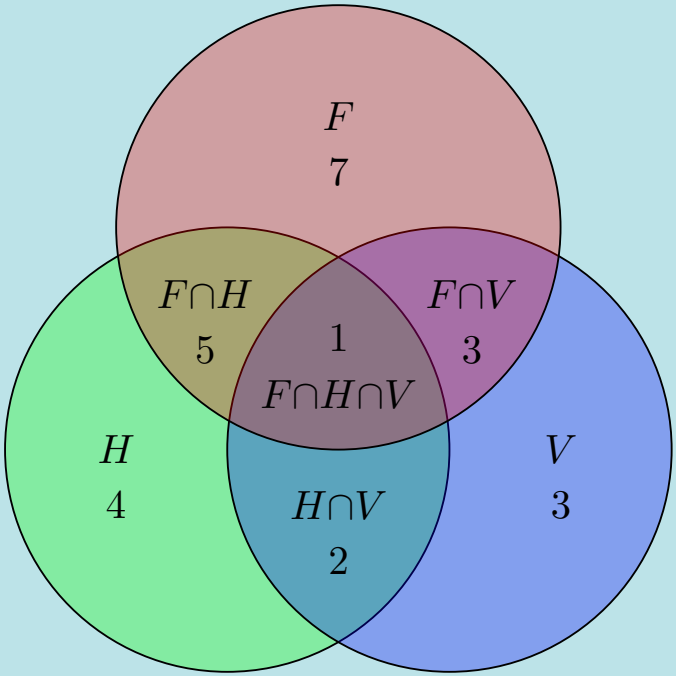
Matematikk P1



Sindre Sogge Heggen

Innhold

Kapittel 0

Digitale verktøy

En viktig del av å beherske digitale verktøy, er å forstå grunnleggende **programmering**. Programmering handler om å gi instrukser til en datamaskin. Slik kan datamaskiner utføre utregninger, framstille bilder, animasjoner, spill, og mye mer. For å gi instrukser bruker vi forskjellige **programmeringsspråk**, og det er et hav av forskjellige språk å velge i. I norsk skole er de de mest brukte språkene Scratch, Python og JavaScript[2]. Det finnes et stort utvalg av gratis ressurser for å lære seg programmeringsspråk, blant andre

- code.org (koding generelt)
- w3schools.com (koding generelt)
- scratch.mit.edu (Scratch)
- microbit.org (koding med micro:bit)
- espensklasserom.co (Koding i Sracht, micro:bit m.m.)
- kidsakoder.no (koding i Scratch, micro:bit, Python m.m)

[2]Rett nok i blokkbasert utgave ved koding av micro:bit.

Har du allerede nådd et høyt nivå som programmerer, og føler du har god kontroll på data-typer, funksjoner, klasser o.l.? Da anbefales språket Rust. Mange holder dette for å være arvtakeren til C++ og liknende språk.

Innhold

0.A Introduksjon til Python

Python er et programmeringsspråk for **tekstbasert koding**. Dette innebærer at handlingene vi ønsker utført, må kodes som tekst. Filen som inneholder hele koden kaller vi et **skript**. Det synlige resultatet av å kjøre skriptet, kaller vi **utdata**[1]. Det er mange måter å få kjørt skriptet sitt på, blant annet kan man bruke en online compiler som programmiz.com.

0.A.1 Objekt, type, funksjon og uttrykk

Vårt første skript består av bare én kodelinje:

```
1 print("Hello world!")
```

Utdata

```
Hello world!
```

I kommende avsnitt vil begrepene **objekt**, **type**, **funksjon** og **uttrykk** stadig dukke opp.

- Det aller meste i Python er objekter. I skriptet over er både `print()` og `"Hello world"` objekter.
- Objekter vil være av forskjellige typer. `print()` er av typen `function`, mens `"Hello world"` er av typen `str`[2]. Hvilke handlinger som kan utføres med forskjellige objekter avhenger av hvilke typer de er.
- Funksjoner kan ta imot **argumenter**, for så å utføre handlinger. I skriptet over tar `print()`-funksjonen imot argumentet `"Hello world"`, og printer teksten til utdata.
- Uttrykk har sterke likehetstrekk med funksjoner, men tar ikke imot argumenter.

---

[1] **Output** på engelsk.

[2] 'str' er en forkortelse for det engelske ordet 'string'.

Tilvising og utregning

Tekst og tall kan vi se på som noen av de minste byggesteinene (objektene). Python har én type for tekst, og to typer for reelle tall:

`str`	tekst
`int`	heltall
`float`	desimaltall

Det er som regel nyttig å gi objektene våre navn. Dette gjør vi ved å skrive navnet etterfulgt av `=` og objektet. **Kommentarer** er tekst som ikke blir behandlet som kode. Kommentarer kan vi skrive ved å starte setningen med `#`.

```
1 hei = "hei" # hei er av typen str. Legg merke til "
          ved start og slutt
2 a = 3 # a er av typen int
3 b = 2.8 # b er av typen float
4 c = 2. # c=2.0, og er av typen float
5 d = .7 # d=0.7, og er av typen float
6 e = -5 # e er av typen int
7 f = -0.01 # f er av typen float
```

Med Python kan vi selvsagt utføre klassiske regneoperasjoner:

```
a = 5
b = 2

print("a+b = ", a+b);
print("a-b = ", a-b);
print ("a*b = " ,a*b);
print ("a/b = " ,a/b);
print("a**b = " ,a**b); # potens med grunntall a og
                        eksponent b
print ("a//b = ", a//b); # a/b rundet ned til nærmeste
                        heltall
print ("a%b = ", a%b); # resten til a//b
```

Utdata

```
a+b = 7
a-b = 3
a*b = 10
a/b = 2.5
a**b = 25
a//b = 2
a%b = 1
```

Funksjonene `str()`, `int()` og `float()` kan vi bruke til å gjøre om objekter til typene `int` eller `float`:

```
1  s = "2"
2  b = 3
3  c = 2.0
4
5  b_s = str(b) # b omgjort til str
6  c_s = str(c) # c omgjort til str
7  print(b_s+c_s)
8
9  s_i = int(s)
10 print(s_i*b)
11
12 s_f = float(s)
13 print(s_f*b)
```

Utdata

```
32.0
6
6.0
```

En viktig ting å være klar over er at `=` i Python *ikke* betyr det samme som $=$ i matematikk. Mens $=$ kan oversettes til "er lik", kan vi si at `=` kan oversettes til 'er'.

```
1  a = 5 # a ER nå 5
2  print(a)
3  a = a+1 # a ER nå det a VAR, + 1
4  print(a)
```

Utdata

```
5
6
```

At et objekt legger til seg selv og en annen verdi er så vanlig i programmering at Python har en egen operator for det:

```
a = 5 # a ER nå 5
a+= 1 # Samme som å skrive a = a+1
print(a)
```

Utdata
5
6

Selv om datamaskiner er ekstremt raske til å utføre utregninger, har de en begrensning det er viktig å være klar over; avrundingsfeil. En av grunnene til dette er at datamaskiner bare kan bruke et visst antall desimaler for å representere tall. En annen grunn er at datamaskiner anvender totallssystemet. Det er mange verdier vi kan skrive eksakt i titallssystemet som ikke lar seg skrive eksakt i totallssystemet. For å bøte på dette kan vi bruke funksjonen `round()`:

```
a = 8.3/10
print(a) # avrundingsfeil, da vi skulle hatt a = 0.83

a = round(a, 2) # runder av a til tall med to desimaler
print(a)
```

Utdata
0.8300000000000001
0.83

0.A.2 Egne funksjoner

Ved å bruke metoden `def` kan man lage sine egne funksjoner. En funksjon kan utføre handlinger, og den kan **returnere** (return på engelsk) ett eller flere objekt. Den kan også ta imot argumenter. Koden vi skriver inni en funksjon blir bare utført hvis vi **kaller** (call på engelsk) på funksjonen.

```python
# a er en funksjon som ikke tar noen argumenter.
# Legg merke til 'def' først og ':' til slutt.
# Kodelinjene som hører til funksjonen må stå med innrykk
def a():
  print("Hei, noen kalte visst på funksjon a?")


# b er en funksjon som tar argumentet 'test'
def b(tekst):
  print("Hei. Noen kalte på funksjon b. Argumentet som ble
    gitt var: ", tekst)

# c er er funksjon som tar argmunentene a og b
# c returnerer et objekt
def c(a, b):
  return a+b

b("Hello!") # vi kaller på b med argumentet "hello"

d = c(2,3) # Vi kaller på a med argumentene 2 og 3

print(d)

# merk at teksten gitt i a ikke blir printet, fordi vi ikke
    har kalt på a.
```

Utdata

```
Hei. Noen kalte på funksjon b. Argumentet som ble gitt var:
Hello!
5
```

0.A.3 Boolske verdier og vilkår

Verdiene `True` og `False` kalles **boolske verdier**. Disse vil være resultatet når vi sjekker om objekter er like eller ulike. For å sjekke dette har vi de

sammenlignende operatorene:

operator	betydning
==	er lik
!=	er *ikke* lik
>	er større enn
>=	er større enn, eller lik
<	er mindre enn
<=	er mindre enn, eller lik

```
a = 5
b = 4

print(a == b)
print(a != b)
print(a > b)
print(a < b)
```

Utdata

```
False
True
True
False
```

I tillegg til de sammenlignende operatorene kan vi bruke de **logiske operatorene** `and`, `or` og `not`

```
a = 5
b = 4
c = 9

print(a == b and c > a)
print(a == b or c > a)
print(not a == b)
```

Utdata

```
False
True
True
```

Språkboksen

Sjekker som bruker de sammenlignende og de logiske operatorene, skal vi heretter kalle **vilkår**.

0.A.4 Uttrykkene `if`, `else` og `elif`

Når vi ønsker å utføre handlinger bare *hvis* et vilkår er sant (`True`), bruker vi uttrykket `if` foran vilkåret. Koden vi skriver med innrykk under `if`-linjen, vil bare bli utført hvis vilkåret gir `True`.

```
a = 5
b = 4
c = 9

if c > b: # legg merke til kolon (:) til slutt
  print("Jepp, c er større enn b")

if a > c: # legg merke til kolon (:) til slutt
  print("Denne teksten kommer ikke i output, siden vilkåret
     er False")
```

Utdata

```
Jepp, c er større enn b
```

Hvis man først vil sjekke om et vilkår er sant, og så utføre handlinger hvis det *ikke* er det, kan vi bruke uttrykket `else`:

```
a = 5
c = 9

if a > c: # legg merke til kolon (:) til slutt
  print("Denne teksten kommer ikke i output, siden vilkåret
     er False")

else: # legg merke til kolon (:) til slutt
  print("Men denne kommer, fordi vilkåret i if-linja over
    var False")
```

Utdata

```
Men denne kommer, fordi vilkåret i if-linja over var False
```

Uttrykket `else` tar bare hensyn til (og gir ikke mening uten) `if`-uttrykket like over seg. Hvis vi vil at handlinger skal utføres *bare* hvis ingen tidligere `if` uttrykk ga noe utslag, må vi bruke[1] uttrykket `elif`. Dette er et `if`-uttrykk som slår inn hvis `if`-uttrykket over *ikke* ga utslag.

```
a = 2

if a > 3:
  print("Denne linja printes ikke, vilkåret er False")

elif a < 1: #Siden if uttrykket over ikke ga utslag,
          sjekkes vilkåret b < 1
  print("Denne linja printes ikke, vilkåret er False")

else:
  print("Nå er vi sikre på at 1 < b < 3")
```

Utdata

```
Nå er vi sikre på at 1 < b < 3
```

[1] `elif` er en forkortelse for `else if`, som også kan brukes.

Merk

Når du jobber med tall, kan noen vilkår du forventer skal være `True` vise seg å være `False`. Dette handler ofte om avrundingsfeil, som vi har omtalt på side 8.

0.A.5 Lister

Lister kan vi bruke for å samle objekter. Objektene som er i listen kalles **elementene** til listen.

```
strings = ["98", "99", "100"]
floats = [1.7, 1.2]
ints = [96, 97, 98, 99, 100]
mixed = [1.7, 96, "100"]
empty = []
```

Elementene i lister er **indekserte**. Første objekt har indeks 0, andre objekt har indeks 1 og så videre:

```
strings = ["98", "99", "100"]
floats = [1.7, 1.2]
ints = [96, 97, 98, 99, 100]
mixed = [1.7, 96, "100"]
empty = []
```

Utdata

```
96
99
98
```

Med den innebygde funksjonen `append()` kan vi legge til et objekt i enden av listen. Dette er en **innebygd funksjon**[1], som vi skriver i enden av navnet på listen, med et punktum foran.

[1] Kort fortalt betyr det at det bare er noen typer objekter som kan bruke denne funksjonen.

```python
min_liste = []
print(min_liste)

min_liste.append(3)
print(min_liste)

min_liste.append(7)
print(min_liste)
```

Utdata

```
[]
[3]
[3, 7]
```

Med funksjonen `pop()` kan vi hente ut et objekt fra listen

```
min_liste = [6, 10, 15, 19]

a = min_liste.pop() # a = det siste elementet i listen
print("a =",a)
print("min_liste =",min_liste)

a = min_liste.pop(1) # a = elementet med indeks 1
print("a =",a)
print("min_liste =",min_liste)
```

Utdata

```
a = 19
min_liste = [6, 10, 15]
a = 10
min_liste = [6, 15]
```

Forklar for deg selv

Hva er forskjellen på å skive `a = min_liste[1]` og å skrive `a = min_liste.pop(1)`?

Med funksjonen `sort()` kan vi sortere elementene i listen.

```
heltall =  [9, 0, 8, 3, 1, 7, 4]
bokstaver = ['c', 'a', 'b', 'e', 'd']

heltall.sort()
bokstaver.sort()

print(heltall)
print(bokstaver)
```

Utdata

```
[0, 1, 3, 4, 7, 8, 9]
['a', 'b', 'c', 'd', 'e']
```

Med funksjonene `count()` kan vi telle gjentatte elementer i listen.

```
heltall =  [2, 7, 2, 2, 2]
frukt = ['banan', 'eple', 'banan']

antall_toere = heltall.count(2)
antall_sjuere = heltall.count(7)
antall_bananer = frukt.count('banan')
antall_appelsiner = frukt.count('appelsin')

print(antall_toere)
print(antall_sjuere)
print(antall_bananer)
print(antall_appelsiner)
```

Utdata

```
4
1
2
0
```

Med funksjonen `len()` kan vi finne antall elementer i en liste, og med funksjonen `sum()` kan vi finne summen av lister med tall som elementer.

```
heltall =  [2, 7, 2, 2, 2]
frukt = ['banan', 'eple', 'banan']

print(len(heltall))
print(len(frukt))
print(sum(heltall))
```

Utdata

```
5
3
15
```

Med uttrykket `in` kan vi sjekke om et element er i en liste.

```
heltall =  [1, 2, 3]

print(1 in heltall)
print(0 in heltall)
```

Utdata

```
True
False
```

0.A.6 Looper; `for` og `while`

`for` loop

For objekter som inneholder flere elementer, kan vi bruke `for`-looper til å utføre handlinger for hvert element. Handlingene må vi skrive med et innrykk etter `for`-uttrykket:

```
1 min_liste = [5, 10, 15]
2
3 for number in min_liste:
4   print(number)
5   print(number*10)
6   print("\n") # lager et blankt mellomrom
7
```

Utdata

```
5
50

10
100

15
150
```

Språkboksen

Å gå gjennom hvert element i (for eksempel) en liste kalles "å iterere over listen".

Ofte er det ønskelig å iterere over heltallene $0, 1, 2$ og så videre. Til dette kan vi bruke `range()`:

```
ints = range(3)

for i in ints:
  print(i)

```

Utdata
0
1
2

`while` loop

Hvis vi ønsker at handlinger skal utføres fram til et vilkår er sant, kan vi bruke en `while`-loop:

```
a = 1

while a < 5:
  print(a)
  a += 1
```

Utdata
1
2
3
4

0.A.7 `input()`

Vi kan bruke funksjonen `input()` til å skrive inn tekst mens skriptet kjører:

```
innskrevet_tekst = input("Skriv inn her: ")
print(innskrevet_tekst)
```

Teksten vi har skrevet inni `input()` i skriptet over er teksten vi ønsker vist foran teksten som ønskes innskrevet. Linje 2 i denne koden vil ikke kjøres før en tekst er innskrevet.

```
1 innskrevet_tekst = input("Skriv inn her: ")
2 print(innskrevet_tekst)
```

Utdata

```
Skriv inn tekst her: OK
OK
```

Objektet gitt av en `input()`-funksjon vil alltid være av typen `str`. Man må alltid passe på å gjøre om objekter til rett type:

```
print("La oss regne ut a*b")
a_str = input("a = ")
b_str = input("b = ")
a = float(a_str)
b = float(b_str)
print("a*b = ", a*b)
```

Utdata

```
La oss regne ut a*b
a = 3.7
b = 4
a*b = 14.8
```

0.A.8 Moduler

Noen ganger er vi nødt til å importere **moduler** for å få tilgang til objekter vi ønsker. Dette gjør vi ved å bruke `import`-metoden:

```
# importerer konstanten pi fra modulen 'math'
from math import pi
print(pi)

# importerer funksjonen randint fra modulen 'random'
from random import randint

# randint(2, 5) returnerer tilfeldig valgt heltall
# mellom (og inkludert) 2 og 5.
print(randint(2, 5))
```

Utdata

```
3.141592653589793
4
```

0.A.9 Feilmeldinger

Påstand: Alle programmerere vil erfare at skriptet ikke kjører fordi vi ikke har skrevet koden på rett måte. Dette kalles en **syntax error**. Ved syntax error vil man få beskjed om på hvilken linje feilen befinner seg, og hva som er feil. De vanligste feilene er

- **Å glemme innrykk når man bruker metoder som `def`, `for`, `while`, og `if`**

```
1 a = 472
2 b = 98
3
4 if a*b > 48000:
5 print("a*b er større enn 48000")
```

Utdata

```
line 5, in <module>
print("a*b er større enn 48000")
^
IndentationError: expected an indented block after 'if'
statement on line 4
```

- **Å utføre operasjoner på typer det ikke gir mening for**

```
1 b = "98"
2 b_opphøyd_i_andre = b**2
```

Utdata

```
line 2, in <module>
b_opphøyd_i_andre = b**2
TypeError: unsupported operand type(s) for ** or
pow(): 'str' and 'int'
```

Oppgaver for kapittel 0

0.A.1

Lag en funskjon `f(a, b)` som

a) printer $a \cdot b$.

b) printer $a^2 + b^2$. Hva tilsvarer dette tallet hvis a og b er lengdene til katetene i en rettvinklet trekant?

c) runder av et desimaltall a til et tall med b desimaler.

d) printer om a er større, lik, eller mindre enn b.

0.A.2

Skriv outputen til dette skriptet:

```
for i in range(4):
  print(i + 2)
```

0.A.3

Skriv outputen til dette skriptet:

```
i = 0
while i < 5:
  i = i + 2
  print(i)
```

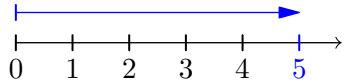
Kapittel 1

Tall

1.A Tallregning

Introduksjon til negative tall

Vi har tidligere sett at (for eksempel) tallet 5 på ei tallinje ligger 5 enerlengder til høyre for 0.

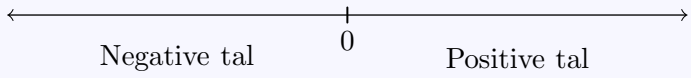


Men hva om vi går andre veien, altså mot venstre? Dette spørsmålet svarer vi på ved å innføre *negative tall*. .

1.1 Positive og negative tal

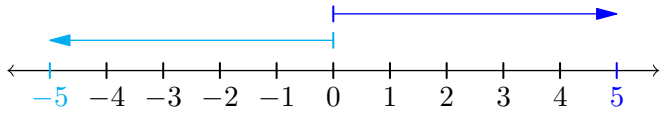
På en tallinje gjelder følgende:

- Tall plassert *til høyre* for 0 er **positive tall**.
- Tall plassert *til venstre* for 0 er **negative tal**.



Vi kan ikke hele tiden bruke en tallinje for å avgjøre om et tall er negativt eller positivt, og derfor bruker vi et tegn for å vise at tall er negative. Dette tegnet er rett og slett $-$, altså det samme tegnet som vi bruker ved subtraksjon. 5 er med dét et positivt tall, mens -5 er et negativt tall. På tallinja er det slik at

- 5 ligger 5 enerlengder *til høyre* for 0.
- -5 ligger 5 enerlengder *til venstre* for 0.



Den store forskjellen på 5 og -5 er altså på hvilken side av 0 tallene ligger. Da 5 og -5 har samme avstand til 0, sier vi at 5 og -5 har samme **lengde**.

1.2 Lengde

Lengden til et tall skrives ved symbolet $|\ |$.

Lengden til et positivt tall er verdien til tallet.

Lengden til et negativt tall er verdien til det positive tallet med samme siffer.

Eksempel 1 $|27| = 27$

Eksempel 2 $|-27| = 27$

Språkboksen

Andre ord for lengde er **tallverdi** og **absoluttverdi**.

Fortegn

Fortegn er en samlebetegnelse for $+$ og $-$. 5 har $+$ som fortegn og -5 har $-$ som fortegn.

Regning med negative tall

Ved innføringen av negative tall får de fire regneartene nye sider som vi må se på trinnvis. Når vi adderer, subtraherer, multipliserer eller dividerer med negative tall vil vi ofte, for tydeligheten sin skyld, skrive negative tall med parentes rundt. Da skriver vi for eksempel -4 som (-4).

Addisjon

Når vi adderte i seksjon **??** så vi på $+$ som vandring *mot høyre*. Negative tall gjør at vi må utvide begrepet for $+$:

$+$ "Like langt og i *samme* retning som"

La oss se på regnestykket

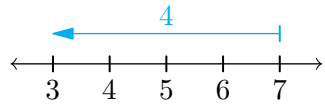
$$7 + (-4)$$

Vår utvidede definisjon av $+$ sier oss nå at

$$7 + (-4) = \text{"7 og like langt og i } \textit{samme} \text{ retning som } (-4)\text{"}$$

(-4) har lengde 4 og retning *mot venstre*. Vårt regnestykke sier altså at vi skal starte på 7, og deretter gå lengden 4 *mot venstre*.

$$7 + (-4) = 3$$



1.3 Addisjon med negative tall

Å addere et negativt tall er det samme som å subtrahere tallet med samme tallverdi.

Eksempel 1

$$4 + (-3) = 4 - 3 = 1$$

Eksempel 2

$$-8 + (-3) = -8 - 3 = -11$$

Addisjon er kommutativ

Regel **??** er gjeldende også etter innføringen av negative tall, for eksempel er

$$7 + (-3) = 4 = -3 + 7$$

Subtraksjon

I seksjon **??** så vi på $-$ som vandring *mot venstre*. Også betydningen av $-$ må utvides når vi jobber med negative tall:

$-$ "Like langt og i *motsatt* retning som"

La oss se på regnestykket

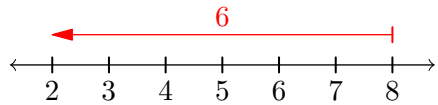
$$2 - (-6)$$

Med vår utvidede betydning av $-$, kan vi skrive

$$2 - (-6) = \text{"2 og like langt og i } \textit{motsatt} \text{ retning som } (-6)\text{"}$$

-6 har lengde 6 og retning *mot venstre*. Når vi skal gå samme lengde, men i *motsatt* retning, må vi altså gå lengden 6 *mot høyre*[1]. Dette er det samme som å addere 6:

$$2 - (-6) = 2 + 6 = 8$$



[1] Vi minner enda en gang om at rødfargen på pila indikerer at man skal vandre fra pilspissen til andre enden.

1.4 Subtraksjon med negative tall

Å subtrahere et negativt tall er det samme som å addere det positive tallet med samme tallverdi.

Eksempel 1

$$11 - (-9) = 11 + 9 = 20$$

Eksempel 2

$$-3 - (-7) = -3 + 7 = 4$$

Multiplikasjon

I seksjon **??** introduserte vi ganging med positive heltall som gjentatt addisjon. Med våre utvidede begrep av addisjon og subtraksjon, kan vi nå også utvide begrepet multiplikasjon:

1.5 Multiplikasjon med positive og negative tall

Ganging med et positivt heltall er det samme som gjentatt addisjon.

Ganging med et negativt heltal er det samme som gjentatt subtraksjon.

Eksempel 1

$$\begin{aligned} 2 \cdot 3 &= \text{"Like langt og i } \textit{samme} \text{ retning som 2, 3 ganger"} \\ &= 2 + 2 + 2 \\ &= 6 \end{aligned}$$

Eksempel 2

$$\begin{aligned} (-2) \cdot 3 &= \text{"Like langt og i } \textit{samme} \text{ retning som } (-2), \text{ 3 ganger"} \\ &= -2 - 2 - 2 \\ &= -6 \end{aligned}$$

Eksempel 3

$$\begin{aligned} 2 \cdot (-3) &= \text{"Like langt og i } \textit{motsatt} \text{ retning som 2, 3 ganger"} \\ &= -2 - 2 - 2 \\ &= -6 \end{aligned}$$

Eksempel 4

$$\begin{aligned} (-3) \cdot (-4) &= \text{"Like langt og i } \textit{motsatt} \text{ retning som } -3\text{, 4 ganger"} \\ &= 3 + 3 + 3 + 3 \\ &= 12 \end{aligned}$$

Multiplikasjon er kommutativ

Eksempel 2 og *Eksempel 3* på side 31 illustrerer at regel **??** også er gjeldende etter innføringen av negative tall:

$$(-2) \cdot 3 = 3 \cdot (-2)$$

Det blir tungvint å regne ganging som gjentatt addisjon/subtraksjon hver gang vi har et negativt tall involvert, men som en direkte konsekvens av regel 1.5 kan vi lage oss følgende to regler:

1.6 Multiplikasjon mellom et negativt og et positivt tall

Produktet av et negativt og et positivt tall er et negativt tall.

Tallverdien til faktorene ganget sammen gir tallverdien til produktet.

Eksempel 1

Regn ut $(-7) \cdot 8$

Svar

Siden $7 \cdot 8 = 56$, er $(-7) \cdot 8 = -56$

Eksempel 2

Regn ut $3 \cdot (-9)$.

Svar

Siden $3 \cdot 9 = 27$, er $3 \cdot (-9) = -27$

1.7 Multiplikasjon mellom to negative tall

Produktet av to negative tall er et positivt tall.

Tallverdien til faktorene ganget sammen gir verdien til produktet.

Eksempel 1 $(-5) \cdot (-10) = 5 \cdot 10 = 50$

Eksempel 2 $(-2) \cdot (-8) = 2 \cdot 8 = 16$

Divisjon

Definisjonen av divisjon (se seksjon **??**), kombinert med det vi vet om multiplikasjon med negative tall, gir oss nå dette:

$-18 : 6 =$ "Tallet jeg må gange 6 med for å få -18"

$6 \cdot (-3) = -18$, altså er $-18 : 6 = -3$

$42 : (-7) =$ "Tallet jeg må gange -7 med for å få 42"

$(-7) \cdot (-8) = 42$, altså er $42 : (-7) = -8$

$-45 : (-5) =$ "Tallet jeg må gange -5 med for å få -45"

$(-5) \cdot 9 = -45$, altså er $-45 : (-5) = 9$

1.8 Divisjon med negative tall

Divisjon mellom et positivt og et negativt tall gir et negativt tall.

Divisjon mellom to negative tall gir et positivt tall.

Tallverdien til dividenden delt med tallverdien til divisoren gir tallverdien til kvotienten.

Eksempel 1 $-24 : 6 = -4$

Eksempel 2 $24 : (-2) = -12$

Eksempel 3 $-24 : (-3) = 8$

Eksempel 4 $\dfrac{2}{-3} = -\dfrac{2}{3}$

Eksempel 5 $\frac{-10}{7} = -\frac{10}{7}$

Ganging med 10, 100, 1 000 osv.

1.9 Å gange heltall med 10, 100 osv.

- Når man ganger et heltall med 10, får man svaret ved å legge til sifferet 0 bak heltallet.
- Når man ganger et heltall med 100, får man svaret ved å legge til sifrene 00 bak heltallet.
- Det samme mønsteret gjelder for tallene 1 000, 10 000 osv.

Eksempel 1

$$6 \cdot 10 = 60$$

$$79 \cdot 10 = 790$$

$$802 \cdot 10 = 8020$$

Eksempel 2

$$6 \cdot 100 = 600$$

$$79 \cdot 100 = 7\,900$$

$$802 \cdot 100 = 80\,200$$

Eksempel 3

$$6 \cdot 1\,000 = 6\,000$$

$$79 \cdot 10\,000 = 790\,000$$

$$802 \cdot 100\,000 = 80\,200\,000$$

1.10 Å gange desimaltall med 10, 100 osv.

- Når man ganger et desimaltall med 10, får man svaret ved å flytte komma en plass til høgre.
- Når man ganger et heltall med 100, får man svaret ved å flytte komma to plasser til høgre.
- Det samme mønsteret gjelder for tallene 1 000, 10 000 osv.

Eksempel 1

$$7{,}9 \cdot 10 = 79{,} = 79$$

$$38{,}02 \cdot 10 = 380{,}2$$

$$0{,}57 \cdot 10 = 05{,}7 = 5{,}7$$

$$0{,}194 \cdot 10 = 01{,}94 = 1{,}94$$

Eksempel 2

$$7{,}9 \cdot 100 = 790{,} = 790$$

$$38{,}02 \cdot 100 = 3802{,} = 3\,802$$

$$0{,}57 \cdot 100 = 057{,} = 57$$

$$0{,}194 \cdot 100 = 019{,}4 = 19{,}4$$

Eksempel 3

$$7{,}9 \cdot 1\,000 = 7900{,} = 7\,900$$

$$38{,}02 \cdot 10\,000 = 380020{,} = 380\,200$$

$$0{,}57 \cdot 100\,000 = 57000{,} = 57\,000$$

Merk

Regel 1.9 er bare et spesialtilfelle av regel 1.10. For eksempel, å bruke regel 1.9 på regnestykket $7 \cdot 10$ gir samme resultat som å bruke regel

1.10 på regnestykket $7{,}0 \cdot 10$.

Å gange tall med 10, 100 osv. (forklaring)

Titallsystemet baserer seg på grupper av ti, hundre, tusen osv., og tideler, hundredeler og tusendeler osv (se regel **??**). Når man ganger et tall med 10, vil alle enere i tallet bli til tiere, alle tiere bli til hundrere osv. Hvert siffer forskyves altså én plass mot venstre. Tilsvarende forskyves hvert siffer to plasser mot venstre når man ganger med 100, tre plasser når man ganger med 1 000 osv.

Deling med 10, 100, 1 000 osv.

1.11 Deling med 10, 100, 1 000 osv.

Når man deler et desimaltall med 10, får man svaret ved å flytte komma én plass til venstre.

Når man deler et desimaltall med 100, får man svaret ved å flytte komma to plasser til venstre.

Det samme mønsteret gjelder for tallene 1 000, 10 000 osv.

Eksempel 1

$$\begin{aligned} 200 : 10 &= 200{,}0 : 10 \\ &= 20{,}00 \\ &= 20 \end{aligned}$$

$$\begin{aligned} 45 : 10 &= 45{,}0 : 10 \\ &= 4{,}50 \\ &= 4{,}5 \end{aligned}$$

Eksempel 2

$$
\begin{aligned}
200 : 100 &= 200{,}0 : 100 \\
&= 2{,}000 \\
&= 2
\end{aligned}
$$

$$
\begin{aligned}
45 : 100 &= 45{,}0 : 100 \\
&= 0{,}450 \\
&= 0{,}45
\end{aligned}
$$

Eksempel 3

$$143{,}7 : 10 = 14{,}37$$

$$143{,}7 : 100 = 1{,}437$$

$$143{,}7 : 1\,000 = 0{,}1437$$

Eksempel 4

$$93{,}6 : 10 = 9{,}36$$

$$93{,}6 : 100 = 0{,}936$$

$$93{,}6 : 1\,000 = 0{,}0936$$

Deling med 10, 100, 1 000 osv. (forklaring)

Titallsystemet baserer seg på grupper av ti, hundre, tusen osv., og tideler, hundredeler og tusendeler osv (se regel **??**). Når man deler et tall med 10, vil alle enere i tallet bli til tideler, alle tiere bli til enere osv. Hvert siffer forskyves altså én plass mot høgre. Tilsvarende forskyves hvert siffer to plasser mot høgre når man deler med 100, tre plasser når man deler med 1 000 osv.

Overslagsregning

Det er ikke alltid vi trenger å vite svaret på regnestykker helt nøyaktig, ofte er det viktigere at vi fort kan avgjøre hva svaret *omtrent* er det samme som, aller helst ved hoderegning. Når vi finner svar som omtrent er riktige, sier vi at vi gjør et **overslag**. Et overslag innebærer at vi avrunder[1] tallene som inngår i et regnestykke slik at utregningen blir enklere.

Språkboksen

[1] *Obs!* Avrunding ved overslag trenger ikke å innebære avrunding til nærmeste tier og lignende.

At noe er "omtrent det samme som" skriver vi ofte som "cirka" ("ca."). Tegnet for "cirka" er $\approx$.

Overslag ved addisjon og ganging

La oss gjøre et overslag på regnestykket

$$98{,}2 + 24{,}6$$

Vi ser at $98{,}2 \approx 100$. Skriver vi 100 istedenfor 98,2 i regnestykket vårt, får vi noe som er litt mer enn det nøyaktige svaret. Skal vi endre på 24,6 bør vi derfor gjøre det til et tall som er litt mindre. 24,6 er ganske nærme 20, så vi kan skrive

$$98{,}2 + 24{,}6 \approx 100 + 20 = 120$$

Når vi gjør overslag på tall som legges sammen, bør vi altså prøve å gjøre det ene tallet større (runde opp) og et tall mindre (runde ned).

Det samme gjelder også hvis vi har ganging, for eksempel

$$1\,689 \cdot 12$$

Her avrunder vi 12 til 10. For å "veie opp" for at svaret da blir litt mindre enn det egentlige, avrunder vi 1 689 opp til 1 700. Da får vi

$$1\,689 \cdot 12 \approx 1\,700 \cdot 10 = 17\,000$$

Overslag ved subtraskjon og deling

Skal et tall trekkes fra et annet, blir det litt annerledes. La oss gjøre et overslag på

$$186{,}4 - 28{,}9$$

Hvis vi runder 186,4 opp til 190 får vi et svar som er større enn det egentlige, derfor bør vi også trekke ifra noe. Det kan vi gjøre ved også å runde 28,9 oppover (til 30):

$$\begin{aligned} 186{,}4 - 28{,}9 &\approx 190 - 30 \\ &= 160 \end{aligned}$$

Samme prinsippet gjelder for deling:

$$145 : 17$$

Vi avrunder 17 opp til 20. Deler vi noe med 20 istedenfor 17, blir svaret mindre. Derfor bør vi også runde 145 oppover (til 150):

$$145 : 17 \approx 150 : 20 = 75$$

Overslagsregning oppsummert

1.12 Overslagsregning

- Ved addisjon eller multiplikasjon mellom to tall, avrund gjerne et tall opp og et tall ned.
- Ved subtraksjon eller deling mellom to tall, avrund gjerne begge tall ned eller begge tall opp.

Eksempel

Rund av og finn omtrentlig svar for regnestykkene.

a) $23{,}1 + 174{,}7$ b) $11{,}8 \cdot 107{,}2$

c) $37{,}4 - 18{,}9$ d) $1054 : 209$

Svar

a) $32{,}1 + 174{,}7 \approx 30 + 170 = 200$

b) $11{,}8 \cdot 107{,}2 \approx 10 \cdot 110 = 1\,100$

c) $37{,}4 - 18{,}9 \approx 40 - 20 = 20$

d) $1\,054 : 209 \approx 1\,000 : 200 = 5$

Kommentar

Det finnes ingen konkrete regler for hva man *kan* eller ikke *kan* tillate seg av forenklinger når man gjør et overslag, det som er kalt regel 1.12 er strengt tatt ikke en regel, men et nyttig tips.

Man kan også spørre seg hvor langt unna det faktiske svaret man kan tillate seg å være ved overslagsregning. Heller ikke dette er det noe fasitsvar på, men en grei føring er at overslaget og det faktiske svaret skal være av samme **størrelsesorden**. Litt enkelt sagt betyr dette at hvis det faktiske svaret har med tusener å gjøre, bør også overslaget ha med tusener å gjøre. Mer nøyaktig sagt betyr det av det faktiske svaret og ditt overslag bør ha samme tierpotens når de er skrevet på standardform[1].

---

[1] Se seksjon 1.C

1.B Potenser

$$\text{grunntal} \longrightarrow 2^3 \longleftarrow \text{eksponent}$$

En potens består av et **grunntall** og en **eksponent**. For eksempel er 2^3 en potens med grunntall 2 og eksponent 3. En positiv, heltalls eksponent sier hvor mange eksemplar av grunntallet som skal ganges sammen, altså er

$$2^3 = 2 \cdot 2 \cdot 2$$

1.13 Potenstall

a^n er et potenstall med grunntall a og eksponent n.

Hvis n er et naturlig tall, vil a^n svare til n eksemplar av a multiplisert med hverandre.

Eksempel 1

$$\begin{aligned} 5^3 &= 5 \cdot 5 \cdot 5 \\ &= 125 \end{aligned}$$

Eksempel 2

$$c^4 = c \cdot c \cdot c \cdot c$$

Eksempel 3

$$\begin{aligned} (-7)^2 &= (-7) \cdot (-7) \\ &= 49 \end{aligned}$$

Eksempel 4

$$a^1 = a$$

Språkboksen

Vanlige måter å si 2^3 på er

- "2 i tredje"
- "2 opphøyd i 3"

I programmeringsspråk brukes gjerne symbolet `^` eller symbolene `**` mellom grunntall og eksponent.

Å opphøye et tall i 2 kalles "å kvadrere" tallet.

Merk

De kommende sidene vil inneholde regler for potenser med tilhørende forklaringer. Selv om det er ønskelig at de har en så generell form som mulig, har vi i forklaringene valgt å bruke eksempel der eksponentene ikke er variabler. Å bruke variabler som eksponenter ville gitt mye mindre leservennlige uttrykk, og vi vil påstå at de generelle tilfellene kommer godt til synes også ved å studere konkrete tilfeller.

1.14 Multiplikasjon mellom potenser

$$a^m \cdot a^n = a^{m+n} \tag{1.1}$$

Eksempel 1

$$\begin{aligned} 3^5 \cdot 3^2 &= 3^{5+2} \\ &= 3^7 \end{aligned}$$

Eksempel 2

$$\begin{aligned} b^4 \cdot b^{11} &= b^{3+11} \\ &= b^{14} \end{aligned}$$

Eksempel 3

$$\begin{aligned} a^5 \cdot a^{-7} &= a^{5+(-7)} \\ &= a^{5-7} \\ &= a^{-2} \end{aligned}$$

(Se regel 1.17 for hvordan en potens med negativ eksponent kan tolkes.)

1.14 Multiplikasjon mellom potenser (forklaring)

La oss se på tilfellet

$$a^2 \cdot a^3$$

Vi har at

$$a^2 = 2 \cdot 2$$

$$a^3 = 2 \cdot 2 \cdot 2$$

Med andre ord kan vi skrive

$$\begin{aligned} a^2 \cdot a^3 &= \overbrace{a \cdot a}^{a^2} \cdot \overbrace{a \cdot a \cdot a}^{a^3} \\ &= a^5 \end{aligned}$$

1.15 Divisjon mellom potenser

$$\frac{a^m}{a^n} = a^{m-n}$$

Eksempel 1

$$\frac{3^5}{3^2} = 3^{5-2} = 3^3$$

Eksempel 2

$$\begin{aligned} \frac{2^4 \cdot a^7}{a^6 \cdot 2^2} &= 2^{4-2} \cdot a^{7-6} \\ &= 2^2 a \\ &= 4a \end{aligned}$$

1.15 Divisjon mellom potenser (forklaring)

La oss undersøke brøken

$$\frac{a^5}{a^2}$$

Vi skriver ut potensene i teller og nevner:

$$\begin{aligned}\frac{a^5}{a^2} &= \frac{a \cdot a \cdot a \cdot a \cdot a}{a \cdot a} \\ &= \frac{\cancel{a} \cdot \cancel{a} \cdot a \cdot a \cdot a}{\cancel{a} \cdot \cancel{a}} \\ &= a \cdot a \cdot a \\ &= a^3\end{aligned}$$

Dette kunne vi ha skrevet som

$$\begin{aligned}\frac{a^5}{a^2} &= a^{5-2} \\ &= a^3\end{aligned}$$

1.16 Potens med 0 som eksponent

$$a^0 = 1$$

Eksempel 1 $1000^0 = 1$

Eksempel 2 $(-b)^0 = 1$

1.16 Potens med 0 som eksponent (forklaring)

Et tall delt på seg selv er alltid lik 1, derfor er

$$\frac{a^n}{a^n} = 1$$

Av dette, og regel 1.15, har vi at

$$\begin{aligned} 1 &= \frac{a^n}{a^n} \\ &= a^{n-n} \\ &= a^0 \end{aligned}$$

1.17 Potenser med negativ eksponent

$$a^{-n} = \frac{1}{a^n}$$

Eksempel 1

$$a^{-8} = \frac{1}{a^8}$$

Eksempel 2

$$(-4)^{-3} = \frac{1}{(-4)^3} = -\frac{1}{64}$$

1.17 Potenser med negativ eksponent (forklaring)

Av regel 1.16 har vi at $a^0 = 1$. Altså er

$$\frac{1}{a^n} = \frac{a^0}{a^n}$$

Av regel 1.15 er

$$\begin{aligned} \frac{a^0}{a^n} &= a^{0-n} \\ &= a^{-n} \end{aligned}$$

1.18 Brøker med potenser

$$\left(\frac{a}{b}\right)^m = \frac{a^m}{b^m} \tag{1.2}$$

Eksempel 1

$$\left(\frac{3}{4}\right)^2 = \frac{3^2}{4^2} = \frac{9}{16}$$

Eksempel 2

$$\left(\frac{a}{7}\right)^3 = \frac{a^3}{7^3} = \frac{a^3}{343}$$

1.18 Brøker med potenser (forklaring)

La oss studere

$$\left(\frac{a}{b}\right)^3$$

Vi har at

$$\begin{aligned}\left(\frac{a}{b}\right)^3 &= \frac{a}{b}\cdot\frac{a}{b}\cdot\frac{a}{b} \\ &= \frac{a\cdot a\cdot a}{b\cdot b\cdot b} \\ &= \frac{a^3}{b^3}\end{aligned}$$

1.19 Faktorer som grunntall

$$(ab)^m = a^m b^m \tag{1.3}$$

Eksempel 1

$$\begin{aligned}(3a)^5 &= 3^5 a^5 \\ &= 243a^5\end{aligned}$$

Eksempel 2

$$(ab)^4 = a^4 b^4$$

1.19 Faktorer som grunntall (forklaring)

La oss bruke $(a \cdot b)^3$ som eksempel. Vi har at

$$\begin{aligned}(a \cdot b)^3 &= (a \cdot b) \cdot (a \cdot b) \cdot (a \cdot b) \\ &= a \cdot a \cdot a \cdot b \cdot b \cdot b \\ &= a^3 b^3\end{aligned}$$

1.20 Potens som grunntall

$$(a^m)^n = a^{m \cdot n} \tag{1.4}$$

Eksempel 1

$$\begin{aligned} \left(c^4\right)^5 &= c^{4 \cdot 5} \\ &= c^{20} \end{aligned}$$

Eksempel 2

$$\begin{aligned} \left(3^{\frac{5}{4}}\right)^8 &= 3^{\frac{5}{4} \cdot 8} \\ &= 3^{10} \end{aligned}$$

1.20 Potens som grunntall (forklaring)

La oss bruke $\left(a^3\right)^4$ som eksempel. Vi har at

$$\left(a^3\right)^4 = a^3 \cdot a^3 \cdot a^3 \cdot a^3$$

Av regel 1.14 er

$$\begin{aligned} a^3 \cdot a^3 \cdot a^3 \cdot a^3 &= a^{3+3+3+3} \\ &= a^{3 \cdot 4} \\ &= a^{12} \end{aligned}$$

1.21 *n*-rot

$$a^{\frac{1}{n}} = \sqrt[n]{a}$$

Symbolet $\sqrt{\ }$ kalles et **rottegn**. For eksponenten $\frac{1}{2}$ er det vanlig å utelate 2 i rottegnet:

$$a^{\frac{1}{2}} = \sqrt{a}$$

Eksempel

Av regel 1.20 har vi at

$$\begin{aligned}\left(a^b\right)^{\frac{1}{b}} &= a^{b\cdot\frac{1}{b}} \\ &= a\end{aligned}$$

For eksempel er

$$9^{\frac{1}{2}} = \sqrt{9} = 3, \text{ siden } 3^2 = 9$$

$$125^{\frac{1}{3}} = \sqrt[3]{125} = 5, \text{ siden } 5^3 = 125$$

$$16^{\frac{1}{4}} = \sqrt[4]{16} = 2, \text{ siden } 2^4 = 16$$

Språkboksen

$\sqrt{9}$ kalles "kvadratrota til 9"

$\sqrt[5]{9}$ kalles "femterota til 9".

1.C Standardform

Vi kan utnytte regel 1.10 og regel 1.11, og det vi kan om potenser, til å skrive tall på **standardform**.

La oss se på tallet 6 700. Av regel 1.10 vet vi at

$$6\,700 = 6{,}7 \cdot 1\,000$$

Og siden $1000 = 10^3$, er

$$6\,700 = 6{,}7 \cdot 1\,000 = 6{,}7 \cdot 10^3$$

$6{,}7 \cdot 10^3$ er 6 700 skrevet på standardform fordi

- 6,7 er større eller lik 1 og mindre enn 10.
- 10^3 er en potens med grunntall 10 og eksponent 3, som er et heltall.
- 6,7 og 10^3 er ganget sammen.

La oss også se på tallet 0,093. Av regel 1.11 har vi at

$$0{,}093 = 9{,}3 : 100$$

Men å dele med 100 er det samme som å gange med 10^{-2}, altså er

$$0{,}093 = 9{,}3 : 100 = 9{,}3 \cdot 10^{-2}$$

$9{,}3 \cdot 10^{-2}$ er 0,093 skrevet på standardform fordi

- 9,3 er større eller lik 1 og mindre enn 10.
- 10^{-2} er en potens med grunntall 10 og eksponent -2, som er et heltall.
- 9,3 og 10^{-2} er ganget sammen.

1.22 Standardform

Et tall skrevet som

$$a \cdot 10^n$$

hvor $1 \leq |a| < 10$ og n er et heltall, er et tall skrevet på standardform.

Eksempel 1

Skriv 980 på standardform.

Svar $980 = 9{,}8 \cdot 10^2$

Eksempel 2

Skriv 0,00671 på standardform.

Svar $0{,}00671 = 6{,}71 \cdot 10^{-3}$

Tips

For å skrive om tall på standardform kan du gjøre følgende:

1. Flytt komma slik at du får et tall som ligger mellom 0 og 10.
2. Gang dette tallet med en tierpotens som har eksponent med tallverdi lik antallet plasser du flyttet komma. Flyttet du komma mot venstre/høgre, er eksponenten positiv/negativ.

Eksempel 3

Skriv 9 761 432 på standardform.

Svar

1. Vi flytter komma 6 plasser til venstre, og får 9,761432

2. Vi ganger dette tallet med 10^6, og får at

$$9\,761\,432 = 9{,}761432 \cdot 10^6$$

Eksempel 4

Skriv 0,00039 på standardform.

Svar

1. Vi flytter komma 4 plasser til høgre, og får 3,9.
2. Vi ganger dette tallet med 10^{-4}, og får at

$$0{,}00039 = 3{,}9 \cdot 10^{-4}$$