# 1 Tekst baserte interaksjoner med brukeren

## 1.1 Skrive ut tekst: `print()` funksjonen

Den berømte "Hello World" programmet: Skriver ut en tekst (streng) som hilser verden.

- Vi kan definere en streng (tekst) ved å bruke `""` eller `''`.
- Vi kan skrive ut tekst ved å *kalle på funksjonen* `print()`.
- Vi kan skrive flere strenger samtidig med funksjonen `print()`, en etter den andre, og det vil være ett mellomrom mellom dem.
- Hvis vi *kaller på* funksjonen flere ganger, blir det et linjeskift innimellom.

[1]:
```
# Her skriver vi ut 2 strenger og det vil være ett mellomrom mellom dem
print("Hallo Verden!", "Jeg lærer Python!")
# Hvis vi kaller på funksjonen igjen, blir det et linjeskift mellom
# den første og den andre
print("Jeg er 32 år gammel.")
```

```
Hallo Verden! Jeg lærer Python!
Jeg er 32 år gammel.
```

### 1.1.1 Første variabler med tekst

- En variabel la oss lagre en verdi. Her, tilordner vi verdien `"Hallo Verden! Jeg lærer Python!"` til variabelen `melding1`.
- Vi kan velge navnet på variabelen vår. Det anbefales at du bruker et navn som gjenspeiler verdien knyttet til variabelen.
- Når variabelen har blitt opprettet, kan vi bruke verdien som er lagret i variabelen så mange ganger vi ønsker ved å bruke variabelnavnet.
- For eksempel kan vi skrive ut verdien lagret i en variabel med `print()` funksjonen

[2]:
```
# opprett variabler som innholder tekst
melding1 = "Hallo Verden! Jeg lærer Python!"
melding2 = "Jeg er 32 år gammel."
# Skriv ut variablene
print(melding1, melding2)
```

```
Hallo Verden! Jeg lærer Python! Jeg er 32 år gammel.
```

- Variabler er kjempe hjelpsomme fordi de lar oss gjenbruke en verdi og hvis vi vil endre verdien, er det bare å oppdater variabel en gang.
- Vi kan inkorporere verdien lagret i en variabel i en (streng) tekst ved å bruke f-strenger. En f-streng er en streng som vi definerer ved å bruke `f` rett før teksten og `{}` rundt hver variabel vi ønsker å bruke i teksten.
- Vi kan også tilordne tallverdier til en variabel.

[3]:
```
navn = "Odin"
alder = 32

# med f-strenger kan vi plassere variabelverdier inne i strengen
melding1 = f"Hallo {navn}! Jeg lærer Python!"
melding2 = f"Jeg er {alder} år gammel."
print(melding1, melding2, f"Hvor gammel er du, {navn}?")
```

```
Hallo Odin! Jeg lærer Python! Jeg er 32 år gammel. Hvor gammel er du, Odin?
```

### 1.2 Interaksjon med brukeren: `input()` funksjonen

- `input()` funksjonen la oss be brukeren om å gi oss en strengverdi. Vi kan få tak i verdien og opprette en variabel som inneholder verdien, slik at vi kan gjenbruke verdien senere i koden.

[4]:
```python
navn = "Odin"
alder = 32

# Brukeren må gi oss et navn når de kjører koden
navn = input("Hva heter du? ")
melding1 = f"Hallo {navn}! Jeg lærer Python!"
melding2 = f"Jeg er {alder} år gammel."

# Vi ser at navn (og alder) har en forskjellig verdi nå
print(melding1, melding2)
alder = input(f"Hvor gammel er du, {navn}? ")
```

```
Hallo Vilma! Jeg lærer Python! Jeg er 32 år gammel.
```

### 1.3 Enkle matematiske operasjoner: beregne prisen for en natt på et hotell

La oss forestille oss at vi vil beregne prisen for en natt på et hotell og la oss si at en natt på et hotell består av ett rom for alle og én frokost per person. Det betyr at for å beregne prisen, bruker vi formelen:

$$\texttt{pris\_rom} + \texttt{antall\_personer} \times \texttt{pris\_forkost}$$

[5]:
```python
# Definer prisene
pris_frokost = 149.90
pris_rom = 1000

# Skriv ut priser på en strukturert måte
# Med \n kan vi spesifisere at vi vil ha et linjeskift
melding = f"***Priser***\nFrokost: {pris_frokost} \nRom. {pris_rom}\n"
print(melding)

n_personer = 3
print(f"Dere er {n_personer} personer.")

# Beregne den totale prisen for natten
pris = pris_rom + n_personer*pris_frokost
print(f"Da bør dere betale {pris}Kr.")
```

```
***Priser***
Frokost: 149.9
Rom. 1000

Dere er 3 personer.
Da bør dere betale 1449.7Kr.
```

### 1.3.1 Prisen for en natt på et hotell med interaksjon med brukeren

Vi kan gjøre koden vår litt mer interaktiv ved å be brukeren om å angi antall personer istedenfor å definere antallet i koden vår... Men vi skall se at det gir en feil!

Kommentar om utførelsen: Det oppstår en feil!

[6]:
```
pris_frokost = 149.90
pris_rom = 1000
melding = f"***Priser***\nFrokost: {pris_frokost} \nRom. {pris_rom}\n"
print(melding)

# Nå ber vi brukeren om å angi antall personer
n_personer = input("Hvor mange personer er dere?")
print(f"Dere er {n_personer} personer.")

# /!\ Det utløser en feilmelding /!\:
# "TypeError: can't multiply sequence by non-int of type 'float'"
pris = pris_rom + n_personer*pris_frokost
print(f"Da bør dere betale {pris}Kr.")
```

```
***Priser***
Frokost: 149.9
Rom. 1000

Dere er 3 personer.
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[6], line 12
      8 print(f"Dere er {n_personer} personer.")
     10 # /!\ Det utløser en feilmelding /!\:
     11 # "TypeError: can't multiply sequence by non-int of type 'float'"
---> 12 pris = pris_rom + n_personer*pris_frokost
     13 print(f"Da bør dere betale {pris}Kr.")

TypeError: can't multiply sequence by non-int of type 'float'
```

## 1.4 Datatyper: flyttall, heltall og streng

Vi ser i cellen før at feilmeldingen gir oss nyttig information:

- Linjen som utløser en feil. Her er det `pris = pris_rom + n_personer*pris_frokost` .
- Det som presist utløser feilen er uthevet. Her er det `n_personer*pris_frokost`
- Feiltypen. Her er det `TypeError`. (Så problemet stammer sannsynligvis fra noe som ikke har riktig type).
- Meldingen sier at det går galt når vi prøver å multiplisere noe med noe annet (så det som ikke har riktig typen er sannsynligvis `pris_frokost` eller `n_personer`).

Kort sagt: **feilmeldinger er ikke farlige men kjempe hjelpsomme fordi de gir oss hint om hvordan vi kan fikse koden vår.**

Men hva er en "type"? Hvordan man sjekker typen til en variabel?

- Vi bruker funksjonen `type()` for å sjekke typen til en variabel.

- Når vi feilsøker, er det ganske vanlig å bruke funksjonene `print` og `type` for å sjekke statusen til våre variabler.

[7]:
```
# La oss opprette noen variabler før vi sjekker typen deres
pris_frokost = 149.90
pris_rom = 1000
melding = f"***Priser***\nFrokost: {pris_frokost} \nRom. {pris_rom}\n"
n_personer = input("Hvor mange personer er dere?")

# Vi ser at de har ulike typer!
print(f"type(pris_frokost):                {type(pris_frokost)}")
print(f"type(pris_rom):                    {type(pris_rom)}")
print(f"type(pris_frokost + pris_rom):     {type(pris_frokost + pris_rom)}")
print(f"type(melding):                     {type(melding)}")

# Også ga vi et heltall men typen er streng!
print(f"type(n_personer):                  {type(n_personer)}")
```

```
type(pris_frokost):                <class 'float'>
type(pris_rom):                    <class 'int'>
type(pris_frokost + pris_rom):     <class 'float'>
type(melding):                     <class 'str'>
type(n_personer):                  <class 'str'>
```

### 1.4.1 Datatyper: konvertering

- Hver variabel har en spesifikk datatype.
- For å utføre matematiske operasjoner datatypen må være et tall (heltall `int` eller desimaltall `float`), men ikke en tekst (streng `str`).
- Vi kan konvertere datatypen til en variabel med for eksempel funksjonene `int()`, `float()`, `str()` når det er mulig.

[8]:
```
# Her konverterer vi et heltall (int) til et flyttall (float)
print(f"float(pris_rom)):   {float(pris_rom)}")

# Vær forsiktig når du konverterer flyttall til heltall!
# int(149.90) = 149!
print(f"{pris_frokost} != {int(pris_frokost)}")

# Vær forsiktig når du konvertere et uttrykk
# Utrykket evalueres før det konverteres!
print(f"{4+7} er lik {str(4+7)} men {'4+7'} er forskjellig.")
```

```
float(pris_rom)):   1000.0
149.9 != 149
11 er lik 11 men 4+7 er forskjellig.
```

### 1.4.2 Datatyper: konvertering er ikke alltid mulig!

- En streng som representerer et flyttall (desimaltall) kan ikke konverteres til et heltall.
- En streng som ikke representerer et tall kan ikke konverteres til et flyttall eller et heltall.

Kommentar om utførelsen: Det oppstår en feil om de to siste linjene ikke er kommenterte. Og kjøring koden gir oss ingen informasjon om den andre feilen så lenge den første feilen ikke er fikset!

[9]:
```
# Det er ikke alltid mulig og konvertere en variabel
float("3.0")     # Ja
# /!\ Det utløser en feilmelding /!\:
int("3.0")       # Nei
# /!\ Det utløser en feilmelding /!\:
float("Hello")   # Nei
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[9], line 4
      2 float("3.0")     # Ja
      3 # /!\ Det utløser en feilmelding /!\:
----> 4 int("3.0")       # Nei
      5 # /!\ Det utløser en feilmelding /!\:
      6 float("Hello")

ValueError: invalid literal for int() with base 10: '3.0'
```

### 1.4.3 Prisen for en natt på et hotell med korrekt type

- Ved å sjekke typen til alle variablene, fikk vi vite at `input()` alltid returnerer en streng (tekst) selv om brukeren skriver et tall. Det er forskjellen mellom `3` som er et tall og `"3"` som er en streng.
- Feilmeldingen vi fikk ble utløst fordi vi prøvde å multiplisere en streng (`"3"`) med et flyttall `149.90`.
- Det er bare å konvertere strengen til et tall (heltall eller flyttall) for å løse problemet.

[10]:
```
pris_frokost = 149.90
pris_rom = 1000
melding = f"***Priser***\nFrokost: {pris_frokost} \nRom. {pris_rom}\n"
print(melding)

# Nå konverterer vi strengen som input funksjonen returnerer til et heltall
n_personer = int(input("Hvor mange personer er dere?"))
print(f"Dere er {n_personer} personer.")

pris = pris_rom + n_personer*pris_frokost
print(f"Da bør dere betale {pris}Kr.")
```

```
***Priser***
Frokost: 149.9
Rom. 1000

Dere er 3 personer.
Da bør dere betale 1449.7Kr.
```