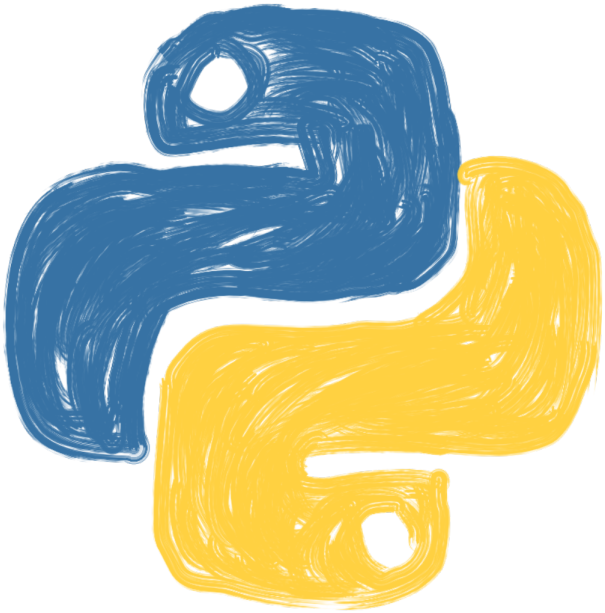
# *Introduksjon til grunnleggende programmering i Python*



**Heftet og tilhørende videoer er laget av:**

**Daniel Andreas Aubert**

# Før du starter:

## *Skritt 1: Se på introduksjonsvideoen.*

*Følg QR-koden eller linken til høyre for å sette i gang.*
*Se på denne videoen alene eller med klasse før du starter med heftet for å få en best mulig start. For mange vil den være absolutt nødvendig.*

**Introduksjonsvideo:**

https://pythonintroduksjon.studenthjelp.no/

## *Skritt 2: Installer et IDE (Integrert utviklingsmiljø):*

***Anbefalt (Benyttes aktivt i heftet):***
***https://www.spyder-ide.org***
*Andre populære:*
*https://thonny.org*

*Som forklart i introduksjonsvideoen, det går an å benytte seg av online verktøy også (men ikke anbefalt).*
***Anbefalt (om du MÅ benytte online løsning):***
***https://www.programiz.com/python-programming/online-compiler/#google_vignette***
*Populær (ikke spesielt anbefalt): https://trinket.io*

**Hvordan installere IDE (forskjellige platformer):**

https://pythonide.studenthjelp.no/

## *Skritt 3: Start læringen*

- ***Følg linkene øverst i høyre hjørnet av hvert delkapittel for å se programmeringsforelesning av delkapittelet.*** *(Alternativt følg linken til samlingsvideoen markert i blått under)*
  *Husk at det går an å endre hastigheten på forelesningene etter eget ønske på YouTube.*

- ***Gå igjennom delkapittelene:*** *Det går kanskje an å gjøre oppgavene ved bare å se film, men det anbefales at du sikrer deg om at du har fått med deg alt. Prøv gjerne å skrive og kjøre eksemplene for god læring!*

- ***Gjør oppgavene:*** *Ta en titt på videoløsningsforslaget som er linket ved siden av ALLE innføringsoppgavene om det blir vanskelig! Det viktigste er at du skriver koden selv.*
  *NB: Om du benytter løsningsforslaget veldig tungt på en oppgave så er det anbefalt å prøve seg igjen på den på et senere tidspunkt.*

**Samling av nettforelesninger til heftet:**

http://pythonhefte01.studenthjelp.no/

# *Innholdsfortegnelse:*





# 1.1 Kommentarer, print()-funksjonen og feilmeldinger

http://pythonhefte0101.studenthjelp.no/

Når vi skriver programkode i Python så ønsker vi å gjøre den forståelig for oss selv og andre som leser koden i ettertid. Det er derfor nyttig å kunne skrive forklarende tekst ved siden av koden i programmet. Det å kunne skrive **kommentarer** er derfor viktig. Når du skriver en kommentar skriver du en tekst i koden som ikke utføres som vanlig programkode.

## To typer kommentarer:

### Hash (#)

For å skrive en kommentar i Python så bruker vi normalt å sette hash (**#)** foran det vi ønsker å kommentere. Dersom vi skriver **#** så vil Python ikke lenger lese det som kommer etter på samme linje i koden.

### Trippel anførselstegn (""")

En annen måte å kommentere på er ved å bruke **trippel anførselstegn(""")** foran og bak det du skriver. Gjør du dette kan du kommentere over flere linjer.

**Eksempel 1**


*Her er ett eksempel på hvordan vi kan bruke kommentarer. Om vi kjører programmet under vil det ikke komme noen utskrift. Dette er helt naturlig ettersom alt vi skriver er kommentarer.*

**Kode:**

```
# En «hash» gjør at alt skrevet etter på samme linje regnes som en kommentar

"""
Om vi skriver tre (") på rad forran og bak en tekst så kan vi lage
kommentarer over flere linjer
"""
```

**Oppgave 1.1 (kommentar.py)**

Det er nå viktig at du har en Python-editor installert. **Bruk programnavnet foreslått over** når du lagrer programmet.

**NB:** Dette programmet vil ikke ha noen utskrift når du kjører det. Følg QR-koden til høyre eller linken like under for en **videogjennomgang av oppgaven.**

**a)** Skriv **# Python leser ikke kommentarer** på den første linjen i programmet ditt. (**NB: Husk å kjøre programmet**)

**b)** Lag en kommentar over flere linjer ved å bruke **trippel anførselstegn (""")** som i eksempel 1. Skriv din egen tekst eller bruk tekstforslaget til høyre.

**Tekstforslag:**

*"Jobb hardt, strev imot å være ditt beste selv. Kan du kanskje være et lys for deg selv og de rundt deg?"*

**Løsningsforslag:**

http://pyløs0101.studenthjelp.no/

## print()-funksjonen:

*print()* funksjonen er utrolig viktig når du skriver i Python. Det er nemlig denne funksjonen du vil bruke til å få en utskrift i ***konsollen***. *print()* funksjonen kan ta ett eller fler *"argumenter"/"verdier"* inn samtidig Ved fler argumenter setter du komma imellom.

### EKSEMPEL 2


**Kode:**

```
print("Hallo verden")  # Vi printer Hallo verden i konsollen.

print(42)              # Siden 42 er ett heltall og trenger derfor ikke "

print("Jeg er", 18, "år") # Setter vi komma kan vi få fler ting på samme linje
```

**Utskrift i konsollen:**

```
Hallo verden
42
Jeg er 18 år
```

### Oppgave 1.2 (print.py)

**a)** Skriv **print("Jeg har laget mitt første program")** i kodefeltet. Husk anførselstegn inne i **print()**-funksjonen. Prøvekjør programmet.

**b)** Bruk **print()**-funksjonen til å skrive ut tallet **71**. Her trenger du ikke bruke anførselstegn siden det kun er et heltall.

**c)** Nederst i eksempelet over ser du at vi har brukt fler "argumenter" inne i **print()**-funksjonen. Skriv én **print()**-funksjon med **"3 + 2 ="** og **5** som argumentene. Husk å sette komma imellom argumentene.

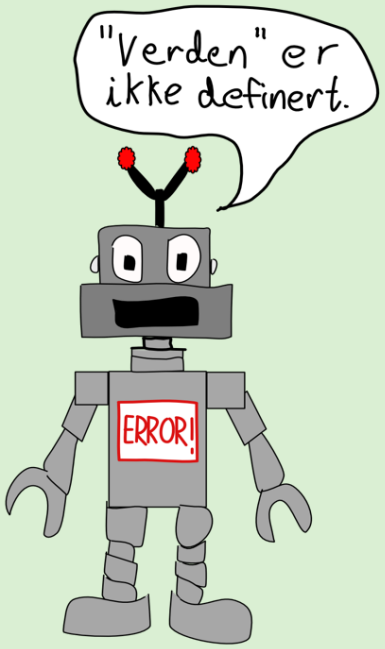
**Løsningsforslag:**

http://pyløs0102.studenthjelp.no/

## Feilmeldinger

Når du lager din egen kode så vil du utvilsomt møte på feilmeldinger. Det kan virke skummelt å møte på en feilmelding første gang, men vit at det er helt naturlig å kode feil en gang i blant. Dersom du får en feilmelding så vil oftest Python fortelle deg akkurat hvor feilen er i koden din.

### EKSEMPEL 3




**Kode:**

```
# Første bit av koden er det ikke noe galt med.
print("Hei")
# I koden under har jeg lagt inn en feil.
print(Verden)
```

*Her er det verdt å merke seg at koden i Python går kronologisk ifra topp til bunn. Det vil si at denne koden kjører helt frem til den møter på feilen.*
*I utskriften under vil du derfor se at det står **"Hei"** før vi får en feilmelding på grunn av manglende anførselstegn. Når programmet møter på en slik feil så kjøres ikke koden som står etter uansett om den er helt riktig.*

**Utskrift i konsollen:**

```
Hei
Traceback (most recent call last):

  File "/Users/danielaubert/untitled0.py", line 4, in <module>
    print(Verden)

NameError: name 'Verden' is not defined
```

*Vi kan se i utskriften at programmet har en feil i "**line 4**" og at feilen er at **"Verden"** ikke er definert. Det er ikke alltid like lett å tolke alle feil, men prøver du deg frem går det som oftest veldig bra!*

***NB: Når du programmerer i Python er det veldig viktig å skrive koden helt perfekt. Om du ikke skriver koden akkurat slik den skal være vil det normalt ende opp i feilmeldinger eller rare utskrifter.***

### Oppgave 1.3 (feilmeldinger.py)

**a)** Kopier koden som står under **Kode** i eksempelet ovenfor. Kjøre koden og pass på at du får samme utskrift. (Inkluder alt i koden)
**NB:** Du har nå fått en **NameError**. Denne feilmeldingen kommer vanligvis når noe ikke har blitt riktig **definert.** Hva å definere noe betyr kommer vi tilbake til i neste delkapittel.

**b)** Vi skal nå rette opp koden vi skriv i oppgave **a)** dette gjør vi ved å sette anførselstegn rundt **Verden.** Det skal altså stå **print(*"Verden"*).** Kjør programmet og pass på at du nå får en utskrift uten feilmeldinger.

**c)** Prøv å fjerne **#** i den øverste kommentaren du kopierte i oppgave **a)** for så å kjøre programmet. Du får nå enten en **SyntaxError** eller en **IndentationError.**

- Om du får en **IndentationError** så er det fordi du har et mellomrom imellom starten av linjen og koden din. Et slikt mellomrommet har en spesiell oppgave i Python som vi skal komme tilbake til senere.
- Du får en **SyntaxError** når Python ikke klare å tolke det du skriver.

**Løsningsforslag:**

http://pyløs0103.studenthjelp.no/

# 1.2 Variabeltyper/Datatyper og tilordning av verdier

## Datatyper

http://pythonhefte0102.studenthjelp.no/

I Python har vi flere vanlige *datatyper*. De mest brukt i matematikk pensum er:

| VARIABELTYPER/ DATATYPER | EKSEMPLER |
|---|---|
| | |
| **Heltall** → *På engelsk integer eller **int** forkortet.* | **2, -3, 200** |
| **Flyttall** → *Desimaltall, på engelsk floating point number eller bare **float**.* | **0.125, 2.355** |
| **Strengverdier** → En tekststreng, *på engelsk string eller **str** forkortet.* | **"Hei", "God morgen fru Nilsen"** |
| **Lister** → En datatype som samler data. I en liste kan det være flere forskjellige datatyper samtidig. | **[2,3,4,5,6]** En liste med kun heltall<br>**[3, "Hei", 3.0]** En liste med fler datatyper |
| **Arrayer** → *Likner lister, men er mer egnet for matematiske operasjoner.* | Se kapittel 2 |
| **Boolske verdier** → *På engelsk boolean, **bool** forkortet.* | **True, False** |

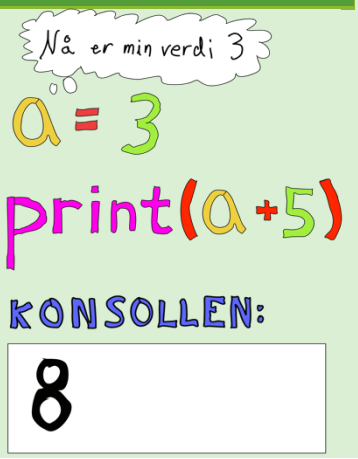
## Variabler

Ved å skrive **a = 2** så har vi opprettet vår første variabel i Python. Her har vi tilordnet (gitt) **variablene a** heltalls verdien **2**. Om vi da senere i programmet skriver **a** for seg selv (og ikke for eksempel i en tekststreng) så vet Python at vi egentlig mener **2** og programmet vil handle deretter. Vi kan også gi en variabel en verdi av en annen datatype. Skriver vi **b="Jeg er en tekst"** så har vi opprettet *streng objektet* med verdi **"Jeg er en tekst"**.

**For listetegn: [**
**MAC: option + 8**
**PC: alt gr + 8**

**For listetegn: ]**
**MAC: option + 9**
**PC: alt gr + 9**

### EKSEMPEL 4




**Kode:**

```
a = 2       # Vi tilegner (altså gir) variabelen a heltallsverdien 2
print(a)    # Vi ber Python om å printe verdien til a (altså 2)

b = 5.0     # Vi gir b flyttallsverdien 5.0
print(b)    # Vi ber Python om å printe verdien til b (altså 5.0)

c = "Jeg er en tekst" # c er nå et "streng objekt" med verdi "Jeg er en tekst"
print(c)    # Vi printer verdien til c ("Jeg er en tekst")

d = ["hei", 3, 5.7] # Vi lager en liste med fler objekttyper
print(d)    # Vi printer liste-objektet
```

**Utskrift i konsollen:**

```
2
5.0
Jeg er en tekst
['hei', 3, 5.7]
```

Det som står på høyre siden av likhetstegnet legges inn i variabelen til venstre.

### Oppgave 1.4 (variabler.py)

**a)** Lag et program hvor du gir variabelen **a** heltallsverdien 4 (skriv **a = 4**), **b** flyttalsverdien **47.35**, og **c** strengverdien **"Dette er gøy"**.
**b)** Under dette lag en ny variabel **d = [a,b,c]** (et listeobjekt).
**c)** Bruk **print()** på **a**, **b**, **c** og **d** hver for seg. (Skriv **print(a)**, **print(b)** etc.)

**Løsningsforslag:**

http://pyløs0104.studenthjelp.no/

# 1.3 Regneoperatorer

## Pluss, minus, gange og dele

http://pythonhefte0103.studenthjelp.no/

Om du hadde mistanker om at de vanlige regneoperasjonene ville være enkle å gjennomføre i Python, ja da hadde du rett!

EKSEMPEL 5


**Kode:**

```
a = 3
b = 5

c = a + b          # Lager en ny variabel c lik summen av a og b (3+5)
print(c)           # Vi gir Python beskjed om å skrive ut svaret

c = a - b          # Gir c en ny verdi lik a - b (3-5)
print(c)

c = a * b          # Gir c en ny verdi lik a * b (3*5)
print(c)

c = a/b            # Gir c en ny verdi lik a/b (3/5) (c blir nå et flyttall)
print(c)
```

**Utskrift i konsollen:**

```
8
-2
15
0.6
```

**Oppgave 1.5 (regneoperatorer.py)**

**a)** Sett **a = 4** og **b = 9**.
**b)** Gi **c** verdien **a + b** (altså skriv **c = a + b**)
**c)** Skriv nå **print(c)** og kjør programmet for å se resultatet.
**d)** Gi nå **d** verdien **a – b**, **e** verdien **a*b** og gi **f** verdien **b/a.**
**e)** Bruk **print()**-funksjonen på **d**, på **e** og på **f**.

**Løsningsforslag:**

http://pyløs0105.studenthjelp.no/

## Å legge sammen og gange streng-elementer

Vi kan faktisk også bruke pluss (**+**) og gange(*****) på tekststrenger! Om vi for eksempel skriver **"hal"+"lo"** får vi **"hallo"** og om vi skriver **"ha"*3** for vi **"hahaha"**. Vi kan ikke bruke minus og dele operatorene på tekststrenger.

**Oppgave 1.6 (streng_pluss_gange.py)**

**a)** Sett **a = "Å legge sammen", b = " "** (her er det et mellomrom imellom anførselstegnene), **c = "tekststrenger er litt rart."**
**b)** Skriv **print(a+c)** og **print(a+b+c)** på hver sin linje og kjør programmet.
**c)** Skriv nå **print(5*c)** og kjør programmet.
**d)** Prøv nå å skrive **print("Nå"+"programmerer"+"jeg.")**, kjør programmet. Gjør deretter nødvendige endinger for å få en penere utskrift.

**Løsningsforslag:**

http://pyløs0106.studenthjelp.no/

# Eksponenter (**), heltallsdivisjon (//) og modulo(%)

Eksponenter bør du allerede være kjent med, men modulo og heltallsdivisjon er kanskje nytt?

Om du har lært deg å dele så har du kanskje lært denne måten å tenke på: Eksempel "Vi har $13 \div 3$, hvor mange ganger går $3$ i $13$. $4 \cdot 3 = 12$ så $3$ går $4$ ganger i $13$. Vi får deretter $1$ i rest.

## EKSEMPEL 6


*Hva er **$3^4$**? Hvor mange hele ganger går 2 i 7? Og hva blir i så fall resten?*

**Kode:**

```
# Eksponential funksjoner 3^4
print(3**4)     # 3^4 = 3*3*3*3 = 81
# Heltallsdivisjon
print(7//2)     # Hvor mange ganger går 2 i 7. Svaret er 3
# Resten ved heltallsdivisjon (modulus)
print(7%2)      # Hva blir resten når vi heltallsdividerer 7 på 2. Den blir 1.
```

**Utskrift i konsollen:**

```
81
3
1
```

### Oppgave 1.7 (eksponenter_heltall_mod.py)

**a)** Skriv **print(5**3)** og **print(2**8)** for å finne ut hva **$5^3$** og hva **$2^8$** blir.

**b)** Skriv **print(9 // 4)** for å finne ut av heltallsdivisjonen **9//4** blir.

**c)** Bruk **print(9 % 4)** for å finne ut **resten** i divisjonen over (altså modulo)

**Løsningsforslag:**

http://pyløs0107.studenthjelp.no/

### Oppgave 1.8 (konsolloppgave)

Det går også an å programmere direkte i konsollen. Dette gjøres noen ganger når vi ønsker å teste kode vi ikke har brukt før, eller om vi kun ønsker å gjøre noe raskt. Vi programmerer nå også. Forskjellen er at koden kjøres hver gang vi skriver en ny linje og Python husker det som har blitt kjørt tidligere i konsollen.

**a)** Skriv **print("*Hallo verden*")** inn i konsollen. Trykk ENTER. Da vil koden kjøre og du vil se utskriften **Hello Verden** i konsollen.

**b)** Skriv **5 + 3** inn i konsollen. Trykk ENTER. Vi trenger ofte ikke å bruke **print()**-funksjonen i python for å få et svar i konsollen.

**c)** Skriv **a = 5** i konsollen. Trykk ENTER. Skriv så **a + 5.** Trykk ENTER. Her kan vi se at **a** fremdeles har verdien **5** også når vi kjører neste linje kode.

**d)** Finn ut hva $4^2$ (altså **4**2**) blir ved å bruke konsollen.

**e)** Hva blir 20 heltallsdividert med 3 for noe? (altså **20//3**). Bruk konsollen

**f)** Hva blir restleddet i heltallsdivisjonen ovenfor? (altså **20%3**)

**Løsningsforslag:**

http://pyløs0108.studenthjelp.no/

# *1.4 input()-funksjonen og endring av datatyper*

http://pythonhefte0104.studenthjelp.no/

## Endring av datatyper

I Python-programmering så er "datatypene" viktige. Det er forskjell på hvordan programmet vil behandle **"26"** (Et strengobjekt) og **26** (Et heltall). Siden vi noen ganger ønsker å (for eksempel) gi informasjon (input) til programmet vårt i tekstform (altså streng-form) så er det nyttig med egne funksjoner som kan forandre denne typen data til noe Python kan bruke bedre.

| | |
|---|---|
| **int()** → Gjør om et flyttall eller en tekststreng om til et heltall (integer) | **int("43")** gjør om tekststrengen **"43"** til heltallet **43.**<br>**int(50.65)** gjør om flyttallet **50.65** til heltallet **50** (tallene rundes altså ned) |
| **float()** → Gjør om et heltall eller en tekststreng om til et flyttall (tall med desimal) | **float("32.52")** gjør om tekststrengen **"32.52"** til flyttallet **32.52** |
| **str()** → Gjør om et objekt (tall eller annet) om til en tekststreng. | **str(34)** gjør om heltallet **34** til tekststrengen **"34"** |

### EKSEMPEL 7


**Kode:**

```
a = "5"
b = 6.3
c = 2
d = ["hei","på", "deg"]

print(int(a))        # Tekst-strengen ("5") lagret i a blir til helttall (5)

print(int(b))        # Flyttallet lagret i b (6.3) blir til heltallet 6

print(float(a))      # Tekst strengen ("5") lagret i a blir til flyttall (5.0)

print(str(d))        # Listen lagret i d blir printet ut som en tekststreng
```

**Utskrift i konsollen:**

```
5
6
5.0
['hei', 'på', 'deg']
```

### Oppgave 1.9 (dataendring.py)

**a)** Lag et program der du gir variabelen **a** streng-verdien **"42"** og **b** flyttallsverdien **3.1.** Skriv **print(a + b).** Når du kjører programmet vil du nå få en **TypeError** fordi vi ikke kan legge sammen strengobjekter og flyttall.

**b)** Du skal nå bruk **int()** til å gjøre om **a** til et heltall. Skriv **a = int(a)** i en egen kodelinje etter kodelinjen **a = "42"**, men før **print(a+b)**. Om du har gjort det riktig bør nå kommandoen **print(a + b)** gi et svar.

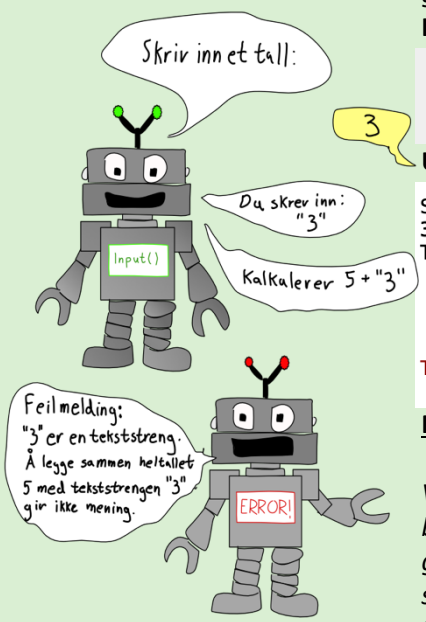
**Løsningsforslag:**

http://pyløs0109.studenthjelp.no/

## input() funksjonen:

Vi skal nå lage programmer hvor brukeren har mulighet til å gi informasjon til programmet. *input()*-funksjonen gjør dette mulig! Skriver vi: **a = input("Skriv inn en tekst: ")** så vil programmet skrive: **Skriv inn en tekst:** i konsollen. Programmet vil deretter stoppe opp og vente på vår input før det fortsetter å kjøre. Vi beveger musen vår dit vi skal skrive inn informasjonen og klikker. Deretter skriver vi inn det vi ønsker å gi av informasjon og trykker ENTER. Informasjonen vi har gitt til programmet vil nå lagres i *variabelen* **a**.
**NB: input()**-funksjonen tar bare inn **"streng-objekter"** (altså tekst).

### EKSEMPEL 8




*For å forstå er det best å se på et eksempel. Først skal vi se på et eksempel der noe går galt fordi vi glemmer å forandre datatype.*

**Kode:**

```
 9    a = 5                                  # Vi gir variabelen a heltallsverdien 5.
10    b = input("Skriv inn et tall: ")       # Vi ber brukeren om en verdi.(Streng)
11    print(b)                               # Skriver ut det vi nettopp skrev inn.
12    print(a+b)                             # Vi prøver å summere a og b
```

**Utskrift i konsollen:**

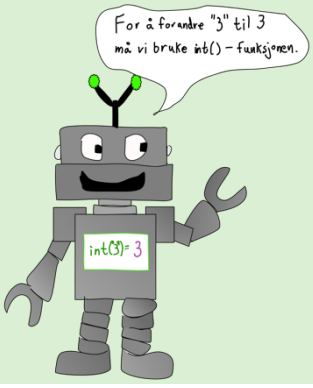
```
Skriv inn et tall: 3
3
Traceback (most recent call last):
  File "/Users/danielaubert/sammenlikning.py", line 12, in <module>
    print(a+b)                             # Vi prøver å summere a og b

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

NB: Her MÅ du skrive fysisk inn i konsollen før programmet kan fortsette å kjøre.

**Feilmelding:** *(Dette kan du komme til å støte på litt for ofte, så følg med)*

*Vi lar programmet kjøre og velger å skrive inn tallet **3** og trykk **Enter**. Det vi skriver blir lagret i variabelen **b** som en **tekst-streng**. Det neste vi har bedt programmet om å gjøre er å **skrive ut b**. Det klarer programmet helt fint ettersom det klarer å skrive ut streng objekter. Men da vi ba programmet om å legge sammen **a** og **b**. Altså **heltallet 5** (int) og **tekststrengen "3"** (str). Så visste ikke Python lenger hva vi ville og valgte å kjøre en feilmelding. Legg merke til at Python forteller oss hvor feilen ligger "**line 12**".*

### EKSEMPEL 9




*Om vi ønsker at det vi skriver inn i **input()** funksjonen skal gjøres om til et heltall må vi bruke **int()**-funksjonen på variabelen vi lagde med **input()**-funksjonen slik vist under:*

**Kode:**

```
a = 5                                      # Vi gir variabelen a heltallsverdien 5.
b = input("Skriv inn et tall: ")           # Vi ber brukeren om en verdi.(Streng)
b = int(b)                                 # Vi gjør om inputten til heltall
print(b)                                   # Skriver ut det vi nettopp skrev inn.
print(a+b)                                 # Vi finner summen av a og b
```

**Utskrift i konsollen:**

```
Skriv inn et tall: 3
3
8
```

### Oppgave 1.10 (input_kopi.py)

Kopier det ferdige programmet i eksempelet ovenfor inn i et eget kodevindu. Du trenger ikke å kopiere kommentarene. Kjør så programmet. Skriv inn tallet 3 til høyre for **Skriv inn et tall:** trykk deretter ENTER.
Du har fullført oppgaven når utskriften i konsollen din er lik den i eksempelet.

**Løsningsforslag:**

http://pyløs0110.studenthjelp.no/

### Oppgave 1.11 (input_selv.py)

NB: Du skal nå skrive selv og altså ikke kopiere programmet i forrige oppgave.

- **a)** Sett **a = input("Skriv inn ett heltall: ")** og **b = 3.2**
- **b)** Skriv nå **a = int(a)** under det du skrev i oppgave **a)** for å gjøre om strengobjektet som blir lagret i **a** om til et heltallsobjekt som kan brukes til å regne med.
- **c)** Skriv **print(a+b)** nederst i koden din og kjør så programmet. Husk at du må skrive inn ett tall og trykke **ENTER** når koden spør deg om et tall.
- **d)** Gjør en endring i programmet slik at **b = input("Skriv inn ett flyttall:")** istedenfor **3.2**. Prøv å kjøre programmet, du vil nå få en feilmelding etter du har skrevet inn egenvalgte tall i konsollen. Hvorfor?
- **e)** For å løse problemet ifra oppgave **d)** gjør nok en endring på **b** og skriv **b = float(b)** i linjen like under **b = input("Skriv inn ett flyttall:").** Her er det viktig at denne linjen kommer før **print(a+b).**

**Løsningsforslag:**

http://pyløs0111.studenthjelp.no/

### Oppgave 1.12 (input_fruktkiosk.py)

*Knut jobber i en kiosk som selger epler og bananer. Eplene koster **60 kr/kg** og bananene kun **45 kr/kg**. Knut veier frukten ved disken, men han tuller hele tiden med å huske alle prisene. Han har bestemt seg for å løse problemet ved å be deg lage et lite program som gjør jobben hans enklere. Programmet skal motta informasjon om hvor mange kilo epler og hvor mange kilo bananer en kunde ønsker å kjøpe. Programmet skal deretter returnere hvor mye det koster for eplene, hvor mye det koster for bananene og hvor mye det koster til sammen. (På en oversiktlig måte)*

**NB:** Under beskrives en mer skritt for skritt oppgavedeling på denne problemstillingen. Men om du ønsker kan du allerede nå prøve deg på denne oppgaven (Dette vil øke utfordringen og derfor læringen). Du oppfordres uansett til å ta en titt under når du er ferdig.

- **a)** Lag variabler **pris_kg_eple** og **pris_kg_banan** der du lagrer relevante kilopriser.
- **b)** Bruk **input()**-funksjonen til å motta antall kilo epler og antall kilo bananer legg informasjonen i variablene **kg_eple** og **kg_banan. NB: Husk å gjøre om til flyttall.**
- **c)** Sett **kostnad_epler = kg_eple*pris_kg_eple** og gjør tilsvarende med banan.
- **d)** Skriv **print("Eplene koster til sammen:", kostnad_eple)** for å få en svarsetning for hvor mye eplene koster. Gjør det samme for bananer og den samlede prisen.

**Løsningsforslag:**

http://pyløs0112.studenthjelp.no/

# *1.5 Sammenlikningsoperatorer og if/else – setninger*

http://pythonhefte0105.studenthjelp.no/

Vi ønsker ofte å be programmet gjøre en bestemt handling dersom **(if)** noe er "sant" **(True)** og noe annet ellers **(else)**. Eksempelvis kan vi ønske at programmet skal skrive ut **"Elise har nok penger til å kjøpe sokker"** dersom **(if) x > 40** og **"Elise har dessverre ikke nok penger til å kjøpe sokker"** om ikke vilkåret er oppfylt altså ellers **(else)**. Her bruker vi **sammenlikningsoperatorer** og **if/else** setninger.

| DE VANLIGSTE SAMMENLIKNINGSOPERATORENE | EKSEMPEL IFRA KONSOLLEN |
|---|---|
| == → Sjekker om variabler, tall eller annet **er like** hverandre. | `In [1]: 2 == 2`<br>`Out[1]: True` |
| != → Sjekker om noe **ikke er likt** noe annet og gir False kun hvis noe er likt. | `In [2]: 2 != 3`<br>`Out[2]: True` |
| < → Sjekker om noe er **mindre enn** noe annet. | `In [3]: 5 < 2`<br>`Out[3]: False` |
| > → Sjekket om noe er **større enn** noe annet | `In [4]: 5 > 2`<br>`Out[4]: True` |
| <= → Sjekker om noe er **mindre enn eller lik** noe annet. | `In [5]: 2 <= 2`<br>`Out[5]: True` |
| >= → Sjekker om noe er **større eller lik** noe annet. | `In [6]: 3 >= 3`<br>`Out[6]: True` |

***Når vi bruker en sammenlikningsoperator ender vi opp med ett av to mulige svar: "True" eller "False". Altså en boolsk verdi.***

**Oppgave 1.13 (konsolloppgave)**

I denne oppgaven skal du bruke Python-konsollen for å gjennomføre.

**a)** Skriv inn 3==4 i konsollen. Se hva slags resultat du nå får. Prøv deretter igjen med 3==3.

**b)** Prøv å finn ut om de følgende utsagnene gir TRUE eller FALSE uten å skrive de inn i konsollen. Sjekk svarene dine ved å skrive de inn i konsollen.
**1)** "Hei" != 3
**2)** 5 !=9
**3)** 4 > 4
**4)** 4 <= 8
**5)** 99 != 98+1
**6)** 100 + 33 >= 98+35

**Løsningsforslag:**

http://pyløs0113.studenthjelp.no/

## EKSEMPEL 10


*Vi prøver å bruke det vi har lært for å sjekke noen påstander:*
*Er påstandene under sanne eller falske?*

*a)* $3 \cdot 5 - 10 = 5$
*b)* $4^2 \geq 15$
*c)* $-16 \neq -16.0$

**Kode:**

```
print(3*5-10 == 5)    # Er 3 ganger 5 minus 10 lik 5?
print(4**2 >= 15)     # Er 4^2 større eller lik 15?
print(-16 != -16.0)   # Er heltallet -16 ulikt flyttallet -16.0?
```

**Utskrift i konsollen:**

```
True
True
False
```

*Som vi ser er de to første påstandene sanne (altså stemmer de) imens den siste påstanden gir False fordi -16 og -16.0 faktisk er helt like.*



### Oppgave 1.14 (sammenlikning.py)

Lag et program der du sjekker om følgende påstander er sanne eller falske:

**NB: Husk at målet er å lære å skrive programmet. Gjør derfor fremdeles programmeringen selv om svaret er åpenbart for deg.**

**a)** $3 + 3 = 6$
**b)** $9^2 = 88$
**c)** $5 < 3 \cdot 3 - 2$
**d)** $4^2 \neq 3 \cdot 5 + 1$
**e)** $-11 \leq (-5)^2$
**f)** $6^3 \geq 333$

Vi kan også sjekke andre ting med sammenlikningsoperatorene. For eksempel om 2 ord er like hverandre. Dette kan virke overflødig, men er nyttig når vi for eksempel benytter oss av setninger. Prøv deg på følgende oppgaver:

**g)** "Ja" == "Ja"
**h)** "$Nei$" == "Ja"
**i)** "Ja" == "ja"
**j)** Hvorfor tror du oppgave i) ga FALSE?

### Løsningsforslag:

http://pyløs0114.studenthjelp.no/

# if/else– setninger

**if-** og **else** -setninger er grunnleggende innenfor programmering. Mye av programmeringen du skal lære senere i dette heftet baserer seg på nettopp **if** -og **else-setninger.**

## EKSEMPEL 11

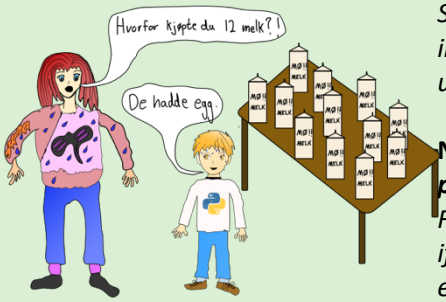

*Den første **if-setningen** sjekker om påstanden ved siden av er sann (**True**), hvis den er det så utføres koden som står under **if.** Koden som er under **else** vil kun bli utført dersom koden under **if** ikke blir utført.*

**Kode:**

```
x = 2

if x == 3:
    print("Yes")
else:
    print("No")
```

**Utskrift i konsollen:**

```
No
```




*Siden påstanden i dette tilfellet åpenbart ikke er sann (**x** er jo lik 2 ikke 3) kommer ikke koden under **if** til å bli utført. Programmet sjekker da hva som står under **else** og utfører denne koden istedenfor.*

**NB:** *Legg merke til at vi har et mellomrom like under **if**-testen før vi skriver **print("Yes").** Dette kommer automatisk etter vi skriver kolon(:) og trykker **ENTER**. Formålet er at programmet skal skille det som skal utføres dersom if-testen er "true" ifra resten av programmet. Før vi skriver **else** må vi fjerne dette mellomrommet. (Se eksempel over) Under **else** vil vi få et nytt mellomrom.*

### Oppgave 1.15 (if_else_kopi.py)

**a)** Skriv av programmet i eksemplet over. Kjør programmet og pass på at utskriften blir som den skal.

**b)** Gjør en endring i **x** slik at programmet nå skriver ut **Yes.**

**Løsningsforslag:**

http://pyløs0115.studenthjelp.no/

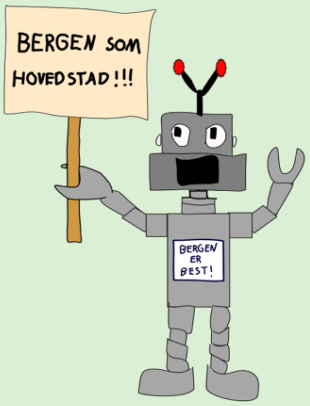
### Oppgave 1.16 (if_else_vq.py)

**a)** Sett **v = 400** og **q = 900.**
**b)** Lag en **if-setning** der koden: **print("q er størst")** utløses dersom **q > v**. NB: Her er det greit å merke seg at vi ikke må ha en **else-setning** som utløses om det ikke er sant. Kjør programmet.
**c)** Legg til en **else-setning** som utløser koden: **print("q er mindre enn eller lik v")** dersom ikke **if setningen** utløses.
**d)** Kjør programmet for forskjellige verdier av **v** og **q.**

**Løsningsforslag:**

http://pyløs0116.studenthjelp.no/

## EKSEMPEL 12




*Det vi sammenlikner må ikke nødvendigvis være tall. Vi kan for eksempel også sjekke om **x** er en spesifikk streng, liste eller hva som helst annet.*

**Kode:**

```
x = input("Hva er hovedstaden i Norge?: ")

if x == "Oslo":          # Legg merke til vi skriver stor bokstav i navn
    print("Riktig!")     # Skriver du oslo som svar får du derfor ikke riktig
else:
    print("Dine geografikunnskaper er noe manglende")
```

**Utskrift i konsollen:**

```
Hva er hovedstaden i Norge?: Bergen
Dine geografikunnskaper er noe manglende
```

### Oppgave 1.17 (if_fin_dag.py)

**a)** Sett **x = "Ja"**, lag deretter en **if-setning** som utløser koden: **print("Det var godt å høre.")** dersom **x == "Ja"**.

**b)** Lag en **else-setning** som utløser koden **print("Det var trist.")** dersom **if-setningen** ikke utløses. (Altså ellers)

**c)** Erstatt nå **x = "Ja"** med **x = input("Har du en fin dag? Svar: ")**. Kjør programmet. Husk å skrive inn Ja i konsollen med stor forbokstav når du prøver å utløse **if-setningen**.

**Løsningsforslag:**

http://pyløs0117.studenthjelp.no/

### Oppgave 1.18 (penger.py)

**a)** Sett **penger = 200** og **pris = 150**.

**b)** Lag en **if-setning** som utløses dersom **penger > pris**. Den skal utløse koden: **print("Einar har råd til den dyreste fotballen og har ", penger – pris, " kroner igjen.")**

**c)** Lag en **else-setning** som utløses dersom ikke **if-setningen** utløses. Koden som utløses skal være: **print("Einar mangler ", pris-penger," kroner for å ha råd til fotballen.")**

**d)** Kjør programmet for forskjellige verdier av **penger** og **pris**.

**e)** I starten av koden din bruk en **print()-funksjon** slik at programmet skriver ut følgende: **Lille Einar skal på lekebutikken for å kjøpe seg en fotball. Han får litt penger av foreldrene sine. Han er ganske liten og vet verken hva de forskjellige fotballene koster eller hvor mye han har. Han går med en fotball og pengene til kassen.**

**f)** Erstatt nå **penger = 200** med **penger = int(input("Mannen i kassa ser at Einar har fått (Skriv inn antall kroner): "))**
Du kan nå skrive inn beløpet Einar har fått selv.

**g)** Erstatt nå **pris = 150** med **pris = int(input("Ballen har en pris på (Skriv inn antall kroner): "))**
Prisen på ballen kan du nå også bestemme selv.

**Løsningsforslag:**

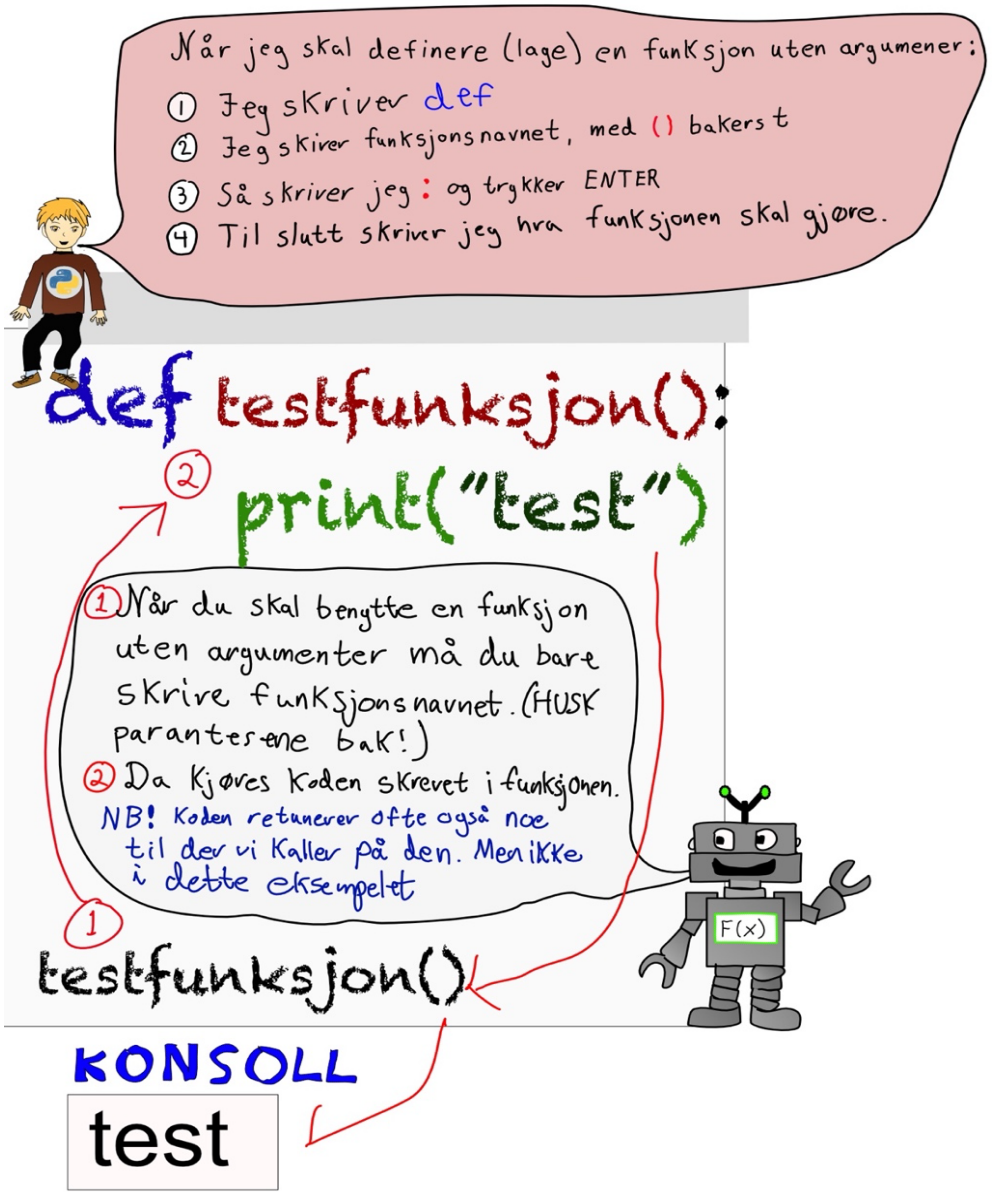
http://pyløs0118.studenthjelp.no/

# 1.6 Funksjoner

## Hvordan lage en funksjon i Python

http://pythonhefte0106.studenthjelp.no/

Vi skal nå lære å definerer funksjoner i Python. Vi begynner med en funksjon **uten argumenter**.




**Oppgave 1.19 (funksjon_test.py)**

Les det som står ovenfor nøye og lag en kopi av programmet. Pass på at du ikke får noen feilmeldinger når du kjører programmet.

**Løsningsforslag:**

http://pyløs0119.studenthjelp.no/

Ikke fortvil dersom du ikke får med deg alt i eksempelet under. Det er mye å lære ved å gå igjennom dette eksempelet en gang først for så å komme tilbake til det etter å ha gjort litt mer av delkapittelet.

## EKSEMPEL 13

Vi skal nå lage en funksjon som skal returnere en gladmelding til der vi kaller på funksjonen i tekst/streng form. Den skal ikke skrive det direkte ut slik som ovenfor (legg merke til at ovenfor er print()-funksjonen inne i funksjonen vi kaller på) men først sende tekststrengen tilbake til der vi kaller på funksjonen ved hjelp av en return kommando.

1 *Vi skriver først **def** (for definer) deretter mellomrom og så **funksjonsnavnet** som i dette tilfellet er **gladmelding**. Like etter **gladmelding** har vi to parenteser **()**.*

*NB: Inne i parentesene står vanligvis **argumentene** til funksjonen. Denne funksjonen skal ikke ta imot argumenter. Vi lar derfor parentesene stå tomme. Vi må likevel ha med parentesene så Python gjenkjenner at det vi lager er en funksjon.* 2

**Kode:**

```
def gladmelding():
    resultat = "Du kan allerede masse programmering!"
    return resultat

print(gladmelding())
```

*Som i **if-testen i kapittel 1.5** har vi her et mellomrom på de neste linjene. Det mellomrommet er der for å skille det funksjonen skal utføre ifra resten av programmet.*

*Vi skriver en tekststreng med en gladmelding og lagrer gladmeldingen i **resultat**. Til sist skriver vi **return resultat**. Tekststrengen i resultat sendes nå til der vi kaller på funksjonen.* 3

*Når vi til slutt skriver **gladmelding()** "kaller" vi på funksjonen **gladmelding()**. Koden i **gladmelding()** vil da gjennomføres og tekststrengen lagret i **resultat** vil da settes inn der vi kaller på funksjonen. Siden vi vil at tekststrengen som returneres skal skrives ut til konsollen må vi kalle på funksjonen inne i en **print()**-funksjon.* 4

*NB: Når vi kaller på **gladmelding()** må vi skrive **()** etter funksjonsnavnet. Dette forteller nemlig Python at vi ønsker funksjonen **gladmelding()** og ikke en eventuell variabel tidligere definert som **gladmelding**.* 5

**Utskrift i konsollen:**

```
Du kan allerede masse programmering!
```

***Oppskrift for å definere funksjoner i Python:***

```
def funksjonsnavn(argument1,argument2,...):   # En funksjon kan ta imot ingen, ett eller fler argumenter.
    Handlingen i funksjonen spesifiseres.     # Det funksjonen faktisk skal gjøre beskrives her.
    return resultat                           # Her spesifiserer vi hva funksjonen skal sende tilbake.
```

***Merk:** Når vi trykker ENTER etter kolon(:) vil vi automatisk på neste linje havne litt ut på linja. Dette er slik det skal være og markerer at det vi skriver er inne i funksjonen. Når vi er ferdig å skrive funksjonen fjerner vi dette mellomrommet på linjestarten. Da er vi ute av funksjonen.*

***Merk også**: Dersom vi prøver å kalle på funksjonen "før" vi har definert den vil vi få en feilmelding.*

### Oppgave 1.20 (funksjon_test2.py)

**a)** Kopier koden i eksempelet ovenfor og kjør programmet.
**NB:** Dersom du fjerner **print()** funksjonen rundt **gladmelding()** så vil fremdeles programmet kjøre, men det vil ikke komme noen utskrift.
**b)** Istedenfor **return resultat** skriv **return "Du er superflink"** og kjør programmet.

**Løsningsforslag:**

http://pyløs0120.studenthjelp.no/

**Oppgave 1.21 (funksjon.py)**

**a)** Definer en funksjon du kaller **femplussto()**. Husk kolon (:)
**b)** I funksjonen skal du sette **resultat = 5 + 2**
**c)** Returner resultat.
**d)** Kall på funksjonen inne i en **print()**-funksjon.

**Løsningsforslag:**

http://pyløs0121.studenthjelp.no/

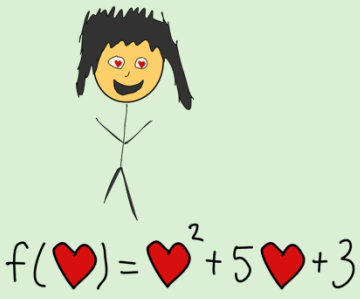
## En funksjon med ett argument

F(**x**) = **x** * 🍎 + 2 * 🍎 (altså **x** epler + 2 epler)
Antall epler vi ender opp med er avhengig av verdien vi setter inn for **x** i funksjonen.

Om vi **erstatter x** i funksjonen med **3** eller med **10**, hvor mange epler gir funksjonen oss da?
F(**3**) = **3** * 🍎 + 2 * 🍎 = 5 🍎 # Lar vi x (altså argumentet i funksjonen) være 3 får vi 5 epler.
F(**10**) = **10** * 🍎 + 2 * 🍎 = 12 🍎 # Lar vi x være 10 får vi 12 epler.

Her sier vi at **F** er en funksjon som er avhengig av variabelen/argumentet **x**. Vi kaller **x** i funksjonen en variabel fordi det er det eneste i funksjonen vi kan forandre. Alt annet er konstant.

### Eksempel 14


**Se på koden under imens du leser:**
*Vi vil nå skrive funksjonen **F** i Python. Da skriver vi først **def F(x):**. Denne gangen har vi en **x** inne i parentesene. **x** er argumentet vi selv kan styre når vi kaller på/bruker funksjonen.*

*Inne i funksjonen lager vi en variabel **antall_epler**, den setter vi lik **x + 2** (altså tallet vi bruker når vi kaller på funksjonen pluss 2).*

*Vi ønsker å returnere svaret som nå er lagret i **antall_epler** til der brukeren kaller på funksjonen. Vi skriver derfor **return antall_epler**.*

**Kode:**

```
def F(x):
    antall_epler = x + 2
    return antall_epler

a = F(3)        # Vi kaller på funksjonen F med x = 3 og lagrer resultatet i a
b = F(10)       # Vi kaller på funksjonen F med x = 10 og lagrer resultatet i b

# Vi skriver ut resultatene sammen med passende svarsetninger.

print("Dersom x er 3 får vi",a,"epler.")

print("Dersom x er 10 får vi",b,"epler.")
```

*Når vi skriver **F(3)** kaller vi på funksjonen med **x** verdi lik **3**. Da sendes tallet 3 inn i funksjonen og erstatter alle x-ene inne i funksjonen, **x + 2** blir derfor **3 + 2.** Svaret (altså **5**) lagres i **antall_epler** og verdien i **antall_epler** returneres. Vi lagrer dette resultatet videre i **a**. **a** bruker vi i vår første **print()**-kommando. Tilsvarende skjer når vi skriver **F(10)** svaret på dette lagres i **b**.*

**Utskrift i konsollen:**

```
Dersom x er 3 får vi 5 epler.
Dersom x er 10 får vi 12 epler.
```

**Oppgave 1.22 (funksjon_argument.py)**

**a)** Definer en funksjon kalt **x_pluss_3** som tar imot argumentet **x.** Her må du altså skrive **x_pluss_3(x).**
**b)** I funksjonen sett resultat lik **x+3**.
**c)** Returner resultatet.
**d)** Kall på funksjonen med **x** verdi **2** inne i en **print()**-funksjon (altså skriv: **print(x_pluss_3(2))** ). Kjør programmet.

**Løsningsforslag:**

http://pyløs0122.studenthjelp.no/

## En funksjon som tar imot fler argumenter

Vi skal nå se på funksjonen **G** som er avhengig av to variable. Disse variablene kaller vi her **x** og **y**, men vi kunne like gjerne kalt de noe annet som for eksempel **a** og **b**.

G(**x**,**y**) = **x*** 🍎 + 2***y*** 🍎 (altså **x** epler + (2***y**) epler)

Vi begynner med å regne ut hvor mange epler vi får dersom vi lar **x** være **3** og **y** være **10**. Deretter bytter vi om og ser hvor mange epler vi får dersom vi lar **x** være **10** og **y** være **3**.

G(**3,10**) = **3*** 🍎 + 2***10*** 🍎 = 3 🍎 + 20 🍎 = 23 🍎 # Vi erstatter **x** med **3** og **y** med **10** og regner ut.

G(**10,3**) = **10*** 🍎 + 2***3*** 🍎 = 10 🍎 + 6 🍎 = 16 🍎 # Vi erstatter **x** med **10** og **y** med **3** og regner ut.

Vi ser at vi erstattet **x** med det første tallet vi skrev inn (altså der vi skrev **x** da vi definerte funksjonen) og **y** med det andre tallet vi skrev inn.

### Eksempel 15

*Vi lager funksjon som tar inn to variabler/argumenter. Her prøver vi å kopiere funksjonen **G** i det andre eple-eksempelet vårt.*

**Kode:**

```
def G(x,y):
    resultat = x + 2*y
    return resultat

# Vi kan også kalle på funksjonen inne i print()-kommandoen.
print("Når x = 3 og y = 10 får vi", G(3,10), "epler")
print("Når x = 10 og y = 3 får vi", G(10,3), "epler")
```

**Utskrift i konsollen:**

```
Når x = 3 og y = 10 får vi 23 epler
Når x = 10 og y = 3 får vi 16 epler
```

**NB: Vi kan ha så mange argumenter vi vil i funksjonen.**

### Oppgave 1.23 (fler_argumenter.py)

**a)** Lag en funksjon kalt **x_ganger_y** la funksjonen ta imot 2 argumenter **x** og **y.** La funksjonen returnere **x*y**.
**b)** Kall på funksjonen med egenvalgte **x** og **y** verdier og skriv det ut til konsollen med **print()**.
**c)** Gjør en modifikasjon i funksjonen din. Legg nå til en tredje variabel **z**. Funksjonen skal nå returnere **x*y*z**.

**Løsningsforslag:**

http://pyløs0123.studenthjelp.no/

### Oppgave 1.24 (valgfri_funk.py)

Lag en funksjon hvor du selv bestemmer funksjonsnavn, antall argumenter som skal inn og ikke minst hva funksjonen skal returnere. Kjør funksjonen.

Lag gjerne flere funksjoner!

**Løsningsforslag:**

http://pyløs0124.studenthjelp.no/

# 1.7 Lister og listeoperasjoner

http://pythonhefte0107.studenthjelp.no/

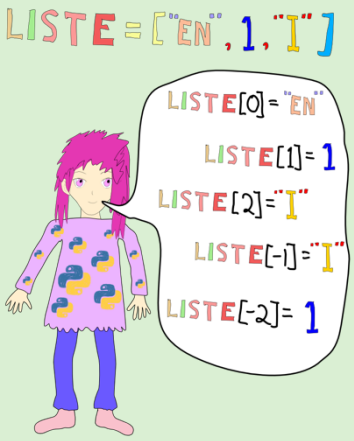
Vi har hittil sett raskt at det finnes en datatype som heter **"liste",** men nå er det på tide å se litt nærmere på hvordan vi kan bruke denne datatypen. For å repetere raskt så er en liste en datatype som samler data. I en liste kan det være flere forskjellige datatyper samtidig.
Eksempel: **liste_eksempel = [1, "eksempel", "liste"]**

## Å trekke elementer ut ifra en liste:

Om vi ønsker å trekke ut det første elementet i listen vi har kalt **liste_eksempel** så må vi skrive **liste_eksempel[0].** For å trekke ut det andre elementet må vi skrive **liste_eksempel[1]**. Vi trekker ut element nummer **n** i listen ved å skrive **liste_eksempel[n-1].** Skal vi trekke ut det tredje elementet i listen skriver vi **liste_eksempel[3-1]** altså **liste_eksempel[2]**.

Vi kan også trekke ut det siste elementet i listen ved å skrive **liste_eksempel[-1]**, det nest siste elementet ved å skrive **liste_eksempel[-2]** etc. Vi trekker ut element nummer n ifra slutte av listen ved å skrive **liste_eksempel[-n].**

### EKSEMPEL 16




**Kode:**

```
liste_a = []        # Vi lager en tom liste og kaller den liste_a
liste_b = [1,4,7]   # Vi lager en liste med tre heltall og kaller den liste_b
# Vi oppretter en liste med ett flyttall, en streng, et heltall og en boolean.
liste_c = [3.14, "Python", 42, True]    # Navn: liste_c

print(liste_c[0])       # Trekker ut det første elementet i liste_c
print(liste_c[1])       # Trekker ut det andre elementet i liste_c


print(liste_c[-1])      # Vi trekker ut det siste elementet i liste_c
print(liste_c[-2])      # Vi trekker ut det nest siste elementet i liste_c
```

**Utskrift i konsollen om vi kjører programmet nå:**

```
3.14
Python
True
42
```

### Oppgave 1.25 (liste.py)

**a)** Sett **liste_1 = [1,2,3,4,5]** og **liste_2 = [6,7,8,9]**
**b)** Hva tror du **liste_1[2]** og **liste_2[-2]** blir? Skriv **print(liste_1[2])** og **print(liste_2[-2])** inn i koden din og kjør koden for å bekrefte at det du tenkte var riktig.
**c)** Hva blir **liste_1[0] + liste_2[3]**? Skriv kode for å bekrefte svaret ditt.
**d)** Prøv kodebiten **print(len(liste_1)). len()**-funksjonen vil fortelle deg hvor lang en liste er. Dette blir nyttig fremover.
**e)** Bruk **len()** til å finne lengden til **liste_2**

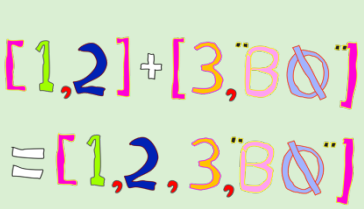
**Løsningsforslag:**

http://pyløs0125.studenthjelp.no/

## Å legge sammen lister med (+)-operatoren

Vi kan legge sammen to lister ved å bruke (**+**) – operatoren. **Liste_1 + Liste_2** vil gi listen som begynner med alle elementene i **Liste_1** og fortsetter med alle elementene i **Liste_2**

FORTSETTELSE EKSEMPEL 16



*Vi fortsetter på **eksempel 15**. Husk at **liste_a, liste_b** og **liste_c** allerede er definerte.*

**Kode:**

```
liste_b_pluss_c = liste_b + liste_c # Altså [1,4,7] + [3.14, "Python",42,True]

print(liste_b_pluss_c)  # Vi har nå listen [1,4,7,3.14,"Python", 42, True]
```

**Utskrift (i tillegg til det over) i konsollen om vi kjører programmet nå:**

```
[1, 4, 7, 3.14, 'Python', 42, True]
```

**Oppgave 1.26 (listeaddisjon.py)**

**a)** Sett **x = ["Lister", "er"]** og **y = ["gøy!"].**
**b)** Sett **z = x + y** bruk deretter **print()**-funksjonen til å skrive ut resultatet.
**c)** Finn **z[0]** og **z[-3]**
**d)** Skriv ut **3*y.**

**Løsningsforslag:**

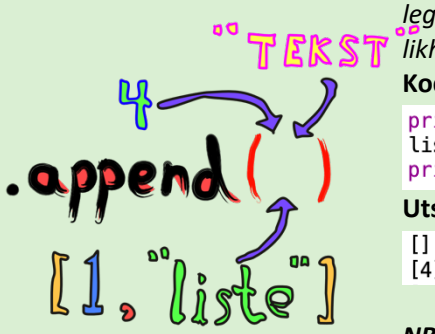
http://pyløs0126.studenthjelp.no/

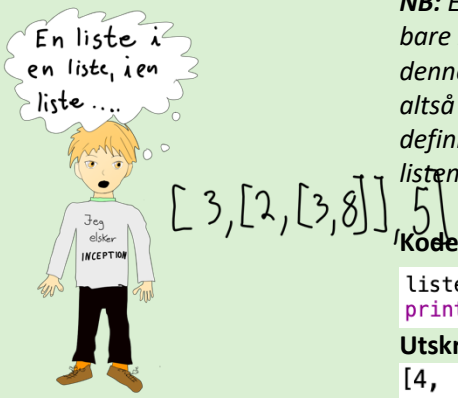
## .append() - metoden:

**.append()** er en liste-metode som brukes til å legge til enkeltelementer til en liste. Dette er første gang vi støter på "metoder" som er et eget tema innenfor programmering. Vi fordyper oss ikke i dette på nåværende tidspunkt, men fokuserer heller på hvordan vi kan bruke denne ene metoden.

Listenavn.append(a) vil legge til elementet **a** på slutten av listen med navn "Listenavn"

### FORTSETTELSE EKSEMPEL 16




*Vi fortsetter nok engang å skrive kode i samme program. Vi tar nå en nærmere titt på **liste_a** som foreløpig er tom. Bruker vi metoden **.append(4)** på denne listen så vil vi legge til heltallet **4** til listen. Legg spesielt merke til at vi nå ikke trenger å skrive likhetstegn i det hele tatt. Dette fordi vi tar i bruk en metode.*

**Kode:**

```
print(liste_a)
liste_a.append(4)
print(liste_a)
```

**Utskrift (i tillegg til det over) i konsollen om vi kjører programmet nå:**

```
[]
[4]
```

***NB:** En kan komme til å stille seg spørsmålet, hvorfor bruke **+** operatoren? Kan vi ikke bare bruke **.append()** for å legge sammen lister. Det vil gå an å legge til en liste på denne måten, problemet er at da legges listen inn som ett eget element i listen. Vi får altså en liste i en liste, også kalt en nestet liste (Også ett eget tema). Nestede lister er definitivt brukbare, men i denne delen av kurset vil vi kun fokusere på de enkleste listene.*

**Kode:**

```
liste_a.append(liste_b) # Vi legger til liste_b som ett eget element i liste_a
print(liste_a)      # Vi tar en titt på svaret vi nå får.
```

**Utskrift (i tillegg til det over) i konsollen om vi kjører programmet nå:**

```
[4, [1, 4, 7]]
```

### Oppgave 1.27 (listception.py)

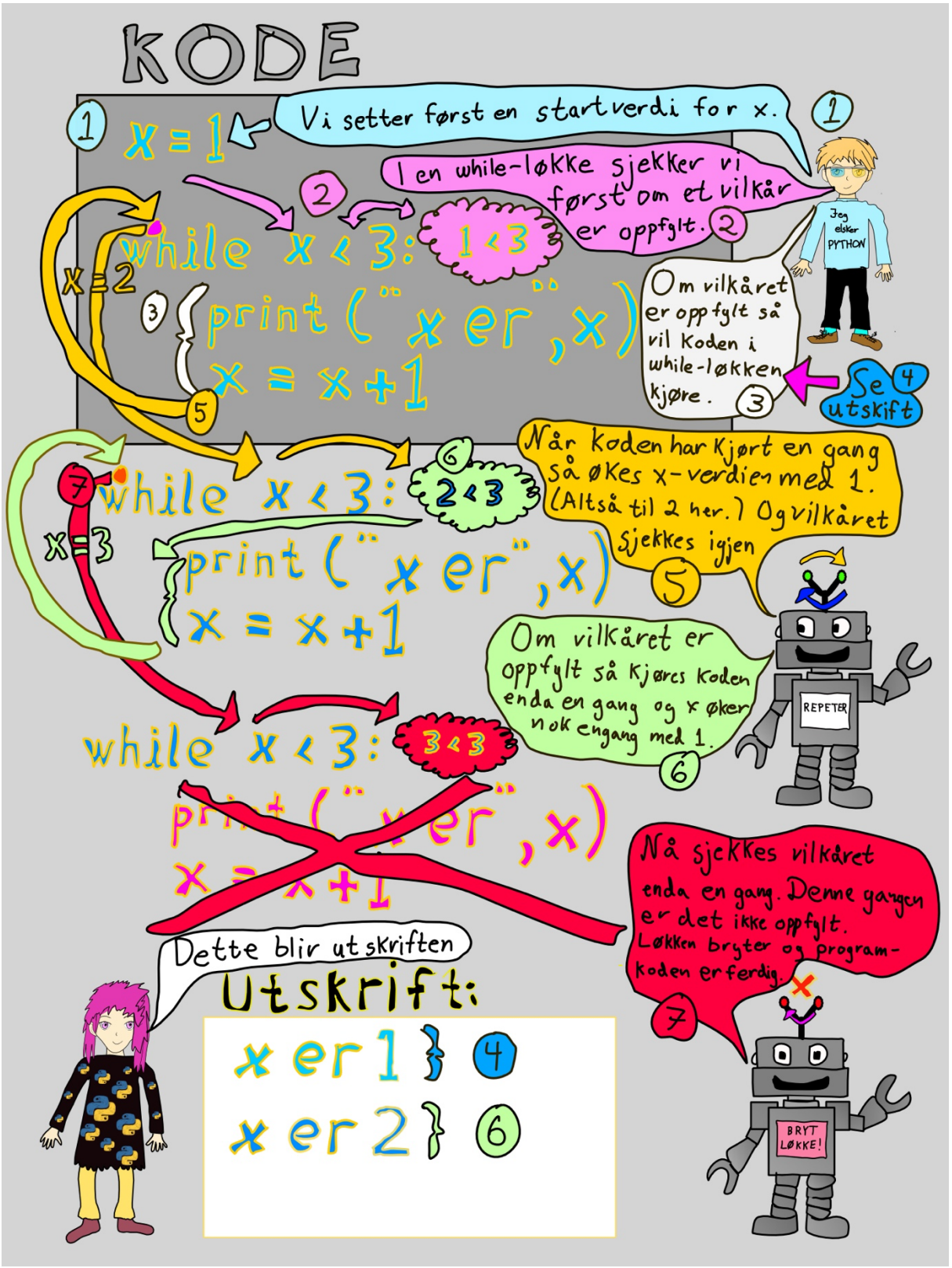
**a)** Sett **liste_1 = [2,"fire",3.14]**
**b)** Skriv **liste_1.append(5).** Skriv ut **liste_1**.
**c)** Sett **liste_2 = ["Enda","en","liste"]**
**d)** Skriv **liste_2.append(liste_1)**. Skriv ut **liste_2**
**e)** Se på det siste elementet i **liste_2** ved å skrive **print(liste_2[-1])**

**Løsningsforslag:**

http://pyløs0127.studenthjelp.no/

## 1.8 while-løkker

http://pythonhefte0108.studenthjelp.no/



## Bruk av while løkker

Du har tidligere lært om **if**, **else** og **sammenlikningsoperatorer**. Det du skal lære nå er veldig liknende. Forskjellen er at det som utføres går i en løkke. Det utføres derfor igjen og igjen til vilkåret vi har satt i utgangspunktet ikke lenger er oppfylt. Dette kan også føre til evige løkker (som aldri slutter) rett og slett fordi om du setter vilkåret feil kan det hende det alltid blir oppfylt!

### EKSEMPEL 17

*Vi ser på en **while**-løkke. Legg merke til at startverdien **x** defineres utenfor løkken. Dette er fordi om vi hadde definert denne inne i løkken ville **x** hele tiden gått tilbake til å bli lik **1** i starten av hver gjennomgang av løkka (og ville derfor aldri blitt større enn 4).*

**Kode:**

```
x = 1              # Vi setter en startverdi for x

while x <= 4:      # Betingelsen som må være oppfylt for at løkken skal kjøre
    print("Nå er x =",x)    # Løkkens formål, operasjonen som skal gjentas
    x = x + 1               # Vi øker x-verdien med 1
```

*Siden vi er i en løkke vil nå programmet gå tilbake opp til **while x <= 4:**. Dersom vilkåret fremdeles er oppfylt (altså dersom **x <= 4**) vil løkken gjennomgås enda en gang. Om ikke vilkåret er oppfylt lenger så vil løkken brytes og programmet begynner å lese neste linje kode under løkken (dersom det er kode der)*

**Kode:**

```
# Neste linje kode kjøres nå etter løkken brytes
print("While løkken er nå utført")
```

**Utskrift i konsollen:**

```
Nå er x = 1
Nå er x = 2
Nå er x = 3
Nå er x = 4
While løkken er nå utført
```

*Det er viktig at du tar deg tid når du leser utskriften slik at du forstår akkurat hvordan den oppstod.*

### Oppgave 1.28 (while_kopi.py)

**a)** Kopier koden i eksempelet over og kjør koden slik at du får samme utskrift som i eksempelet.
**b)** Se nøye på bildet i starten av delkapittelet. Bruk bildet til å forstå hvordan **while**-løkken fungerer.

**Løsningsforslag:**

http://pyløs0128.studenthjelp.no/

### Oppgave 1.29 (while.py)

Begynn med å definere **a = 0**.
Lag deretter en **while**-løkke som går så lenge **a < 7**. I løkken skriv ut **"Dette er gøy"** hver gang løkken kjører. La **a** øke med **2** for hver gjennomgang av løkken.

**Løsningsforslag:**

http://pyløs0129.studenthjelp.no/

## EKSEMPEL 18

Vi må være litt ekstra forsiktige når vi jobber med **while**-løkker. Løkken under vil fortsette å gå i all evighet. Grunnen er at vilkåret vi setter (**x < 5**) vil alltid være oppfylt så lenge løkken ikke endrer x-verdien underveis.

**Kode:**

```
x = 0

while x < 5: # Dette vilkåret vil i denne løkken alltid være oppfylt.
    print("Denne løkken går evig")
```

**Utskrift i konsollen:**

```
Denne løkken går evig
Denne løkken går evig
Denne løkken går evig
Denne løkken går evig
Denne løkken går evig
Denne løkken går evig
Denne løkken går evig
Denne løkken går evig
```

**VIKTIG:** For å avbryte en uendelig løkke bruker du **ctrl+c**

For å fikse dette så kan vi for eksempel øke verdien til x med 1 i hver gjennomgang av løkken. (Som i eksemplene ovenfor)

**Kode:**

```
x = 0

while x < 5: # Med denne nye løkken vil vilkåret bli oppfylt
    print("Denne løkken går ikke evig")
    x = x + 1
```

**Utskrift i konsollen:**

```
Denne løkken går ikke evig
Denne løkken går ikke evig
Denne løkken går ikke evig
Denne løkken går ikke evig
Denne løkken går ikke evig
```

### Oppgave 1.30 (while_evig.py)

**a)** Lag din egen uendelige løkke, når du kjører programmet husk at **ctrl+c** bryter kjøringen (Du må klikke deg inn på konsollen før du trykker)

**b)** Gjør en endring i programmet ditt slik at vilkåret blir oppfylt etter løkken har kjørt 3 ganger.

**Løsningsforslag:**

http://pyløs0130.studenthjelp.no/

## EKSEMPEL 19

Vi skal se på ett litt mer utfordrende eksempel. Legg først merke til at **input()** funksjonen er inne i en **int()** funksjon slik at strengverdien gjøres direkte om til ett heltall. Vi skal i dette eksempelet bruke en **while** løkke til å se hvor mange ganger vi trenger å doble et selvvalgt lite tall for å få ett tall større eller lik ett selvvalgt stort tall.

**Kode:**

```
x = int(input("Skriv inn ett lite tall: "))
y = int(input("Skriv inn ett stort tall: "))

x_start = x

n = 0

while x < y:
    x = 2 * x
    n = n + 1

print("")
print("Det tar", n, "doblinger før",x_start, "blir større enn eller lik",y)
```

**Utskrift i konsollen:**

```
Skriv inn ett lite tall: 3

Skriv inn ett stort tall: 9

Det tar 2 doblinger før 3 blir større enn eller lik 9
```

### Oppgave 1.31 (dobling_kopi.py) (Utfordring)

**a)** Skriv av eksempelet ovenfor og prøvekjør programmet for å sjekke at det virker som det skal.

**b)** Gjør en endring i koden slik at programmet skriver ut hvor mange ganger ett lite tall må triples før det er større enn eller lik det store tallet.

**c)** Gjør endringer i programmet slik at du nå ser hvor mange ganger et stort tall må halveres for å få det mindre enn et lite tall.

**Løsningsforslag:**

http://pyløs0131.studenthjelp.no/

### Oppgave 1.32 (while_epsilon.py) (Utfordring +)

**a)** Sett **x = 0**, **epsilon = 1** og definer en funksjon **f(x)** som returnerer funksjonen **-5*x+302.**

Vi ønsker først å finne ut hva **x** minst må være for at funksjonen skal bli mindre enn eller lik **0.**

**b)** Lag en **while**-løkke som kjører så lenge f(x) ≥ 0. Inne i **while** løkken kan du la verdien av **x** øke med **epsilon** for hver gang løkken kjøres.

**c)** Etter **while** løkken bryter la programmet skrive ut**: "Funksjonen har ett nullpunkt imellom", (x - epsilon), "og", (x)**

**Løsningsforslag:**

http://pyløs0132.studenthjelp.no/

**d)** Sett nå **epsilon = 0.5** og kjør programmet igjen. Er svaret nå annerledes? Prøv med **epsilon = 0.1**, **0.01** og **0.001.**

# *1.9 for-løkker og range()-funksjonen*

http://pythonhefte0109.studenthjelp.no/

I en **for-løkke** bruker vi en liste (eller et listeliknende objekt som du skal se om litt) for å se hvordan og hvor mange ganger vi skal kjøre igjennom løkken. I dette delkapittelet må du huske å lese eksempelforklaringen nøye. Eksemplene vil være essensiell for å forstå **for**-løkker.

## EKSEMPEL 20

*Vi tar for oss en enkel **for-løkke** som går igjennom en liste med **3** elementer.*

**Kode:**

```
for element in [2,"Sokk",7]:
    print(element)
```

***for-løkken** begynner med å tilordne første verdien i listen altså **2** til variabelen **element**. Koden som utføres i løkken altså **print(element)** sørger da for at **2** skrives ut til konsollen.*

*Siden det ikke er fler linjer kode i løkka så er løkken for første element utført.*
*Vi går videre gjennom løkken til andre verdien i løkken (altså streng elementet **"Sokk"**). Koden kjøres igjen og **Sokk** blir skrevet ut til konsollen.*
*Etter koden har kjørt nok engang vil **7** også være skrevet ut til konsollen.*

**Utskrift i konsollen:**

```
2
Sokk
7
```

## EKSEMPEL 21

*Vi ser på nok en **for**-løkke. Listen er denne gangen lagret i en variabel (som vi har kalt **liste_1**). Vi har definert en variabel **x = 0** utenfor løkken. **x** skal vi bruke til å finne summen av tallene i listen.*

**Kode:**

```
liste_1 = [3,4,5,6]      # Vi lager listen vi skal summere.
x = 0                    # Her skal vi lagre summen vår imens vi kjører løkken
print("x er først lik 0")
for tall in liste_1:     # Her kan du tenke deg at det står [3,4,5,6]
    x = x + tall         # Vi summerer x med det liste-elementet.
    print("Når vi legger til", tall, "får vi x =",x) # Vi skriver ut resultat.
```

***NB:** Der vi skrev **element** i forrige oppgave har vi nå skrevet **tall**, men vi kunne like gjerne skrevet **element**.*

**Utskrift i konsollen:**

```
x er først lik 0
Når vi legger til 3 får vi x = 3
Når vi legger til 4 får vi x = 7
Når vi legger til 5 får vi x = 12
Når vi legger til 6 får vi x = 18
```

### Oppgave 1.33 (for_kopi.py)

**a)** Kopier koden i det øverste eksempelet på forrige side og kjør koden. Pass på at utskriften blir den samme.
**b)** Legg til et ekstra element i listen som skrives ut. Kjør koden.
**c)** Gjør en endring på ordet **element** i koden slik at det istedenfor **element** står **i**. Pass på å bytte ut begge steder det står **element** før du nok engang kjører koden.

**Løsningsforslag:**

http://pyløs0133.studenthjelp.no/

### Oppgave 1.34 (priser.py)

**a)** Definer **x = 1** og **samlet_pris = 0** listen **priser = [34,80,20,19,128]**
**b)** Lag en for løkke **for pris in priser:** som kjører koden
1) **print("Varenummer:", x, "koster", pris, "kroner.")**
2) **samlet_pris = samlet_pris + pris**
3) **x = x+1**
**c)** Når koden har blitt kjørt. Skriv ut **samlet_pris**, sammen med en passende svarsetning.

**Løsningsforslag:**

http://pyløs0134.studenthjelp.no/

### Oppgave 1.35 (for_funk.py)

a) Definer funksjonen **f(x)** slik at den returnerer $x^2$, sett **x = [0,1,2,3,4]** og lag en tom liste **y = []** (utenfor funksjonen).
b) Lag en **for**-løkke (**for i in x**) som for kjører koden **y.append(f(i))**
c) Når koden er utført skriv ut resultatet.

**Løsningsforslag:**

http://pyløs0135.studenthjelp.no/

## range()-funksjonen

Når vi arbeider med for-løkker ønsker vi noen ganger en enkel måte å kjøre løkken et gitt antall ganger. Her kan **range()** funksjonen hjelpe oss. **range()** oppretter et listeliknende element som vi kan bruke når vi lager for-løkker! Om vi for eksempel skriver **for i in range(4):** så vil vi ha opprettet en for-løkke som kjører 4 ganger.

Men hva slags elementer gir egentlig range funksjonen oss? Svarer vi på det kan vi bruke hvert enkeltelement i range() funksjonen i for-løkkene våre.

| De forskjellige måtene å bruke range() | Eksempler |
|---|---|
| **range(n)** → Oppretter et objekt som likner på listen **[0, 1 , 2,...,n - 1]** Legg spesielt merke til at listen har n-elementer, men ettersom det første elementet er 0 vil det siste være 1 lavere enn **n**. | **range(4)** gir et objekt som likner på listen (altså listeobjektet) **[0,1,2,3]** |
| **range(start,stopp)** → Oppretter et objekt som ligner på listen som øker med 1 med første element **start** og siste element **(stopp – 1)**. Altså **[start, start + 1, start + 2,... , stopp - 1]**. **Merk igjen at siste element er (stopp – 1)** og altså ikke **stopp.** | **range(3,8)** gir et objekt som likner på listen **[3,4,5,6,7]** |
| **range(start,stopp,skritt)** → Oppretter et objekt som likner på listen med første element **start,** andre element **start + skritt,** tredje element **start + 2* skritt** osv. Det siste elementet blir det høyeste tallet vi kommer til ved å øke med skritt uten at tallet forbigår eller er lik **stopp.**<br><br>Altså:<br>**[start, start + skritt, start + 2* skritt, .... , start + k*skritt]** der **k** er det høyeste heltallet som er slik at **(start + k*skritt) < stopp.**<br><br>Dette kan virke ganske så vanskelig, men i praksis gir det intuitivt mening. | **range(0,28,5)** gir et objekt som likner på listen **[0,5,10,15,20,25]**<br><br>**range(3,17,2)** gir et objekt som likner på listen **[3,5,7,9,11,13,15]** |

### EKSEMPEL 22

*Vi sier at **range()**-funksjonen gir oss et listeliknende objekt ettersom om vi definerer **x = range(n)** (der **n** er et heltall) så kan vi på samme måte som med en vanlig liste finne det første elementet i dette objektet ved å skrive **x[0]**. Vi kan derimot ikke skrive hele listen normalt ut i konsollen.*

**Kode:**

```
x = range(4)        # range(4) tilsvarer listen [0,1,2,3]
print(x[0])         # Vi printer ut det første elementet i x.
print(x)            # Vi printer ut x, men ser at vi ikke får den vante listen.
```



**Utskrift i konsollen:**
*Her ser vi at det første elementet i **range(4)** er **0**. Men om vi prøver å skrive ut **x** så får vi **range(0,4)** og ikke listen **[0,1,2,3]**.*

```
0
range(0, 4)
```

### Oppgave 1.36 (range.py)

**a)** Sett **r_1 = range(5).** Hva blir de 3 første elementene i **r_1**? Det siste?
Test antagelsen din ved å skrive **print(r_1[0], r_1[1], r_1[2], r_1[-1])**

**b)** Sett **r_2 = range(3,10).** Hva blir de 3 første elementene i **r_2**? Det siste?
Test antagelsen din ved å skrive **print(r_2[0], r_2[1], r_2[2], r_2[-1])**

**c)** Sett **r_3 = range(4,20,3).** Hva tror du vil være de 3 første elementene i **r_3**? Hva med det siste?
Test antagelsen din ved å skrive **print(r_3[0], r_3[1], r_3[2], r_3[-1])**

**Løsningsforslag:**

http://pyløs0136.studenthjelp.no/

## range()-funksjonen brukt i for-løkker

Nå skal vi se på noen applikasjoner av **range()**-funksjonen i **for**-løkker. NB: Det er viktig å vite at **range()**-funksjonen kun tar imot heltall som argumenter, om du setter inn noe annet så vil du få feilmeldinger.

### EKSEMPEL 23

*Vi tar for oss en **for**-løkke vi ønsker at skal gåes igjennom **3** ganger (en annen måte å si dette er at vi ønsker å gjøre **3** iterasjoner av løkken).*

**Kode:**

```
x = 0

for i in range(3):
    x = x+1
    print("Dette er den", x,"iterasjonen av denne løkka")
    print("Akkurat nå er i =",i)
```

**Utskrift i konsollen:**

```
Dette er den 1 iterasjonen av denne løkka
Akkurat nå er i = 0
Dette er den 2 iterasjonen av denne løkka
Akkurat nå er i = 1
Dette er den 3 iterasjonen av denne løkka
Akkurat nå er i = 2
```

### Oppgave 1.37 (varer_priser.py)

I denne koden har vi med en liste med varer og i en annen liste så har vi prisene til disse varene lagret (i riktig rekkefølge selvsagt). Husk at len()-funksjonen returnerer lengden til en liste.

**a)** Lag listene **vare = ["Eple", "Brød", "Skinke"]** og **pris = [10,20,25]**

**Løsningsforslag:**

**b)** Skriv **for i in range(len(vare)):** for å kjøre koden **print(vare[i], " koster ",pris[i], " kroner")**
Kjør koden.
**NB:** Her kunne vi også bruk **3** istedenfor **len(varer),** men da må vi i så fall endre koden vår om vi ønsker å forlenge listene.

**c)** Vi får vite at en **Paprika** koster **12** kroner på denne butikken. Legg til informasjonen på relevante steder i listene og kjør koden igjen

http://pyløs0137.studenthjelp.no/

## EKSEMPEL 24

*La oss ta et eksempel der vi bruker **range(start,stopp,skritt)**. Om vi skal lage et program som skriver ut oddetallene fra og med 1 og opp til 10. Måten vi finner neste oddetall etter 1 er ved å legge til 2, så vi setter skritt lik 2 i **range()**-funksjonen.*

**Kode:**

```
for i in range(1,10,2):          # Tilsvarer listen [1,3,5,7,9]
    print(i)                     # Vi skriver til konsollen element for element
```

**Utskrift i konsollen:**

```
1
3
5
7
9
```

**Oppgave 1.38 (for_range_funk.py)**

a) Definer funksjonen $f(x)$ og sett den lik $-x^2 + 2x$. Sett **x = []** og **y = []**
b) Bruk **range(start, stopp, skritt)** til å lage en **for**-løkke som lagrer alle tallene **10,20,30,...,100** i **x**.
c) Modifiser for-løkken slik at den samtidig lagrer **f(10), f(20)...f(100)** i **y**
d) Ved hjelp av en ny for-løkken la programmet skrive ut **(10,f(10)), (20,f(20))...(100,f(100))**

**Løsningsforslag:**

http://pyløs0138.studenthjelp.no/

# 1.10 and, or og elif

## Logiske operatorer (and og or)

http://pythonhefte0110.studenthjelp.no/

Vi skal nå utvide litt på det vi det vi lærte om i **1.5 Sammenlikningsoperatorer og if/else – setninger.** Noen ganger ønsker vi å sjekke om fler vilkår er oppfylt samtidig eller om minst ett av vilkårene vi oppgir er oppfylt. Da kan vi bruke de logiske operatorene **and** og **or.**

| |
|---|
| **and** → Vi sjekker om 2 vilkår er oppfylt samtidig (Eks: **x > 8 and x < 15**) |
| **or** → Vi sjekker om minst ett av vilkårene er oppfylt (Eks: **x < 2 or x > 6**) |

**Oppgave 1.39 (konsolloppgave)**

I denne oppgaven skal du bruke Python-konsollen for å gjennomføre koden.

a) Sett **x = 8**, trykk **Enter**. Som svar på koden **x < 6 and x < 10** tror du vi vil få **True** eller **False**? Skriv inn koden og se om du fikk riktig svar.

b) Gjennomfør koden over igjen, men denne gang gjør en endring ifra **and** til **or**. Fikk du nå resultatet du forventet?

**Løsningsforslag:**

http://pyløs0139.studenthjelp.no/

EKSEMPEL 25

***a)*** *Vi prøver først å se hva som skjer om vi bruker **or**-operatoren (Altså eller)*

**Kode:**

```
x = 2

if x >= 5 or x < 3:
    print("x er større enn (eller lik) 5 eller mindre enn 3")
else:
    print("x er slik at ingen av vilkårene er oppfylt")
```

**Utskrift i konsollen:**

```
x er større enn (eller lik) 5 eller mindre enn 3
```

***b)*** *Nå ser vi på hva som vil skje dersom vi bruker **and**-operatoren (altså og)*

**Kode:**

```
x = 2

if x >= 5 and x < 3:
    print("x er større enn (eller lik) 5 og mindre enn 3") # Umulig vilkår
else:
    print("Kun ett eller ingen av vilkårene er oppfylt")
```

**Utskrift i konsollen:**

```
Kun ett eller ingen av vilkårene er oppfylt
```

## EKSEMPEL 26

*Det er ofte vi har vilkår som er avhengig av forskjellige variabler. Her er et eksempel på en **and**-test som gir **True**.*

**Kode:**

```
x = 5
y = -3

if x > 2 and y < -1:
    print(x,"er større enn 2 og", y, "er mindre enn -1")
else:
    print("Kun ett eller ingen av vilkårene er oppfylt")
```

**Utskrift i konsollen:**

```
5 er større enn 2 og -3 er mindre enn -1
```

### Oppgave 1.40 (x_and_or_y.py)

**a)** Gi **x** verdien **3** og **y** verdien **5**.

**b)** Lag en **if**- test som skriver ut **"Begge tallene er større enn 4"** om både **x** og (**and**) **y** er større enn **4**.

**c)** La programmet skrive ut **"Minst ett av tallene er mindre eller lik 4"** ellers. Kjøre programmet med forskjellige **x-** og **y**-verdier.

**d)** Gjør nå en endring på programmet slik at det sjekker om enten **x** eller (**or**) **y** er større enn **4**. Skriv passende utskrifter til hvert tilfelle.

**Løsningsforslag:**

http://pyløs0140.studenthjelp.no/

### Oppgave 1.41 (kino.py)

Elise og storesøsteren Marie skal på kino. Filmen de skal på har aldersgrense 15 år.

**a)** Sett **Elise = 14** og **Marie = 18**. Skriv en if setning som skriver ut **"Begge får sett filmen."** Dersom både Elise og Marie er **15** år eller eldre.

**b)** Lag en **else**-setning som skriver ut **"De kom seg dessverre ikke inn."** dersom begge ikke er 15 år eller eldre.

**c)** Prøv koden for forskjellige aldre for Elise og Marie.

**Bonusoppgave:** Kinoen har en regel som gjør at i følge med en voksen så kan også de som er eldre enn 12 år se filmen.
Lag en ny **if-setning** i **else-setningen** der vilkåret er at for å komme inn på kinoen må Elise være minst 12 år og Marie minst 18 år. Dersom **denne if setningen** slår ut la programmet **skrive "De kom kun inn fordi Marie er voksen"**.

**Løsningsforslag:**

http://pyløs0141.studenthjelp.no/

## elif (else if) setninger:

Nå skal vi utvide **if/else** konseptet. Vi innfører **elif**.

### EKSEMPEL 27

*Hva hvis du ønsker å utføre en handling når **x > 10** en annen handling når **8 < x < 10** og tredje handling ellers? Da kan **elif** (else if) hjelpe deg.*

**Kode:**

```
x = 9

if x > 10:      # Dersom if koden blir kjørt blir ingen andre kjørt.
    print("x er større enn 10!")
elif x > 8 and x < 10:      # Her kunne vi også ha skrevet bare elif x > 8:
    # Det er fordi elif kommandoen under if bare utløses dersom if ikke kjøres
    print("x er imellom 8 og 10")
else:
    print("x er mindre enn 8")
```

**Utskrift i konsollen:**

```
x er imellom 8 og 10
```

### Oppgave 1.42 (elif_kopi.py)

Kopier koden ifra eksempelet like over. Kjør deretter koden for forskjellige verdier av **x** for å passe på at koden virker slik den skal.

**Løsningsforslag:**

http://pyløs0142.studenthjelp.no/

### Oppgave 1.43 (karakter_elif.py)

En ungdomsskole holder på med ett prøveprosjekt hvor læreren istedenfor karakterer skal gi en tilbakemelding om måloppnåelsen til eleven er høy middels eller lav. Enkelte lærere protesterer og sier det vil være veldig vanskelig å gi denne typen tilbakemelding.
Siden å krangle med en lærer noen ganger er umulig bestemmer skolestyret seg for å be deg lage en enkel programvare for å hjelpe lærere som liker å gjøre ting på gamlemåten å tilpasse seg det nye systemet.

**a)** Sett **x** lik et heltall i området fra og med **1** til og med **6**. Hvis **x ≥ 5** la programmet skrive: **"Måloppnåelsen er høy".**

**b)** Lag nå en **elif**-test under if-testen. Dersom **3 ≤ x < 5** så skal programmet skrive: **"Måloppnåelsen er middels"**.

**c)** Ellers la programmet skrive **"Måloppnåelsen er lav".** Prøv med forskjellige **x** verdier for å sikre at programmet fungerer.

Skolen har enda ikke fått de pengene den trenger til programmeringsopplæring. Lærerne sliter derfor med å endre x-verdien i programmet og kjøre det selv.

**Løsningsforslag:**

http://pyløs0143.studenthjelp.no/

Skoleledelsen vil derfor at du skal gjøre programmet ditt enda enklere.
Programmet skal nå kun kreve en input av en karakter ifra 1-6 og ingenting annet.
Programmet skal fortsette å motta input og gi tilbakemeldinger helt til noe annet enn ett tall ifra 1-6 skrives inn.

**d)** Gjør en endring i programmet slik at **x = input("Skriv inn en karakter: ")**
Husk å gjør om **x** til ett heltall (integer) med **int()**-funksjon.

**e)** Sett opp en **while**-løkke som kjører så lenge **x ≥ 1** og **x ≤ 6**. I **while**-løkken skal du først gi **x** en ny verdi (som i oppgave **d)**). Deretter skal testen ifra oppgave **a), b)** og **c)** kjøres for å gi en indikasjon på måloppnåelse.



# *1.11 Enkel bibliotek import og random-modulen*

http://pythonhefte0111.studenthjelp.no/

## Enkel stjerneimport

Vi skal nå lære om hvordan vi importerer biblioteker i Python. Den enkleste importen for introduksjonsformål er en såkalt stjerneimport. Det er verdt å nevne at det å benytte en slik import ofte ikke er sett på som "god skikk". Dette er blant annet fordi denne typen import ofte importerer mer enn nødvendig og at alternativet ofte gjøre koden vår mer oversiktlig. Vi begynner med å importere **random-modulen direkte til konsollen**. For funksjonene under skal virke må du først skrive inn: **from random import *** i **konsollen** og trykke **Enter**.

| RANDOM-FUNKSJONER | EKSEMPLER IFRA KONSOLLEN |
|---|---|
| **randint(a,b)** → Skriver ut ett tilfeldig heltall fra og med **a** til og med **b**<br><br>**NB: Utfallet vil variere ifra gang til gang.** | `In [64]: randint(1,6)`<br>`Out[64]: 2`<br><br>`In [65]: randint(1,6)`<br>`Out[65]: 5` |
| **uniform(a,b)** → Skriver ut ett tilfeldig tall i området fra og med **a** til og med **b**<br><br>**NB: Utfallet vil variere ifra gang til gang.** | `In [68]: uniform(1,6)`<br>`Out[68]: 2.671179626879729`<br><br>`In [69]: uniform(1,6)`<br>`Out[69]: 4.203240834034414` |
| **choice([listeelement_1, listeelement_2, listeelement_3…, listeelement_n])** → Skriver ut et tilfeldig element ifra en liste.<br><br>**NB: Utfallet vil variere ifra gang til gang.** | `In [71]: choice([1,"To",3,4.0,"fem"])`<br>`Out[71]: 1`<br><br>`In [72]: choice([1,"To",3,4.0,"fem"])`<br>`Out[72]: 'To'` |

**Oppgave 1.44 (konsolloppgave)**

**a)** Skriv inn i konsollen **from random import ***, trykk **Enter.** Du har nå lastet inn random modulen til konsollen.

**b)** Simuler ett terningkast ved å skrive **randint(1,6)**

**c)** Bruk nå **randint()** til å simulere ett myntkast der **0** betyr kron og **1** betyr mynt.

**d)** Bruk kommandoen **round(uniform(0,1),2)** for å simulere ett tilfeldig desimaltall imellom **0** og **1** med **2** gjeldende desimaler.

**e)** Bruk kommandoen **choice(["Susanne","Tormod","Ricardo"])** til å simulere et tilfeldig valg av ett navn.

**f)** Bruk **choice()** til å simulere ett myntkast og ett terningkast.

**Løsningsforslag:**

http://pyløs0144.studenthjelp.no/

## EKSEMPEL 28

*Vi skal simulere at Sara og Lars kaster 2 terninger hver. Vi lar programmet teste vinneren ved å se hvem som har høyest samlet sum.*

```
from random import *

Sara_1 = randint(1,6)       # Vi simulerer terningskast og lagrer det.
Sara_2 = randint(1,6)
Lars_1 = randint(1,6)
Lars_2 = randint(1,6)

Sum_Sara = Sara_1 + Sara_2    # Vi finner summen av terningkastene.
Sum_Lars = Lars_1 + Lars_2

if Sum_Lars > Sum_Sara:
    print("Lars Vant! Han fikk til sammen:", Sum_Lars, "Sara fikk:", Sum_Sara)
elif Sum_Lars == Sum_Sara:
    print("De fikk",Sum_Lars,"begge to! De fikk like mye.")
else:
    print("Sara Vant! Hun fikk til sammen:", Sum_Sara, "Lars fikk:", Sum_Lars)
```

**Kode:**
**Utskrift i konsollen: (Her er det fler muligheter, jeg legger ved 2 av dem.)**

```
Sara Vant! Hun fikk til sammen: 12 Lars fikk: 3
```

```
Lars Vant! Han fikk til sammen: 8 Sara fikk: 5
```

### Oppgave 1.45 (random_kopi.py)

a) Kopier koden ifra eksempelet like over.

Sara og Lars bestemmer seg for å isteden ha en konkurranse om hvem som får best sammenlagt resultat på 2 fiksjonelle terninger med 100 sider (med tallene 1-100)

b) Gjør nødvendige endringer i endring i programmet slik at vi isteden simulerer situasjonen med de nye terningene.

Sara og Lars bestemmer seg for at de ikke ønsker å kun være bundet til heltall i "kastene" sine. De ønsker derfor at hvert "kast" nå gir et tall imellom 0 og 100 med 2 desimaler.

c) Gjør en endring i koden for å reflektere dette. Her bør du få bruk for **uniform()** og **round()**.

**Løsningsforslag:**

http://pyløs0145.studenthjelp.no/

## Import av enkeltfunksjoner fra biblioteker

Vi har frem til nå lært hvordan vi importerer alle funksjonene i et pythonbibliotek. Vi skal nå se på hvordan vi kan importere funksjoner enkeltvis ifra biblioteker. For å gjøre dette ersatter vi * i importkommandoen vår med funksjonsnavn. Ønsker vi å importere fler funksjoner så setter vi komma imellom slik vist i eksempelet under.

### EKSEMPEL 29

*Sara og Lars kjeder seg... VELDIG! Og bestemmer seg for å kaste **1000** terninger hver og ser hvem som får høyest sum. For å gjøre det rettferdig mener Sara at de må kaste med lik terning. De har 4 terninger med 4 forskjellige farger.*
*I koden under simulerer vi et valg av terning og at de hver kaster terningen **1000** ganger og regner ut summen av kastene sine.*

**Kode:**

```
from random import randint, choice

Sum_Lars = 0
Sum_Sara = 0
terningkast = 1000

farge = ["grønn", "blå", "gul", "rosa"]

terningfarge = choice(farge)

for i in range(terningkast):
    Sum_Lars += randint(1,6)
    Sum_Sara += randint(1,6)

print("Sara fikk til sammen:", Sum_Sara, "og Lars fikk:", Sum_Lars)
print("De kastet med en", terningfarge, "terning")
```

**Utskrift i konsollen:** *(En av mange potensielle utskrifter)*

```
Sara fikk til sammen: 3470 og Lars fikk: 3443
De kastet med en gul terning
```

*En liten ting å legge merke til her er at utfallet Sara fikk og utfallet Lars fikk er ganske nært (Med tanke på at laveste mulige utfall er **1000** og høyeste er **6000**) Vi vil faktisk forvente at utfallene blir ganske nærme her.*

**Oppgave 1.46 (random_kast.py)**
**NB: Husk å importere det du trenger ifra random modulen enkeltvis.**

**a)** Lag et program som simulerer et terningkast. Lagre verdien i en variabel du kaller **terning_1**. La programmet skrive ut hva du fikk på terningkastet.

**b)** Simuler et nytt terningkast og lagre verdien av det kastet i **terning_2.** Lag en **if/else**-setning som skriver ut resultatet av den høyeste terningen. (Husk at om terningene er like så er begge høyest og det spiller altså ingen rolle hvilken du skriver ut)

**Løsningsforslag:**

http://pyløs0146.studenthjelp.no/

## Import av enkeltfunksjoner med egenvalgte navn (alias)

Vi har muligheten til å endre navnene på funksjonene vi importerer imens vi importerer de. Dette kalles å importere funksjoner under et alias. Som tidligere spesifiserer vi funksjonene vi ønsker å importere, men vi skriver **as nyttnavn** ved siden av. Vi kan gjøre dette for å korte ned det vi trenger å skrive når vi lager koden vår, eller vi kan gjøre det av nødvendighet fordi vi allerede har importert andre funksjoner med samme navn.

### EKSEMPEL 30

**Kode:**

```
from random import randint as ri, choice as ch

print(ri(1,6)) # Her benytter vi oss av funksjonen randint med alias ri.

print(ch(["En", 2.0, 3, "fire"])) # Her bruker vi choice med alias ch.
```

**Utskrift i konsollen:** *(En av mange potensielle utskrifter)*

```
2
fire
```

**Oppgave 1.47 (random_alias.py)**

a) Skriv **from random import randint** i begynnelsen av koden.
b) Bruk **randint** til å simulere et myntkast der 0 representer mynt og 1 representerer kron. La programmet skrive ute kron eller mynt. (NB: Her kunne vi også brukt choice-funksjonen)
c) Gjør en endring i koden slik at du også importerer **uniform** (ifra **random**) som **uni.**
d) Bruk **uni** til å finne et et tilfeldig tall imellom 0 og 100. bruk **round()** til å runde av svaret til 2 desimaler.

**Løsningsforslag:**

http://pyløs0147.studenthjelp.no/

## Import og bruk av "klasser" som funksjoner

Vi introduserer enda en vanlig import av funksjoner. I dette tilfellet importerer vi hele biblioteket som et eget bibliotekobjekt altså en "klasse". Vi kan benytte oss av dette objektet direkte når vi skal kalle på bibliotekets funksjoner. Teknisk sett er disse funksjonene ikke lengre funksjoner som vi er vant med ifra tidligere, men klasse/objekt metoder.

### EKSEMPEL 31

**Kode:**

```
import random

print(random.randint(1, 8)) # Vi benytter klassen "random"'s metode .randint()
# for å kalle på et tilfeldig heltall i området [1,8]

print(random.uniform(0,10)) # Vi benytter klassen "random"'s metode .uniform()
# for å kalle på et tilfeldig tall i området (0,10)
```

**Utskrift i konsollen:** *(En av mange potensielle utskrifter)*

```
2
5.491118173603354
```

## Endring av navn på importerte klasser

Vi kan også importere **bibliotek (/klasser)** med et andre navn (random kan for eksempel erstattes med **rnd)**. Dette gjøres noen ganger for å korte ned koden. Vi skriver da **import biblioteknavn as nyttnavn.**

EKSEMPEL 32

*Her har vi erstattet **random** med **rnd***
**Kode:**

```
import random as rnd

print(rnd.randint(1, 8))

print(rnd.uniform(0,10))
```

**Utskrift i konsollen:** *(En av mange potensielle utskrifter)*

```
1
0.10001405394228668
```

**Oppgave 1.48 (random_import.py)**

a) Skriv **import random** i toppen av koden din.
b) Bruk **random.randint** til å generere et tilfeldig heltall i området **[3,10]**
c) Bruk **random.uniform(a,b)** til å finne et tilfeldig tall imellom 2 og 4.
d) Gjør en endring i programmet slik at du importerer **random** som **rnd**. Husk at du nå også må endre de andre stedene i koden det står **random** til **rnd.**

**Løsningsforslag:**

http://pyløs0148.studenthjelp.no/

# *Oppgavesamling Kapittel 1*

Husk å lagre programmene du lager i en egen mappe. Dette er spesielt viktig for oppgavene med filnavn med dette utseendet: **oppgavenavn_1.** Disse oppgavene bygger vi nemlig videre på i senere delkapitler.

## Oppgaver: 1.1 Kommentarer og print() funksjonen

### Oppgave 1.1.1 (kommentar_og_print.py)

**c)** Bruk **print()**-funksjonen til å lage et program som skriver ut **"Å lære programmering går overraskende fort!"** til konsollen når du kjører programmet.

**d)** I samme program lag en kommentar der du bruker #.
Kommenter: Copy/Paste-metode når vi lærer programmering er ofte en dårlig idé.

**e)** I samme program lag enda en kommentar, denne gangen bruk trippel anførselstegn (""") foran og bak det du skriver. Kommenter:

**NB:** I programmering gjelder det å være veldig nøyaktig når du skriver kode. Dersom vi skriver en bitteliten feil vil vi komme til å få feilmeldinger, dette er derfor veldig vanlig (også for profesjonelle programmerere). Jo mer kode vi skriver, jo flinkere blir vi til å skrive nøyaktig.

### Oppgave 1.1.2 (navn_og_alder_1.py)

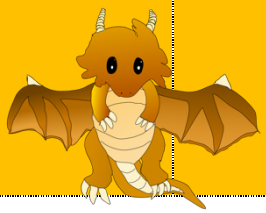
I denne oppgaven skal vi øve på å skrive **komma(,)** imellom argumentene til **print()**-funksjonen. Lag et program der du **skriver print("Jeg heter", "(Ditt navn)", "og er", (Din alder), "år.")**

### Oppgave 1.1.3 (tenke_selv.py)

Du skal begynne å tenke selv fremfor at denne boken skal gi deg litt for spesifikke instruksjoner på hvordan du skal gå frem. For å fremme din selvstendighet som programmerer så vil det være mange oppgaver uten for spesifikke instruksjoner på hvordan du skal gå frem. Gjør **fler** gode forsøk før du tyr til løsningsforslaget.

Lag et program med følgende utskrift (inkludert tomme linjer):
**Det ligger stor verdi i å kunne gjøre ting selvstendig.**

**Kompetente mennesker kan bedre hjelpe seg selv og de rundt seg.**
**Det sagt...** Det er også viktig å kunne spørre andre om hjelp.



# Oppgaver: 1.2 Variabeltyper (Datatyper) og tilordning av verdier

### Oppgave 1.2.1 (variabeltyper.py)

Lag et program der du:

- **a)** Lagrer heltallet **5** i variabelen med navn **int_a.**
- **b)** Lagrer flyttallet **2.73** i variabelen **float_b.**
- **c)** Lagrer tekststrengen **"Programmering er spennende"** i variabelen **string_c**
- **d)** Lagrer listen: **["Du","er","nummer",1]** i variabelen **list_d**.
- **e)** Bruk **print()-funksjonen** til å skrive ut alle variablene. Kjør programmet.
- **f)** Beskriv det du gjør i koden din med relevante kommentarer.

### Oppgave 1.2.2 (navn_og_alder_2.py)

Vi skal nå modifisere programmet vi lagde i ***Oppgave 1.1.2***.

- **a)** På begynnelsen av koden definer **navn** til å være en variabel som holder ditt navn (Et strengobjekt)
- **b)** La **alder** være en variabel som inneholder din alder (et heltall).
- **c)** I **print()**-funksjonen din erstatter du nå navnet ditt med variabelen **navn** og alderen din med variabelen **alder**. Kjør programmet.

Husk å beskrive hva du gjør i koden din med relevante kommentarer.

### Oppgave 1.2.3 (elementer.py)

Lag et program det du definerer variabelen **e_1** til å være et *heltall* (Bestem heltallet selv), **e_2** til å være et *flyttall* og **e_3** til å være et *strengobjekt*. Definer en variabel **elementer** til å være en liste med **e_1** som første element, **e_2** som andre element og **e_3** som tredje element i lista (dvs: **elementer = [e_1, e_2, e_3]**) Bruk så **print()** til å skrive ut **elementer** til konsollen.

### Oppgave 1.2.4 (vanskelig.py)

Definer **a = "Så langt er Python-programmering", b = "ikke" og c = "vanskelig".** Lag en **print()-kommando** med enten **2** eller **3** av variablene ovenfor. La utskriften reflektere hvordan du føler det angående programmeringen så langt.

# Oppgaver: 1.3 Regneoperatorer

### Oppgave 1.3.1 (a_pluss_b_1.py)

Lag et program der du først definerer **a** og **b** til å være heltall/flyttall av eget valg (for eksempel 4.7 og 9). Legg tallene sammen inne i en **print()**-funksjon og kjøre programmet. Prøv nå å endre **a** og **b** for så å kjøre programmet igjen.
Husk å beskrive hva du gjør i koden din med relevante kommentarer.

### Oppgave 1.3.2 (pluss_minus_gange_dele_1.py)

*Regneoperatorene kan også brukes inne i **print()**-funksjonen.*
*Kjører du programlinjen: **print("Summen av 3 og 7 er:", 3+7)***
*vil **"Summen av 3 og 7 er: 10"** skrives ut i konsollen.*

**a)** Definer **a** og **b** til å være heltall av eget valg. La programmet skrive ut **verdien til a er * og verdien til b er *.**
**b)** Skriv ut summen(**+**) av **a** og **b** sammen med en passende svarsetning.
**c)** Skriv ut differansen(-) av **a** og **b** sammen med en passende svarsetning.
**d)** Skriv ut produktet(*****) av **a** og **b** sammen med en passende svarsetning.
**e)** Skriv ut **a** dividert(**/**) med **b** sammen med en passende svarsetning.

### Oppgave 1.3.3 (regnestykker.py)

*Husk **$5^3$** skrives som **5**3** i Python.*
Skriv et program som lagrer svaret på oppgave a) i variabelen **a** og svaret på oppgave b) i variabelen **b** osv. Skriv til slutt ut alle svarene i konsollen.
*NB: Husk at svaret ikke er det viktige i denne oppgaven, det viktige er å lære seg å skrive inn riktig i Python.*

**a)** $3 + 5 - 9$
**b)** $4 \cdot (9 + 2)$
**c)** $(9 - 2) \cdot (8 - 13)$
**d)** $15 \div 3$
**e)** $4^2$
**f)** $15^4$
**g)** $\frac{9-5}{4} - 6$ *NB: Parenteser brukes i Python når vi har brøker.*
**h)** $\frac{9 \cdot 5 - 8}{2} - 5^2$

### 1.3.4 (heltallsdivisjon.py)



*Husk at om vi skal heltallsdividere må vi bruke **//**. Om vi skal finne resten ved en heltallsdivisjon må vi bruke prosenttegn **%** (modulo)*
Skriv et program som lagrer svaret på oppgave a) i variabelen **a** og svaret på oppgave b) i variabelen **b** osv. Skriv til slutt ut alle svarene i konsollen.

**a)** Heltallsdivider 8 med 2
**b)** Finn resten i heltallsdivisjonen i oppgave a)
**c)** Heltallsdivider 9 med 2.
**d)** Finn resten i heltallsdivisjonen i oppgave c)
**e)** Heltallsdivider 17 med 7
**f)** Finn resten i heltallsdivisjonen i oppgave e)

### 1.3.5 (streng_pluss_streng.py)

Om vi bruker **+** imellom to streng-objekter så vil vi få et nytt streng-objekt lik kombinasjonen av de 2 strengene vi la sammen. Vi prøver selv.
Definer **a = "Programmering er ikke det viktigste i verden,"** og **b =" men om oppgavene er skrevet godt kan det hjelpe oss å lære mye nyttig."**
La **c = a + b** og skriv **c** til konsollen.

### 1.3.6 (bmi_kalkulator_1.py)

**BMI** (Body Mass Index) er en populær indikator som tar i bruk en persons høyde (i meter) og deres vekt (i kg) for å indikere om personen er **undervektig**(**BMI < 18.5**), **normalvektig**(**18.5 < BMI < 25**), **overvektig**(**25 < BMI < 30**), eller **sykelig overvektig**(**BMI > 30**). Den er relativt nøyaktig, men har blitt kritisert fordi den ikke tar alle kroppstyper i betraktning. En relativt slank muskelbunt vil kunne komme til å bli definert som overvektig av indikatoren ettersom muskler veier mer enn fett.

**Selve oppgaven:**
I denne oppgaven skal du lage din egen BMI-kalkulator. Da får du bruk for formelen:

$$BMI = \frac{vekt\ (i\ kg)}{høyde^2\ (i\ meter)}$$

Begynn med så sette **vekt = 80** og **hoyde = 1.80**
Lag en variabel **BMI** som regnes ut ved hjelp av **vekt** og **høyde**.
La programmet skrive ut "En person med vekten: **vekt** og høyden: **hoyde** meter har BMI = **BMI"**
Prøv gjerne litt forskjellige høyder og vekter.

### 1.3.7(Partallssjekk_1.py)

Sett **var = 3** (3-tallet skal kunne forandres til et hvilket som helst heltall). Lag et program som skriver ut 0 dersom **var** er et partall og 1 ellers. (Hint: Modulo)

# Oppgaver: 1.4 Forandring av datatyper og input()-funksjonen

### 1.4.1 (streng_til_heltall.py)

Skriv **a = "3"** (Et streng-objekt) og **b = 5.3** Vi ønsker å finne summen av disse to tallene. Vi kan dessverre ikke bare utføre **a + b** med en gang. Gjør nødvendige endringer på **a** for at vi skal kunne skrive summen til konsollen med kommandoen **print(a+b).**

### 1.4.2 (input_output.py)

Nå skal du lage et program som ber om informasjon ifra brukeren. Programmet skal motta én bit med informasjon som skal lagres i variablene **a** . La programmet skrive ut en passende forespørselssetning (for eksempel: **"I variabel a vil jeg lagre:"**)
La programmet lese tilbake informasjonen det har mottatt til brukeren ved å skrive det ut i konsollen. Husk at informasjonen du skriver inn

### 1.4.3 (dobbel_1.py)

Lag et program som tar imot ett **flyttall ifra bruker**. Skriv en passende forespørsel til bruker. La programmet skrive ut det dobbelte av tallet du skrev inn.

### 1.4.4 (navn_og_alder_3.py)

Du skal nå modifisere programmet du lagde i ***Oppgave 1.2.2***
Gjør slik at programmet først ber om navnet ditt og deretter ber om alderen din og lagre disse i variabler. La programmet skrive ut samme setning som tidligere, bare denne gangen med input ifra når du faktisk kjører programmet.

### 1.4.5 (pluss_minus_gange_dele_2.py)

Modifiser programmet du lagde i ***oppgave 1.3.2***. La programmet motta **a** og **b** ifra brukeren for så å skrive ut summen, differansen, produktet og den dividerte som tidligere.

### 1.4.6 (bmi_kalkulator_2.py)

Rediger BMI kalkulatoren du lagde i **oppgave 1.3.6** slik at du nå spør brukeren av programmet om vekt (i kg) og høyde (i meter). La programmet fremdeles skrive ut BMI-en til brukeren.

### 1.4.7 ( fahrenheit_til_celsius_1.py)

I noen få land (Hovedsakelig i USA) bruker de en annen enhet for å beskrive temperaturer enn det vi gjør. Når vi sier at det er 20 grader ute mener vi nemlig 20



grader **Celsius.** I USA bruker de Fahrenheit. Når en amerikaner bryter ut **"It's a hundred degrees outside"** betyr det altså ikke at det regner ild ifra himmelen.

**Selve oppgaven:**
Formelen for omregning ifra Fahrenheit til Celsius er gitt ved:

$$°\boldsymbol{C} = (°\boldsymbol{F} - 32) \cdot \frac{5}{9}$$

a) Lag en variabel **fahrenheit** med verdi 100. Bruk formelen over til å finne ut hva en amerikaner faktisk mener når han sier **"It's a hundred degrees outside".** Skriv ut svaret i konsollen.
b) Gjør en endring i programmet slik programmet spør brukeren om en **fahrenheit** verdi som lagres i variabelen **fahrenheit.** La programmet skrive ut hvor mange celsius dette tilsvarer med en passende svarsetning.
**c) Utfordring:** Lag et nytt program der du gjør om celsius til fahrenheit. Husk å gi programmet et passende navn.

# Oppgaver: 1.5 Sammenlikningsoperatorer og if/else – setninger

### 1.5.1 (storre.py)

Definer **x** til å være **5**. Lag et program som skriver ut **"x er større enn 4"** hvis **x > 4** og skriver ut "x er ikke større enn 4" ellers. Etter du har kjørt programmet en gang prøv å endre x-verdien til 3 for deretter å kjøre programmet igjen.

### 1.5.2 (navn.py)

Lag et program som stiller brukeren spørsmålet "Hva heter du?" og samtidig tar imot et svar og lagrer det i en variabel **navn**. La programmet skrive ut til brukeren "Det var et fint navn!" om det er navnet ditt og "Hyggelig å hilse på deg **navn**" ellers.

### 1.5.3(dikt_ja_nei.py)

Lag et program som spør brukeren om han ønsker å høre ett dikt. Om brukeren svarer ja la programmet skrive ut:

***Hold fast to dreams***
***For if dreams die***
***Life is a broken-winged bird***
***That cannot fly.***

***Langston Hughes***

Ellers la programmet skrive ut:

**Ok...**

### 1.5.4 (sammenlikninger i konsollen)

Bruk Python til å sjekke om følgende påstander er "SANNE" (TRUE)

a) $3 - 5 + 2 \leq 8 - 9$
b) $19 - 16^2 = 493$
c) $3 - 8 \neq 5$

$$\text{"Hei"} = \text{"}Hei\text{"}$$

### 1.5.5 (sammenlikninger.py)

Definer variabelen **x = 5**. Lag et program som skriver ut **"Ja x er større enn 3"** hvis **x > 3** og **"x er ikke større enn 3"** ellers. Kjør programmet for fler verdier av x.



# Oppgaver: 1.6 Funksjoner

**1.6.1 (trippel_1.py)**

Lag en funksjon du kaller **trippel** og la den ta imot ett argument **x**. La funksjonen returnere 3***x**. Skriv: print(trippel(4)) nederst i koden for å teste koden din.

**1.6.2 (dobbel_2.py)**

Du skal nå modifisere programmet du lagde i Oppgave 1.4.3
Lag en funksjon du kaller dobbel som tar imot ett argument . Bruk denne funksjonen for å doble det du skriver inn i input feltet. Skriv ut svaret.

**1.6.3 (print_i_funksjonen.py)**

Lag en funksjon som skriver ut: "Godt Jobba" når du kaller på den. Twist: Du skal bruke print()-funksjonen i denne oppgaven men denne gangen inne i selve funksjonen.

**1.6.4 (legge_sammen_strenger.py)**

Lag en funksjon som tar imot 2 argumenter. Begge argumentene skal være tekst-strenger. La funksjonen legge sammen de 2 tekst-strengene med en **+** og returner resultatet.
Prøv å kalle på funksjonen (inne i en print()-funksjon om du ønsker å se resultatet.)

**1.6.5 (det_virker.py)**

Lag en funksjon kalt **det_virker** som ikke tar imot noen argumenter, men som returnerer teksten "Det virker!". Bruk **print()** når du kaller på funksjonen. **NB**: Husk **()** etter funksjonsnavnet både når du skriver funksjonen og når du kaller på funksjonen.



# Oppgaver: 1.7 Lister og listeoperasjoner

### 1.7.1 (listeoperasjoner.py)

a) Lag en variabel **liste_1** som du tilordner (gir) verdien **["En", 2, 3.0]** og lag enda en variabel **liste_2** som du gir verdien **[4, "Fem"]**.

b) Lag enda en ny variabel **liste_3**, la denne listen være lik **liste_1 + liste_2**. Skriv ut liste 3 til konsollen med **print()** kommandoen.

c) Skriv **liste_3.append(6)** for å legge til heltallet 6 i slutten av **liste_3**.

d) Skriv nå **print(liste_3[0])** for å skrive ut det første elementet i **liste_3**. Skriv også ut det andre og femte elementet i liste 3.

e) Skriv ut det siste elementet i **liste_3** ved å skrive **liste_3[-1]**. Skriv også ut det nest siste og det tredjesiste elementet.

### 1.7.2 (riktig.py)

Uten å programmere
Hva tror du følgende program ville skrevet ut?

```
Liste_a = ["kosthold"," og", "for" "humør"]
Liste_b = ["Riktig", "hjelper", "trening"]
```

**a)**

```
print(Liste_b[0], Liste_a[0], Liste_b[1], Liste_a[2], liste_b[2], liste_a[1], liste_a[3])
```

**b)**

```
print(Liste_b[-3], Liste_b[-1], Liste_b[-2], Liste_a[-2], Liste_a[-1], Liste_a[-3],
Liste_a[-4])
```

**c)**
Bruk listene ovenfor til å lage en **print()-funksjon** som skriver ut følgende setning:

**"Riktig humør hjelper for trening og kosthold"**

## Oppgaver: 1.8 while-løkker

**1.8.1 (null_til_fire.py)**

Gi **x** verdien **0**. Lag en **while-løkke** som skriver ut **x** til konsollen så lenge **x < 5**. La **x** øke med **1** for hver gjennomgang av løkken.

**1.8.2 (tiere.py)**

Gi **a** verdien 10. Bruk en while løkke til å skrive ut alle tier-tall (10,20,30,40...) helt opp til og med 110. Når løkken er ferdig la programmet skrive: "Løkken er ferdig."

1984

## Oppgaver: 1.9 for-løkker og range()-funksjonen

**1.9.1 (tresko.py)**

Lag en **for**-løkke som skriver ut alle elementene i listen **[2,"små", "tresko"]**

**1.9.2 (range_test.py)**

For å løse oppgavene under skal du bruke **range()**-funksjonen og **for**-løkker aktivt

a) Skriv ut tallene fra og med 0 til 10
b) Skriv ut tallene fra og med 4 til 12
c) Skriv ut partallene fra og med 6 til og med 20
d) Skriv alle tallene delelige med 3 fra og med 9 til og med 99

**1.9.3 (trippel_2.py)**

I denne oppgaven skal du modifisere **oppgave 1.6.1 (trippel_1.py)**
Legg tallene 1, 5 og 7 inn i en liste, velg navn på listen selv. Lag en for-løkke som kaller på funksjonen **trippel()** for hvert av elementene i denne listen.

Skriv resultatene ut i konsollen.

**1.9.4 (stå_på.py)**

Definer **liste_1 = ["Du", "står", "på.", "Det", "vil", "lønne", "seg."]**

Lag en **for-løkke** som skriver ut alle ordene i listen over ord for ord.

## Oppgaver: 1.10 and, or og elif

**1.10.1 (hvaskjer.py)**

**Anta i oppgaven under at vi har satt x = 3, y = 50 og z = True**
**Hvilken utskrift gir følgende utsagn? Test koden selv for fasit.**

```
if x>2 and y>300:
    print(302)
elif x>2 or y >300:
    print(2)
else:
    print(0)
```



# Oppgaver: 1.11 Enkel bibliotek import og random-modulen

**1.11.1 (while_not_6.py)**

## a)

La **counter = 0** og **terningkast = 0.** Lag en simulering av en terning i en **while-løkke**. La while løkken gå så lenge terningkastet ikke er lik 6. For hvert "terningkast", la **counter** øke med 1.

## b)

La **counter_2 = 0** og **sum_terningkast = 0**. **.** Lag en simulering av summen av to terning i en **while-løkke**. La while løkken gå så lenge summen ikke er lik 12. For hvert "kast med to terninger" la **counter** øke med 1. Bruk counter_2 til å finne ut hvor mange kast du som gjennomføres