Oppgave 1: Eksempelsett «Vår 2021» Oppgave 4 (forutsatt heldigital eksamen)

Nedenfor ser du en algoritme som styrer en penn i et koordinatsystem.

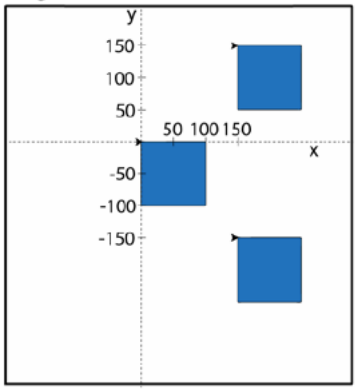
Pennen går fremover i den retningen som pilen peker.

- Start på (0, 0)
- Gjenta 4 ganger:
  - Gå 100 frem
  - Snu 90 grader til høyre
- Flytt til (150, 150)
- Gjenta 4 ganger:
  - Gå 100 frem
  - Snu 90 grader til høyre

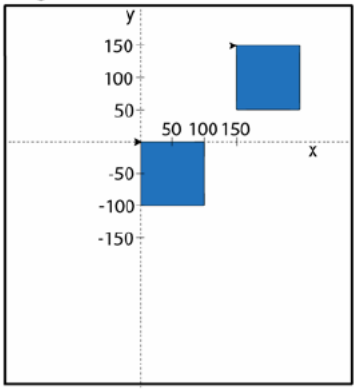
Hvilken figur stemmer med algoritmen?

Figur Skriv svaret her

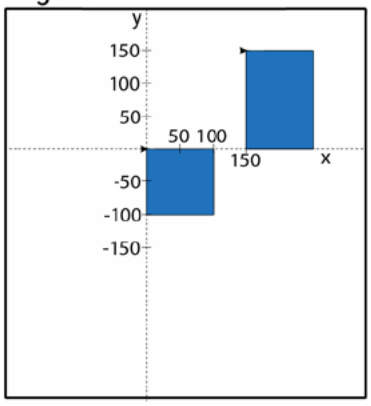
Figur 1



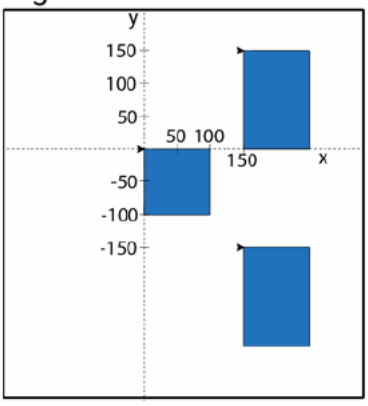
Figur 2



Figur 3



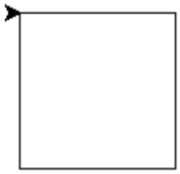
Figur 4

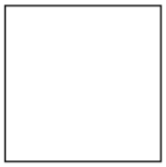


Løsning:

```
from turtle import*

for i in range(4):
    forward(100)
    right(90)
penup()
goto(150, 150)
pendown()
for i in range(4):
    forward(100)
    right(90)
```

Resultat:



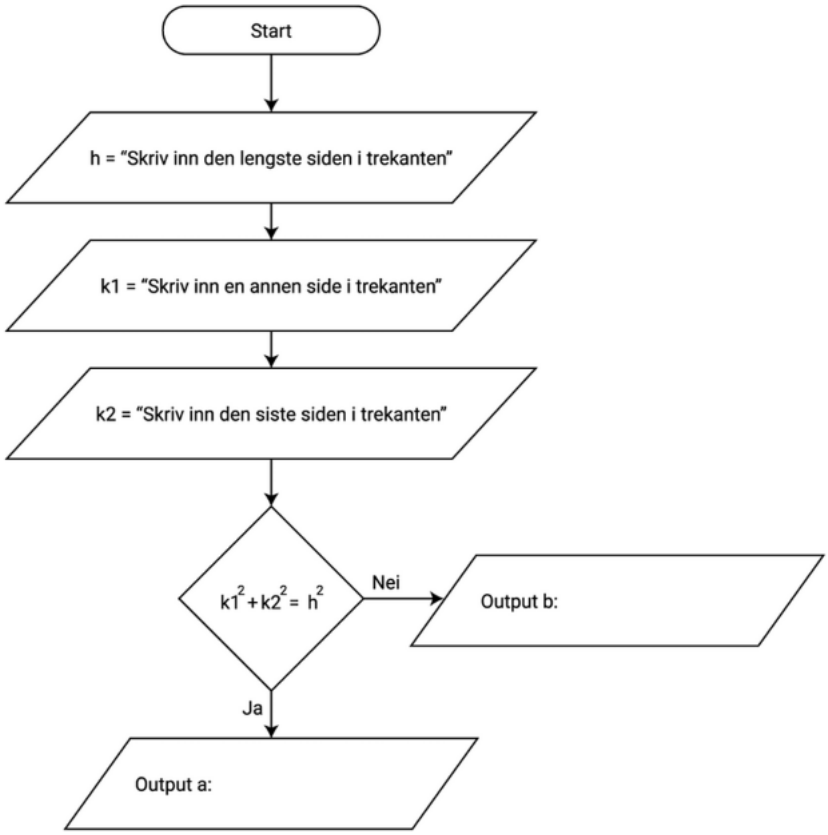


Oppgaven går ut på at elevene skal tolke en algoritme som styrer en penn, og resultatet skal bli en figur. Oppgaven er formulert som en flervalgsoppgave hvor elevene skal velge den figuren som passer best med algoritmen, og var opprinnelig tiltenkt en planlagt og varslet heldigital eksamen, men denne eksamensformen er blitt skrinlagt inntil videre. Oppgaven har en del svakheter og upresise formuleringer, for eksempel «Pennen går fremover i den retningen som pilen peker». For det første er det ikke bare én pil, men flere piler som alle peker mot høyre på figurene, men pennen går definitivt ikke bare mot høyre. Det er også uklart hvordan aksene, verdiene langs aksene, navn på aksene og det blå fyllet blir til.

Løsningen benytter på sin side Turtle-modulen i Python, og denne relativt enkle løsningen er tilstrekkelig til å identifisere riktig svar, selv om resultatet kun består av omrissene. Det er lagt til at pennen må løftes ved forflytning mellom figurene – det mangler i algoritmen/oppgaven. Det kan til slutt nevnes at Turtle-modulen kan være ypperlig for innlæring av Python, da den tilbyr en appellerende visualisering av programmeringskonseptene.

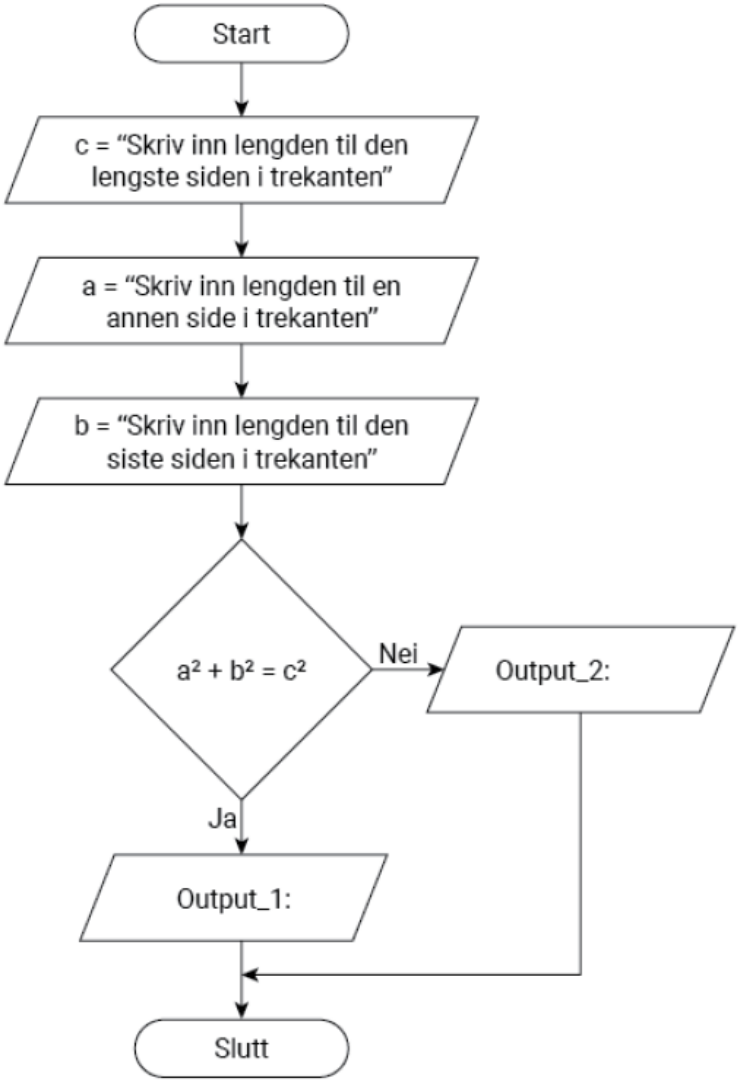
Oppgave 2: Eksempelsett «Vår 2021» Oppgave 7 (øverst), nytt sett datert 18.08.2021 Oppgave 5 (øverst), og nytt sett datert 14.01.2022 Oppgave 5 (nederst)

Bildet viser en algoritme som kan programmeres.



Vurder og kommuniser hva algoritmen undersøker.
Gi eksempler på *output* når *h*, *k1* og *k2* får forskjellige verdier.

Bildet viser en algoritme som kan programmeres.



Forklar hva algoritmen undersøker.
Gi eksempler på tallverdier for *a*, *b* og *c* som gir Output_1

[På figuren over mangler det punktum til slutt i setningen. Inkonsekvens.]

Løsning:

```
h = float(input(f'Skriv inn den lengste siden i trekanten'))
k1 = float(input(f'Skriv inn en annen side i trekanten'))
k2 = float(input(f'Skriv inn den siste siden i trekanten'))

if k1**2 + k2**2 == h**2:
    print('Trekanten er rettvinklet')
else:
    print('Trekanten er ikke rettvinklet')
```

Resultat 1:

```
Skriv inn den lengste siden i trekanten5
Skriv inn en annen side i trekanten4
Skriv inn den siste siden i trekanten3
Trekanten er rettvinklet
```

Resultat 2:

```
Skriv inn den lengste siden i trekanten6
Skriv inn en annen side i trekanten5
Skriv inn den siste siden i trekanten4
Trekanten er ikke rettvinklet
```

Oppgaven kommer i et par ulike varianter, men det de har til felles, er at elevene skal vurdere et flytskjema, i oppgaven noe upresist kalt *algoritme*, som vi assosierer med en sjekk om hvorvidt en trekant er rettvinklet eller ikke. I flytskjemaet bes brukeren om å legge inn lengden av den lengste siden først, og de to andre etterpå. Deretter kommer det en sjekk om hvorvidt Pytagoras' setning er oppfylt eller ikke. Flytskjemaet mangler bevisst hva som skal være output i de to tilfellene. Elevene bes om å forklare hva algoritmen undersøker, og om å gi eksempler på lengder som vil gi ulike outputer.

Løsningen er at vi simpelthen lager et skript som samsvarer med og utfyller flytskjemaet. Vi legger inn forslag til hva som kan være output i de to tilfellene, nemlig om trekanten er rettvinklet eller ikke. For å demonstrere ulike outputer setter vi inn tallverdier som gir rettvinklet trekant, og tallverdier som ikke gjør det.

Denne oppgaven kan ellers være et godt utgangspunkt for det å utfordre elevene til å gi et skript nye egenskaper. For eksempel kan man be elevene tilpasse skriptet slik at man ikke nødvendigvis må legge inn den lengste siden først, men at det skal ta kunne tallverdier for tre sider i vilkårlig rekkefølge og undersøke om tallverdiene svarer til en rettvinklet trekant.

Oppgave 3: Eksempelsett datert 18.08.2021 Oppgave 4 (øverst), og nytt sett datert 14.01.2022 Oppgave 4 (nederst)

Vi har likningen $(4 - a)(4 + b) = 8$.

Det finnes flere tallverdier for *a* og *b* som gjør at denne likningen stemmer.

Finn tre løsninger til likningen.

Vi har likningen (4 – a) (4 + b) = 8

Det finnes flere tallpar som gjør at denne likningen blir gyldig.

Gi eksempler på tre slike tallpar.

Løsning:	Resultat:

`for a in range(-100, 100):`
`  for b in range(-100, 100):`
`  if (4 - a)*(4 + b) == 8:`
`  print(a, b)`
	-4 -3
	0 -2
	2 0
	3 4
	5 -12
	6 -8
	8 -6
	12 -5

I denne oppgaven, som kommer i to varianter med litt ulik ordlyd, blir elevene presentert for en likning med to ukjente, $(4 - a)(4 + b) = 8$, og skal finne tre løsninger til likningen. I utgangspunktet ser ikke denne oppgaven ut til å være ment for programmering, men metodefriheten gjør at vi like gjerne kan bruke programmering som noe annet. Man kan i det hele tatt se for seg mange tilnærminger til en løsning.

Løsningen i Python benytter to for-løkker skrevet inn i hverandre, og vi kan sjekke hvilke intervaller for variablene a og b som vi selv ønsker. Vi benytter altså en numerisk tilnærming i stedet for en analytisk. På grunn av at beregningene her går relativt raskt, kan vi like gjerne velge intervallet $[-100, 100]$ for dem begge, selv om det kanskje virker litt overdrevent. Resultatet blir hele 8 løsninger, som vi kan kontrollere gyldigheten av i etterkant.

Oppgave 4: Eksempelsett datert 18.08.2021 Oppgave 7

Siden 2018 har pant på plastflasker vært 2 kr for små flasker og 3 kr for store flasker.

Ali har pantet flasker for 109 kr.

Til sammen pantet han 51 flasker.

Hvor mange små og store plastflasker pantet Ali?

Sett opp, forklar og løs et likningssett som beskriver den praktiske situasjonen.

Løsning:

```
for smaa in range(51):
    for store in range(51):
        if 2*smaa + 3*store == 109 and smaa + store == 51:
            print(f'Antall små plastflasker er {smaa}')
            print(f'Antall store plastflasker er {store}')
```

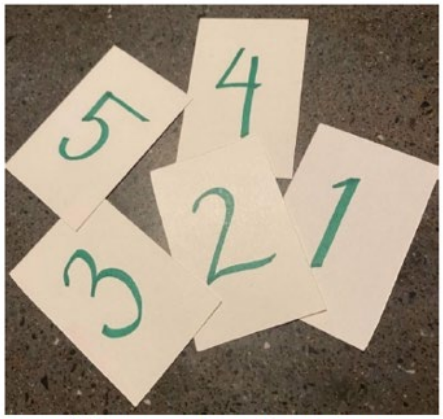
Resultat:

```
Antall små plastflasker er 44
Antall store plastflasker er 7
```

I denne oppgaven skal elevene sette opp et likningssett basert på panting av flasker. De får opplyst at Ali har pantet 51 flasker for 109 kroner, og de skal finne ut hvor mange små og store flasker Ali har pantet når panten på flaskene er hhv. 2 kroner og 3 kroner. Oppgaven er trolig ikke ment for programmering i utgangspunktet, da elevene blir bedt om å sette opp, forklare og løse et likningssett som metode.

Løsningen går ut på at man med et enkelt program som benytter likningssettet, kan sjekke alle muligheter for intervallet $[0, 51]$ for begge flaskestørrelser. Her benytter vi også en numerisk tilnærming fremfor en analytisk. I en fullverdig løsning må man selvfølgelig forklare hvorfor likningssettet blir som det blir.

Oppgave 5: Eksamen 24.05.2022 Oppgave 3



På fem kort står tallene 1, 2, 3, 4 og 5.
Kortene blir lagt på et bord med tallsiden ned og blir deretter blandet.

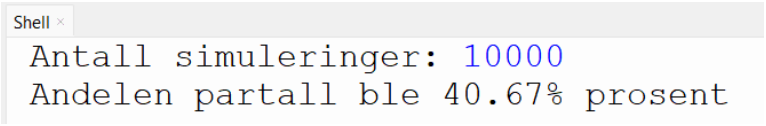
Du trekker to tilfeldige kort og summerer tallene som står på kortene.

Argumenter for at det er 40 % sjanse for at summen av tallene på kortene du trekker blir et partall.

Løsning:


```python
'''Importerer funksjonen randint
fra random-biblioteket for å
generere tilfeldige heltall.'''
from random import randint

'''Definerer antall trekninger
(simuleringer) vi ønsker å utføre.'''
antall_trekninger = int(input('Antall simuleringer: '))

'''Initialiserer en tellevariabel for
å holde styr på hvor mange av
summene som blir partall.'''
antall_partall = 0

'''For-løkke som kjører riktig antall
ganger (én gang for hver simulering).'''
for i in range(antall_trekninger):

    '''Oppretter en liste med tallene 1, 2, 3, 4 og 5.
    Dette representerer lappene vi kan trekke fra.'''
    lapper = [1, 2, 3, 4, 5]

    '''Trekker et tilfeldig tall fra indekser
    0 til 4 (første trekning).'''
    trekk1 = randint(0, 4)

    '''Henter tallet fra listen basert på
    den tilfeldige indeksen.'''
    lapp1 = lapper[trekk1]

    '''Fjerner det trukne tallet fra listen
    for å sikre at det ikke kan trekkes igjen.'''
    lapper.pop(trekk1)

    '''Trekker et nytt tilfeldig tall fra
    indekser 0 til 3 (andre trekning).'''
    trekk2 = randint(0, 3)

    '''Henter det andre tallet fra den oppdaterte listen.'''
    lapp2 = lapper[trekk2]

    '''Beregner summen av de to trukne tallene.'''
    summen = lapp1 + lapp2

    '''Sjekker om summen er et partall.
    Hvis summen er partall (heltall som
    kan deles på 2 uten rest), øker vi telleren'''
    if summen % 2 == 0:
        antall_partall += 1

'''Regner ut andelen partall ved å
dele antall partall på antall trekninger,
og deretter multiplisere med 100 for å
få prosentandel.'''
andel_partall = antall_partall / antall_trekninger

'''Skriver ut resultatet med andelen partall i prosent.'''
print(f'Andelen partall ble {andel_partall:0.2%} prosent')
```

Resultat:

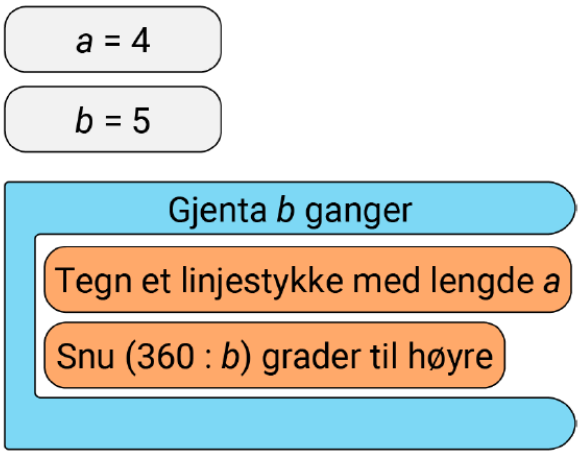


Oppgaven går ut på at blant fem kort som er nummerert 1–5, trekkes to kort tilfeldig. De fem kortene snus og blandes før trekkingen, og tallene på kortene som trekkes, skal summeres. Elevene skal argumentere for at sannsynligheten er 40 % for at summen av de to trukne tallene er et partall. Denne oppgaven er trolig ikke ment for programmering i utgangspunktet, men kan likevel brukes til å illustrere forskjellene på analytisk og numerisk tilnærming. Det som er den mest nærliggende tilnærmingen uten bruk av programmering, er kanskje å telle antall gunstige utfall og dele på mulige utfall, eller å benytte en kombinasjon av produktsetningen og addisjonssetningen. Her kan man se for seg mange ulike måter å argumentere på.

Dersom man skal løse oppgaven med Python, kreves det imidlertid et litt større skript hvor man simulerer situasjonen et visst antall ganger. I et forsøk på å øke lesbarheten til skriptet er det i denne programkoden skrevet inn forklaringer, men antall kodelinjer i denne løsningen er strengt tatt bare 15. Dersom man setter inn en høyere og høyere tallverdi for antall simuleringer, vil man se at andelen partall nærmer seg 40 %, noe som må være en så god argumentasjon som noen. Man kunne eventuelt utvidet skriptet til også å kunne fremstille andelen partall som en funksjon av antall simuleringer grafisk.

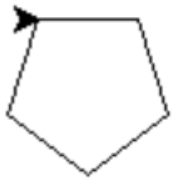
Oppgave 6: Eksamen 24.05.2022 Oppgave 8

Bildet viser et dataprogram.



a) **Forklar hva som skjer når programmet blir kjørt.**

b) **Tegn figuren og sett riktige mål på figuren din.**

Løsning:	Resultat:
1 `from turtle import*` 2 3 `a = 40` 4 `b = 5` 5 6 `for i in range(b):` 7 `    forward(40)` 8 `    right(360/b)`	

I oppgaven blir elevene presentert for en figur som viser en blokkbasert sekvens av instruksjoner, noe upresist kalt *dataprogram*. Variablene *a* og *b* settes til hhv. 4 og 5, før det vi gjenkjenner som en løkke, skal gjenta seg *b* ganger. I løkka skal det først tegnes et linjestykke med lengde *a*, og deretter kommer «snu» 360/b grader til høyre. Oppgaven ber elevene forklare hva som skjer når programmet blir kjørt, og deretter tegne figuren som programmet lager, samt sette riktige mål på figuren.

Det som imidlertid er en forspilt sjanse for eksamensnemnda her, er jo de utforskende spørsmålene oppgaven åpner for! Hva skjer hvis vi endrer på variablene *a* og *b*? Hvorfor gir 360/*b* en lukket figur? Men selv om slike spørsmål ikke ble stilt på eksamen, kan det ha didaktisk verdi i å se på det sammen med elevene.

Oppgaven kan enkelt løses ved hjelp av Turtle-modulen, da oppgaveformuleringen passer godt med syntaksen i Turtle. Men siden bevegelse i Turtle baserer seg på antall datapunkter, må vi bare øke verdien av *a* til 40 for å få en godt nok synlig figur. Vi mangler det å sette målene på figuren her.

Oppgave 7: Eksamen 22.05.2023 Oppgave 5

Emira utforsker store talls lov ved å kaste terning med seks sider.
Hun lager et dataprogram som kaster terning for henne.

Nedenfor vises Emiras forslag til en kode til et dataprogram.

```
antall_terningkast = skriv inn heltall

Gjenta antall_terningkast ganger
    tall = tilfeldig tall fra og med 1 og til og med 6
    legg tall til liste

Skriv liste til skjerm
```

a) Forklar hva som skjer når dataprogrammet blir kjørt

Emira vil lage en tabell for å vise at det er like stor sannsynlighet for å få de ulike resultatene 1, 2, 3, 4, 5 og 6.

b) Hvilken verdi for *antall_terningkast* vil du anbefale Emira å velge? Begrunn svaret ditt.

Løsning:

```
from random import randint
import matplotlib.pyplot as plt

antall_terningkast = int(input('Skriv inn et heltall: '))
en = 0
to = 0
tre = 0
fire = 0
fem = 0
seks = 0

for i in range(antall_terningkast):
    tall = randint(1, 6)
    if tall == 1:
         en += 1
    elif tall == 2:
        to += 1
    elif tall == 3:
        tre += 1
    elif tall == 4:
        fire += 1
    elif tall == 5:
        fem += 1
    elif tall == 6:
        seks += 1

print(f'''
Antall øyne:\t{1:5}{2:5}{3:5}{4:5}{5:5}{6:5}
Frekvens:\t{en:5}{to:5}{tre:5}{fire:5}{fem:5}{seks:5}
''')
```

Resultat 1:

```
Skriv inn et heltall: 10

Antall øyne:        1    2    3    4    5    6
Frekvens:           1    2    0    2    1    4
```

Resultat 2:

```
Skriv inn et heltall: 1000

Antall øyne:        1    2    3    4    5    6
Frekvens:         157  177  179  142  164  181
```

Resultat 3:

```
Skriv inn et heltall: 100000

Antall øyne:        1     2     3     4     5     6
Frekvens:       16830 16655 16597 16603 16630 16685
```

I denne oppgaven skal elevene utforske store talls lov ved å simulere kast med en terning. Programmet lar brukeren angi et tall for hvor mange ganger terningen skal kastes, og for hvert kast blir et tilfeldig tall mellom 1 og 6 valgt og lagt til en liste. Til slutt skrives hele listen ut i konsollen, uten mer bearbeidelse. Altså vil det komme en sekvens med tallene 1–6 som blir akkurat så lang som brukeren legger inn. I hvilken grad en slik tallsekvens vil kunne si noe om sannsynligheten, er høyst uklart.

Elevene på sin side blir bedt om å forklare «hva som skjer når programmet kjøres», og å velge en passende verdi for antall terningkast for å demonstrere at sannsynligheten for å få 1, 2, 3, 4, 5 og 6 øyne er omtrent like stor. Hva som er en passende, eller *anbefalt*, verdi i denne sammenhengen, kan være gjenstand for diskusjon, da det største problemet er at programmet ikke er egnet til å si noe om dette i utgangspunktet. Etter min mening hadde denne oppgaven fungert bedre som en «*Modifisere»*-oppgave, der elevene hadde kunnet utforske, utvide og/eller modifisere algoritmen til å bli funksjonell etter formålet.

Det er heller ikke helt presist å kalle innholdet i figuren «*forslag til kode til et dataprogram*», fordi det simpelthen ikke *er* kode, men heller en beskrivelse av hva programmet skal gjøre ved hjelp av vanlig tekst og blokker. Dette er mer en algoritmebeskrivelse eller pseudokode, som gir en stegvis forklaring av hvordan programmet skal fungere, og ikke er skrevet i et spesifikt programmeringsspråk.

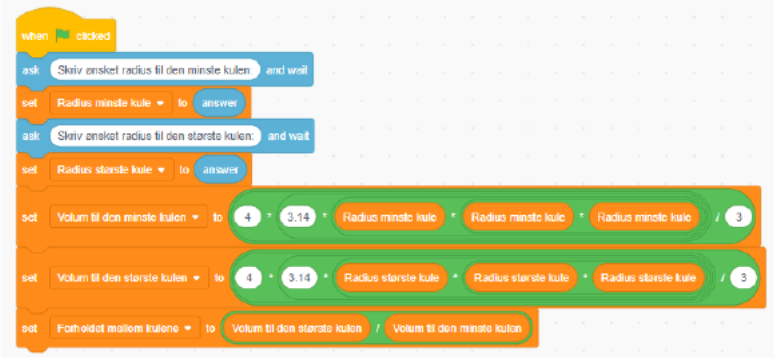
Den valgte Python-løsningen har en mer tradisjonell tilnærming. I stedet for bare å printe en lang liste med tilfeldige tall til konsollen organiserer den resultatene i variabler, slik at elevene kan sammenlikne antallet for de ulike utfallene. De ulike resultatene åpner for diskusjoner rundt absolutte og relative frekvenser, og utvikling i forskjeller mellom de ulike variablene. Man kan også diskutere ulike versjoner av skriptet, om man kan komprimere det og eventuelt utvide det til også å lage grafiske fremstillinger.

Oppgave 8: Eksamen 21.05.2024 Oppgave 6

Guro utforsker volumet til kuler og lager koden nedenfor.

```
1   radius_liten_kule = float(input("skriv ønsket radius for minste kule: "))
2   radius_stor_kule = float(input("skriv ønsket radius for største kule: "))
3
4   volum_liten_kule = (4 * 3.14 * radius_liten_kule ** 3)/3
5   volum_stor_kule = (4 * 3.14 * radius_stor_kule ** 3)/3
6
7   forhold_mellom_kulene = volum_stor_kule/volum_liten_kule
8
9   print("forholdet mellom kulene er: ", forhold_mellom_kulene)
```

Koden som tekst



Koden som blokk

a) Forklar hva som skjer når koden kjøres.

Guro tester koden med å skrive inn en radius til to kuler og får dette resultatet:

```
Skriv ønsket radius til den minste kulen:  2
Skriv ønsket radius til den største kulen:  4
Forholdet mellom kulene er:  8.0
```

b) Forklar hvordan volumet til en kule endres, når radius dobles.

Løsning:

```
radius_liten_kule = float(input("skriv inn ønsket radius for minste kule: "))
radius_stor_kule = float(input("skriv inn ønsket radius for største kule: "))

volum_liten_kule = (4 * 3.14 * radius_liten_kule ** 3)/3
volum_stor_kule = (4 * 3.14 * radius_stor_kule ** 3)/3

forhold_mellom_kulene = volum_stor_kule/volum_liten_kule

print("forholdet mellom kulene er: ", forhold_mellom_kulene)
```

Resultat 1:

```
skriv inn ønsket radius for minste kule: 2
skriv inn ønsket radius for største kule: 4
forholdet mellom kulene er:  8.0
```

Resultat 2:

```
skriv inn ønsket radius for minste kule: 50
skriv inn ønsket radius for største kule: 100
forholdet mellom kulene er:  8.0
```

Dette er altså den første gangen elevene møtte en oppgave som inkluderte Python-kode. I oppgaven hadde elevene imidlertid tilgang på koden i Python og som blokker. Det forelå også resultat fra en kjøring av programmet. I koden tas det to verdier fra brukeren, radius for en «liten» kule, og radius for en «stor» kule. Deretter beregner koden volumet til de to kulene, og bestemmer forholdet mellom volumet til den store og den lille. Dette forholdet skrives ut, og vi ser at dersom man skriver inn radius 2 for den lille og 4 for den store, får man at forholdet blir 8. Elevene bes forklare «hva som skjer når koden kjøres», og «hvordan volumet til en kule endres når radiusen dobles».

Dersom man faktisk skriver av Python-skriptet, kan man gjøre flere kjøringer om man vil, og slik prøve ut ulike verdier for å se etter et mønster. Man kan også vurdere å skrive inn kommentarer i koden for å besvare oppgave a. Kanskje det hadde vært et godt alternativ å formulere oppgaven slik at det ble lagt opp til nettopp dette?