og `tall2`. Gjør om tallene til flyttall ved hjelp av `float`.

b) Utvid programmet dit ved hjelp av `if` slik at dersom `tall2` er større enn `tall2`, skriver du ut en beskjed om at `tall1` er størst.

c) Utvid programmet ditt videre ved hjelp av `elif`, slik at dersom `tall2` er større enn `tall1`, får brukeren beskjed om at `tall2` er størst.

d) Fullfør til slutt koden ved å legge til en `else`-blokk, slik at dersom tallene er like, vil brukeren få beskjed om det.

Løsning oppgave 5 *Finn størst tall*

a)

```
tall1 = float(input('Si et tall')
tall2 = float(input('Si et annet tall')
if tall1 > tall2:
    print(f'{tall1} er størst')
elif tall2 > tall1:
    print(f'{tall2} er størst')
else:
    print('Tallene er like!)
```

Oppgave 6 *Absoluttverdi*

Et reelt tall består av et fortegn og en tallverdi, kalt *absoluttverdi*. Når vi finner absoluttverdien til et tall «fjerner vi fortegnet». Det betyr at absoluttverdien til et tall alltid er positiv. Absoluttverdien til et tall a skrives $| a |$ og er

definert som:

$$\mid a \mid = \begin{cases} a, & \text{hvis } a \geq 0 \\ -a, & \text{hvis } < 0 \end{cases} \tag{1}$$

a) Lag et progam som ber brukeren om et tall ved hjelp av `input` og `float`, og skriver ut absoluttverdien av tallet

b) Lag et program som ber brukeren om to tall og skriver ut hvilket av tallene som har høyest absoluttverdi

Løsning oppgave 6 *Absoluttverdi*

a)

```
tall = float(input('Skriv inn et tall'))

if tall < 0:
    absoluttverdi = -tall
else:
    absoluttverdi = tall

print(absoluttverdi)
```

b)

```
tall1 = float(input('Skriv inn et tall'))
tall2 = float(input('Skriv inn enda et tall'))

if tall1 < 0:
    absoluttverdi1 = -tall1
else:
    absoluttverdi1 = tall1

if tall2 < 0:
    absoluttverdi2 = -tall2
```

```
11  else:
12      absoluttverdi2 = tall2
13
14  if absoluttverdi1 > absoluttverdi2:
15      print(absoluttverdi1)
16  else:
17      print(absoluttverdi2)
```

Oppgave 7 *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

Be brukeren om å gi en vinkel og bruk `if`, `elif` og `else` til å fortelle brukeren om vinkelen er *spiss*, *stump* eller *rett*

Løsning oppgave 7 *Vinkeltyper*

```
vinkel = float(input('Gi meg en vinkel')

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')
```

Oppgave 8 *ABC-formelen*

ABC-formelen for å løse annengradsformler er som følger:

La a, b og c være reelle tall, hvor $a \neq 0$. Da har likningen $ax^2 + bx + c = 0$ løsningene

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

a) Lag et progam som finner løsningene til en annengradslikning med $a = 1$, $b = 2.5$ og $c = 1$.

b) Modifiser programmet ditt så det spør brukeren om verdier for a, b og

c og skriver ut de tilhørende løsningene.

c) Dersom $b^2 - 4ac < 0$ har den tilhørende annengradslikningen ingen reell løsning. Bruk en `if`-betingelse til å informere brukeren om at det ikke finnes noen reell løsning dersom $b^2 - 4ac < 0$

d) Dersom $b^2 - 4ac = 0$ har likningen kun en løsning. Bruk en `elif` til å sjekke om dette er tilfelle og i så tilfelle informere brukeren om at det kun er en løsning

Løsning oppgave 8 *ABC-formelen*

a)

```
from math import sqrt

a = 1
b = 2.5
c = 1
løsning1 = (-b + sqrt(b**2 - 4*a*c))/(2*a)
løsning2 = (-b - sqrt(b**2 - 4*a*c))/(2*a)

print(f'Løsningene for likningen {a}x^2 + {b}x + {
   c} = er {løsning1} og {løsning2}')
```

b)

```
from math import sqrt

a = float(input('Hva er a? '))
b = float(input('Hva er b? '))
c = float(input('Hva er c? '))

løsning1 = (-b + sqrt(b**2 - 4*a*c))/(2*a)
løsning2 = (-b - sqrt(b**2 - 4*a*c))/(2*a)

print(f'Løsningene for likningen {a}x^2 + {b}x + {
   c} = er {løsning1} og {løsning2}')
```

c)

```
from math import sqrt

a = float(input('Hva er a? '))
b = float(input('Hva er b? '))
c = float(input('Hva er c? '))

rot_del = b**2 - 4*a*c

if rot_del < 0:
    print(f' {a}x^2 + {b}x + {c} = 0 har dessverre ingen reelle løsninger')
else:
    løsning1 = (-b + sqrt(rot_del))/(2*a)
    løsning2 = (-b - sqrt(rot_del))/(2*a)
    print(f'Løsningene for likningen {a}x^2 + {b}x + {c} = 0 er {løsning1} og {løsning2}')
```

d)

```
from math import sqrt

a = float(input('Hva er a? '))
b = float(input('Hva er b? '))
c = float(input('Hva er c? '))

rot_del = b**2 - 4*a*c

if rot_del < 0:
    print(f' {a}x^2 + {b}x + {c} = 0 har dessverre ingen reelle løsninger')
elif rot_del == 0:
    løsning = -b/(2*a)
    print(f'Løsningen for likningen {a}x^2 + {b}x + {c} = 0 er {løsning}')
else:
    løsning1 = (-b + sqrt(rot_del))/(2*a)
    løsning2 = (-b - sqrt(rot_del))/(2*a)
    print(f'Løsningene for likningen {a}x^2 + {b}x
```

```
            + {c} = 0 er {løsning1} og {løsning2}')
```

3 Løkker

Oppgave 9 *For-løkker*

Bruk en `for`-løkke til å gjennomføre disse oppgavene.

a) Print meldingen `'Hei, verden'` 5 ganger.

b) Print alle tallene fra 1 til 101.

c) Modifiser programmet slik at det også printer ut alle kvadrattallene (x^2) mellom 1 og 10000 (100^2). Husk at du kan opphøye tallet `tall` i andre med kommandoen `tall**2`.

Løsning oppgave 9 *For-løkker*

a)

```
for verdi in range(5):
    print("Hei, verden")
```

b)

```
for tall in range(1,101):
    print(tall)
```

c)

```
for tall in range(1,101):
    print(tall, tall**2)
```

Oppgave 10 *Finne kvadrattall og kubikktall*

Lag et program som regner ut kvadrattallet og kubikktallet av tallene 1-5, og printer ut på en linje tallet og tilhørende kvadrat og kubikk. Bruk en `for`-løkke og legg til mellomrom mellom tallene.

Løsning oppgave 10 *Finne kvadrattall og kubikktall*

```python
for tall in range(1, 6):
    kvadrat = tall**2
    kubikk = tall**3
    print(f'{tall:4} {kvadrat:4} {kubikk:4}')
```

Oppgave 11 *Fakultet*

Si at du har 5 forskjellige små skulpturer du skal sette opp på rekke. Hvor mange ulike måter kan du gjøre dette på? For den første har du 5 valg, deretter har du 4 valg, så 3, og så videre. Dermed blir antall kombinasjoner til

$$5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120.$$

I matematikken skriver vi dette mer kompakt som 5!, som vi kaller *fakultet*. Å rangere n ting kan altså gjøres på $n!$ måter, som blir

$$n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1.$$

For små tall er det enkelt å regne ut fakultet for hånd, men dersom n begynner å bli litt større vil dette ta fryktelig lang tid.

a) Du skal nå skrive et program som kan regne ut fakultet for oss. Du kan enten prøve å gjøre dette helt selv, eller følge vår «steg-for-steg»-instrukser under.

- Spør brukeren om hva n er, lagre svaret i en variabel. Husk å konvertere svaret med `int(input())`.
- Lag en variabel `fakultet`, som du gir verdien 1.
- Lag en for-løkke som går fra 1 til n.
- For hver iterasjon av løkka, gang `fakultet`-variabelen din med tallene du løkker over.
- Skriv ut det endelige svaret så brukeren kan se det.

b) Test at programmet ditt gir $5! = 120$.

c) En vanlig kortstokk har 52 unike kort. Hvor mange mulige rekkefølger kan vi få om vi stokker en kortstokk?

d) I noen kortspill legger man til 2 jokere i kortstokken, slik at det nå er 54 kort i kortstokken. Hvor mange måter blir det nå å stokke kortstokken på?

Løsning oppgave 11 *Fakultet*

a)

```python
n = int(input("Velg n: "))

fakultet = 1
for i in range(1, n+1):
        fakultet *= i

print(f"{n}! = {fakultet}")
```

b)

```
5! = 120
```

c)

```
52! = 120! =
668950291344912705758811805409037258675274633313802981029567135236
```

d)

```
Velg n:  54
54! =
   2308436973392413804720927426830275810832785645718079411322880
```

Oppgave 12 *Trekanttall*

Et trekanttall er summen av tallrekken

$$1, 2, \ldots, n.$$

For eksempel så er

$$1 + 2 + 3 + 4 + 5 = 15,$$

så da er 15 et trekanttall. Siden det var summen av de 5 første tallene i tallrekka, så sier vi at det er det *femte* trekanttallet.

La oss si at vi vil vite hva det hundrede trekanttallet er, da må vi legge sammen alle tallene fra 1 til 100. Det blir fort slitsomt og kjedelig å gjøre for hånd. Så la oss bruke programmering.

For å gjøre det lettere å sjekke om programmet vårt, så startet vi med å prøve å regne ut summen fra 1 til 5.

a) Lag en løkke som skriver ut tallene

$$1, 2, 3, 4 \text{ og } 5$$

til skjermen.

b) Endre nå løkka så du isteden finner summen av tallene 1 til 5. Da må du først lage en variabel utenfor løkka, og for hvert tall, legge det til variabelen din. Husk at du kan legge noe til en varibel med `+=`.

c) Sammenlign svaret programmet ditt gir med det vi fant for hånd. Er de to like? Hvis de ikke er det så er det noe galt!

d) Hvis programmet ditt fungerte som forventet kan du nå endre sånn at du regner summen av de første 100 tallene

$$1 + 2 + 3 + \ldots + 100.$$

Det er en kjent matematiker, Carl Friedrich Gauss, som fikk denne oppgaven av sin mattelærer når han gikk på skolen på 1700-tallet. Læreren

tenkte nok at dette skulle holde Gauss opptatt en god stund med å legge sammen tall etter tall. Gauss hadde ikke tilgang til en datamaskin, så han kunne ikke automatisere jobben slik vi har gjort, men ha la merke til et mønster i tallene. Gauss la merke til at om vi starter på begge endene av rekka får vi et mønster

$$1 + 100 = 101, 2 + 99 = 101, 3 + 98 = 101, \ldots 50 + 51 = 101.$$

Fra dette mønsteret klarte Gauss å finne en formel for summen av tallene fra 1 til n, og uttrykket hans var

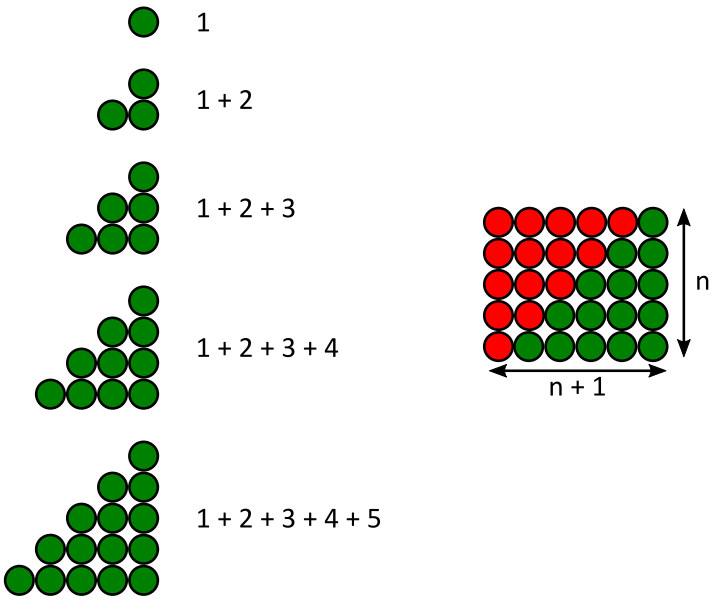
$$T_n = \frac{n(n+1)}{2}.$$

---

e) Bruk formelen til Gauss og sjekk at du får samme svar som programmet ditt for $n = 100$.

f) Gjør det samme for $n = 1000$, så $n = 1000000$ (én million).

For å skjønne hvorfor denne formelen er som den er, så kan det lønne seg å skjønne hvorfor de kalles trekanttall. Om vi tegner opp summene som antall baller, og tenger først 1, så 2, og så 3 bortover, sånn som dette:



Så ser vi at de ulike summene blir trekanter. Vi kan så gjøre om en slik trekant til en firkant ved å legge på like mange nye. Sånn som vist på høyre side av figuren. Denne firkanten har n baller i høyden, og $n+1$ baller bortover. Da er antall baller i hele firkanten $n(n+1)$. Men vi har jo doblet antall baller for å få en firkant, så om vi bare skal telle de grønne ballene må vi dele på to.

---

g) Hvordan kan kan du sjekke om et tall er et trekanttall? Skriv et program som sjekker om et heltall er et trekanttall, og hvis ja, skriver ut faktorene i dette. For eksempel bør programmet skrive ut `1 + 2 + 3 + 4 + 5` hvis det blir gitt input `15`.

Løsning oppgave 12 *Trekanttall*

a) Dette gjør vi med en `for`-løkke, og `range`-funksjonen:

```
1 for tall in range(1, 6):
2     print(tall)
```

Husk at `range` er fra-og-med, til (men ikke med), derfor skriver vi 1-6, for å få tallene 1, 2, 3, 4, og 5.

b)

```
total = 0

for tall in range(1, 6):
    total += tall

print(total)
```

Her må vi både opprette `total` før løkka, og printe den ut etter løkka. Vi ser hvilke kodelinjer som hører til løkka fordi de har fått innrykk. I tilegg har vi lagt til blanke linjer for å skille dem litta fra løkka, men merk at dette er frivillig.

c) Svaret blir 15, som forventet.

d) Vi endrer programmet ved å endre hva løkka går til. For å gå opp til og med hundre må vi skrive `range(1, 101)`. For å gjøre programmet vårt lettere å endre velger vi derimot å lage n som en variabel:

```
n = 100

total = 0
for tall in range(1, n+1):
    total += tall

print(total)
```

Det er nå rett-frem å endre programmet, bare ved å endre den første linja.

Svaret blir 5050

e) Vi fikk 5050 for $n = 100$, formelen gir

$$\frac{n(n+1)}{2} = \frac{100 \cdot 101}{2} = 5050.$$

Som altså er det samme, programmet vårt ser ut til å fungere.

f) For $n = 1000$ gir programmet vårt oss 500500. Formelen gir oss det samme. For $n = 1000000$ gir programmet vårt oss 500000500000, og formelen gir igjen det samme.

4 Funksjoner

Oppgave 13 *Enkle funksjoner*

a) Lag en funksjon `kvadrat(x)` som tar inn et tall x og returnerer kvadratet x^2.

b) Gjenta oppgaven over, men med funksjonen `kubikk(x)` som returnerer x^3.

c) Lag en funksjon `f(x)`, som returner $x^2 + 3x - 1$

Løsning oppgave 13 *Enkle funksjoner*

```
1 def kvadrat(x):
2     return x**2
3
4 def kubikk(x):
5     return x**3
6
7 def f(x):
8     return x**2 + 3*x -1
```

Oppgave 14 *Annengradsfunksjon*

En annengradsfunksjon er definert slik:

$$f(x) = x^2 + 0.3x - 1$$

a) Definer funksjonen som en Python-funksjon, `f(x)`, som tar inn en x-verdi og returnerer tilhørende y-verdi.

b) Test funksjonen din ved å kalle på den med følgende x verdier og skriv ut resultatet til terminalen:

1. $x = 0$

2. $x = 1$
3. $x = -0.4$

Løsning oppgave 14 *Annengradsfunksjon*

a)

```
def f(x):
  return x**2 + 0.3*x - 1
```

b)

1.

```
x = 0
y = f(x)
print(f'x={x}, f(x)={y}')
```

```
x=0, f(x)=-1.0
```

2.

```
x = 1
y = f(x)
print(f'x={x}, f(x)={y}')
```

```
x=1, f(x)=0.30000000000000004
```

3.

```
x = -0.4
y = f(x)
print(f'x={x}, f(x)={y}')
```

```
x=-0.4,  f(x)=-0.96
```

Oppgave 15 *Midtpunktfunksjon*

Midtpunktet mellom to tall, a og b er gitt ved:

$$\frac{a+b}{2}$$

a) Lag en funksjon som tar inn to tall, `a` og `b` og returnerer midtpunktet mellom dem

b) Bruk funksjonen til å finne midtpunktet mellom 34 og 86

Løsning oppgave 15 *Midtpunktfunksjon*

a)

```
1  def midtpunkt(a, b):
2      return (a + b)/2
```

b)

```
1  a = 34
2  b = 86
3
4  m = midtpunkt(a, b)
5  print(f'Midpunktet mellom {a} og {b} er {m}')
```

Oppgave 16 *Kanonkule*

Året er 1384 i en fiktiv borg i Sør-England. Franskmennene har omringet borgen og kanonene er rullet frem. Dette har åpenbart gjort Lorden som eier

borgen nervøs, og han spør deg, hans fremste lærdmann om kanonkulene vil nå borgen med den avstanden de har.

Heldigvis har spionene til kongen klart å stjele en av kanonene, og etter mye om og men, har du funnet ut at kulene forlater kanonen med en fart på ca 50 meter i sekundet. Kanonene er låst i en 30 graders vinkel, så posisjonen til en kanonkule kan beskrives med disse funksjonene

$$x(t) = 25\frac{m}{s}t$$
$$y(t) = 45\frac{m}{s}t - 5\frac{m}{s^2}t^2,$$

hvor $x(t)$ er hvor langt kanonkula har bevegd seg bortover (i meter) t sekund etter at den ble skutt og $y(t)$ er høyden til kanonkula (i meter) t sekund etter at den ble skutt.

a) Definer en funksjon `kanonkule_x(t)` som returnerer hvor langt kanonkula har bevegd seg bortover etter `t` sekund.

b) Definer en funksjon `kanonkule_y(t)` som returnerer høyden til kanonkula etter `t` sekund.

c) Definer en funksjon `kanonkule_posisjon(t)` som returnerer både hvor langt kula har bevegd seg bortover og høyden til den etter `t` sekund.

d) Kanonene er plassert 200 meter unna borgen (i x retning), og er i samme høyde som borgen. Bruk funksjonen vi akkurat lagde og prøv ut forskjellige verdier for `t`. Treffer kanonkula borgen?

e) (Bonusoppgave) Lag et program som bruker `input` funksjonen til å be om et antall sekunder og printer ut posisjonen til kanonkula så lang tid etter at den ble skutt.

f) (Bonusoppgave) Bruk en løkke til å printe ut posisjonen til kanonkula de ti første sekundene kanonkula er i lufta.

Løsning oppgave 16 *Kanonkule*

a)

```
def kanonkule_x(t):
    return 25*t
```

b)

```
def kanonkule_y(t):
    return 45*t - 5*t**2
```

c)

```
def kanonkule_posisjon(t):
    x = kanonkule_x(t)
    y = kanonkule_y(t)
    return x,y
```

d)

```
tid = 10
x, y = kanonkule_posisjon(tid)
print(f"lengde: {x}m  høyde:{y}m")
```

e)

```
tid = float(input("Skriv inn antall sekunder etter
    avfyrt skudd:"))
x, y = kanonkule_posisjon(tid)
print(f"lengde: {x}m  høyde:{y}m")
```

f)

```
print("Posisjon de første 10 sekundene:")
for tid in range(0,11):
    x,y = kanonkule_posisjon(tid)
    print(f"tid:{tid:4.1f}s lengde:{x:3.0f}m  hø
       yde:{y:3.0f}m")
```

```
Posisjon de første 10 sekundene:
tid: 0.0s lengde:  0m  høyde:  0m
tid: 1.0s lengde: 25m  høyde: 40m
```

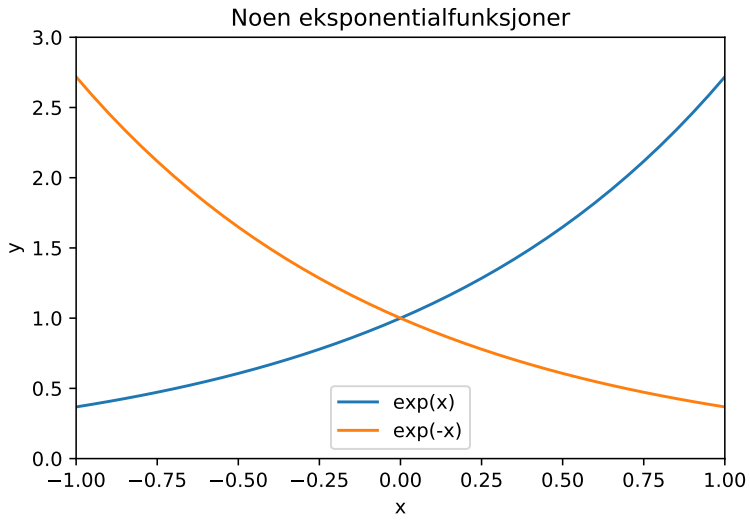
```
tid: 2.0s lengde: 50m  høyde: 70m
tid: 3.0s lengde: 75m  høyde: 90m
tid: 4.0s lengde:100m  høyde:100m
tid: 5.0s lengde:125m  høyde:100m
tid: 6.0s lengde:150m  høyde: 90m
tid: 7.0s lengde:175m  høyde: 70m
tid: 8.0s lengde:200m  høyde: 40m
tid: 9.0s lengde:225m  høyde:  0m
tid:10.0s lengde:250m  høyde:-50m
```

Her ser vi at når kula er framme ved borgen (200m) er den 40 meter over bakken. Den treffer bakken ved 225m, altså 25m bortenforborgmuren, altså treffer den borgen. (med mindre det er en veldig liten borg)

5 Plotting

Oppgave 17 *Plotting*

I denne oppgaven skal vi ende opp med et plot som ser slik ut:



a) Opprett en array med tallrekken som starter på 0, slutter på 1 og har et mellomrom på 0.05 mellom hvert element. Lagre den arrayen i en variabel du kaller `x`.

b) Opprett en array som inneholder e^x for alle verdier i `x` variabelen din (hint: husk `exp` funksjonen i `pylab`) og lagre dette arrayet i en array med navnet `y1`.

c) Lag et plot som viser x på førsteaksen og e^x på andreaksen for x-verdier mellom 0 og 1.

d) Pynt på plottet ved å sette grenser for x-aksen og y-aksen med `xlim` og `ylim` funksjonene. Sjekk at aksene dine har endret seg siden oppgave **b)**.

e) Gi aksene dine merkelapper (f.eks x og y) med `xlabel` og `ylabel` funksjonene.

f) Gi plottet ditt en tittel (f.eks $y = exp(x)$) med `title` funksjonen.

g) Opprett en ny variabel, `y2` og som inneholder e^{-x} for alle verdier i `x` arrayet ditt.

h) Plot $y1$ og $y2$ i samme plot.

i) (Bonusoppgave) Gi kurvene dine merkelapper som forteller hva de representerer og vis frem disse merkelappene med `legend` funksjonen.

Løsning oppgave 17 *Plotting*

a)

```
from pylab import arange

x = arange(0,1,0.05)
```

b)

```
from pylab import arange, exp

x = arange(0,1,0.05)
y1 = exp(x)
```

c)

```
from pylab import arange, exp, plot, show

x = arange(-1,1.05,0.05)
y1 = exp(x)
plot(x,y1)
show()
```

d)

```
from pylab import arange, exp, plot, show, xlim,
   ylim

```

```
x = arange(-1,1.05,0.05)
y1 = exp(x)
plot(x,y1)
xlim(-1,1)
ylim(0,3)
show()
```

e)

```
from pylab import arange, exp, plot, show, xlim,
    ylim, xlabel, ylabel

x = arange(-1,1.05,0.05)
y1 = exp(x)
plot(x,y1)
xlim(-1,1)
ylim(0,3)
xlabel('x')
ylabel('y')
show()
```

f)

```
from pylab import arange, exp, plot, show, xlim,
    ylim, xlabel, ylabel, title

x = arange(-1,1.05,0.05)
y1 = exp(x)
plot(x,y1)
xlim(-1,1)
ylim(0,3)
xlabel('x')
ylabel('y')
title('$y=e^x$')
show()
```

g)

```
from pylab import arange, exp, plot, show, xlim,
    ylim, xlabel, ylabel, title
```

```
2
3  x = arange(-1,1.05,0.05)
4  y1 = exp(x)
5  y2 = exp(-x)
6  plot(x,y1)
7  xlim(-1,1)
8  ylim(0,3)
9  xlabel('x')
10 ylabel('y')
11 title('$y=e^x$')
```

h)

```
1  from pylab import arange, exp, plot, show, xlim,
      ylim, xlabel, ylabel, title
2
3  x = arange(-1,1.05,0.05)
4  y1 = exp(x)
5  y2 = exp(-x)
6  plot(x,y1)
7  plot(x,y2)
8  xlim(-1,1)
9  ylim(0,3)
10 xlabel('x')
11 ylabel('y')
12 title('$y=e^x$ og $y=e^{-x}$')
```

i)

```
1  from pylab import arange, exp, plot, show, xlim,
      ylim, xlabel, ylabel, title, legend
2
3  x = arange(-1,1.05,0.05)
4  y1 = exp(x)
5  y2 = exp(-x)
6  plot(x,y1, label='$y=e^x$')
7  plot(x,y2, label='$y=e^{-x}$')
8  xlim(-1,1)
9  ylim(0,3)
10 xlabel('x')
```

```
11 ylabel('y')
12 title('$y=e^x$ og $y=e^{-x}$')
13 legend()
14 show()
```

Oppgave 18 *Plotte en annengradsfunksjon*

a) Definer en funksjon for den matematiske funksjonen

$$f(x) = x^2 - 5x + 9$$

b) Bruk funksjonen `linspace` for å generere x-verdier mellom 0 og 5. Lagre disse i en `array`. Finn tilsvarende y-verdier ved å sende x-verdiene inn som et argument (du kan sende med hele arrayet som ett argument). Skriv ut begge listene.

c) Plot $f(x)$ mellom $x = 0$ og $x = 5$. Prøv å endre antall x-verdier du valgte i (b) og se hvordan det endrer plottet (du kan ta bort print-delen her hvis du vil).

d) Legg til en tittel til plottet og sett navn på aksene.

Oppgave 19 *Grafisk løsning av likning*

I denne oppgaven skal vi løse en likning *grafiks*, dvs. ved å lage og lese av en graf.

a) Definer en funksjon som returnerer $f(x) = sin(x^2)$.

b) Definer en funksjon som returnerer $g(x) = x^2 + x/5 - exp(-x)$.

c) Opprett arrays for å lagre x-verdier mellom + og 1.7, samt verdier man får ved å kalle på f og g med disse x-verdiene.

d) Tegn grafen til $f(x)$ for x-verdier mellom 0 og 1.7.

e) Tegn grafen til $g(x)$ for x-verdier mellom 0 og 1.7 i samme plot som $f(x)$.

f) For hvilken x er $f(x)$ og $g(x)$ like (ca)?

g) Marker punktet hvor $f(x)$ og $g(x)$ er like med en rød sirkel.

Løsning oppgave 19 *Grafisk løsning av likning*

a)

```
from pylab import *

def f(x):
    return sin(x**2)
```

b)

```
def g(x):
    return x**2 + x/5 - exp(-x)
```

c)

```
x = arange(0, 1.7 + 0.05, 0.05)
```

d)

```
y1 = f(x)
plot(x, y1)
show()
```

e)

```
y2 = g(x)
plot(x, y1)
plot(x, y2)
show()
```

f) Vi leser av grafen at funksjonene er (omtrent) like for 1

g)

```
plot(x, y1)
plot(x, y2)
plot(1, g(1), 'ro')
show()
```

Oppgave 20 *Fortsettelse kanonkule*

Fortsettelse fra kanonkuleoppgaven. Vi har akkurat fortalt lorden i borgen vår at kanonene kan treffe og han tror ikke på oss. Derfor bestemmer vi oss for å lage en tegning hvor vi viser kulas bane. Om du ikke fikk til forrige kanonkuleoppgave kan du bruke denne funksjonen:

```
def kanonkule_posisjon(t):
    x = 25*t               # kanonkule_x(t)
    y = 45*t - 5*t**2      # kanonkule_y(t)

    return x, y
```

a) Lag en array, `t` som inneholder en tallrekke som starter på 0, slutter på 9 og har en steglenge på 0.1 (Tips: hvis du vil at tallrekka skal slutte på 9 er det lurt å sette sluttposisjonen til `slutt+steglengde`, som i dette tilfellet er 9.1).

b) Bruk `kanonkule_posisjon(t)` til å regne ut kanonkulas `x`-koordinater og `y`-koordinater for hvert tidspunkt i `t` array-et vårt.

c) Lag et plot som viser `x`-posisjon på førsteaksen on `y`-posisjon på andre-aksen.

d) Ved å løse likningen $x(t) = 200$ ser vi at kanonkula treffer borgen etter åtte sekund. Marker punktet $(x(8), y(8))$ med et rødt punkt.

Løsning oppgave 20 *Fortsettelse kanonkule*

a)

```
from pylab import *
t = arange(0,9.1,0.1)
```

b)

```
x_pos, y_pos = kanonkule_posisjon(t) #får to
   arrays med x og y posisjoner
```

c)

```
plot(x_pos,y_pos) #plott banen
```

d)

```
treff_x, treff_y = kanonkule_posisjon(8)
#finn treffpunktet ved 8 sek
plot(treff_x,treff_y,'ro')
#plott punktet, 'ro' for å få et rødt punkt
show()
```