1 Løkker

Vi trenger ofte å bruke `while`-løkker i en svært spesifikk sammenheng (for eksempel for å iterere over en sekvens), noe som fører til den samme while-løkkestrukturen. I slike situasjoner er en annen løkke tilgjengelig i de fleste programmeringsspråk som forenkler syntaksen samtidig som koden blir mer effektiv: `for`-løkken.

1.1 `for`-løkker: gjenta en kodeblokk ved å iterere gjennom sekvenser

- En `for`-løkke gjør det mulig å gjenta en kodeblokk flere ganger. Den brukes vanligvis når vi vil gjenta handlinger for hvert element i en sekvens, for eksempel en liste eller en streng.
- I tillegg gir en `for`-løkke oss direkte tilgang til hvert element i sekvensen, én etter én.

Syntaks:

```
for element in sekvens:
    # kodeblokken med innrykk her
    # [...]
```

Merk at vi velger navnet på variabelen som vil gi oss tilgang til elementene én etter én. Det er en god praksis å gi variabelen et navn som samsvarer med den faktiske innholdet i sekvensen.

[1]:
```
min_streng = "Hallo, verden!"
min_liste = [2024, 0.1, "DIGI", True, None]
print(f"{min_liste = }")
print(f"{min_streng = }")

print("Iterere over en streng:")
for bokstav in min_streng:
    # Vi har tilgang til bokstavene i strengen
    print(bokstav)

print("Iterere over en liste:")
for element in min_liste:
    # Vi har tilgang til elementene i listen
    print(element)
```

```
min_liste = [2024, 0.1, 'DIGI', True, None]
min_streng = 'Hallo, verden!'
Iterere over en streng:
H
a
l
l
o
,

v
e
r
d
e
n
!
Iterere over en liste:
```

```
2024
0.1
DIGI
True
None
```

1.1.1 Definere generelle funksjoner på sekvenser

Siden `for`-løkker fungerer for alle typer sekvenser, kan vi definere generelle funksjoner som kan ta en liste eller en streng som argument.

[2]:
```python
def iterere_gjennom_sekvens(sekvens):
    # Denne koden fungerer med strenger og lister!
    for element in sekvens:
        print(f"{element}")

print("Iterere over en streng:")
iterere_gjennom_sekvens(min_streng)
print("Iterere over en liste:")
iterere_gjennom_sekvens(min_liste)
```

```
Iterere over en streng:
H
a
l
l
o
,

v
e
r
d
e
n
!
Iterere over en liste:
2024
0.1
DIGI
True
None
```

1.1.2 funksjonen `range()`: gir en sekvens av tall mellom to tall.

Noen ganger er vi mer interessert i å iterere over indeksene til sekvensen i stedet for selve sekvensen. I så fall kan vi bruke funksjonen `range()` som gir oss en sekvens av tall.

Syntaks `range()` funksjonen + en `for`-løkke:

```python
for i in range(slutt):
    # kodeblokken med innrykk her
```

```
for i in range(start, slutt):
    # kodeblokken med innrykk her

for i in range(start, slutt, steg):
    # kodeblokken med innrykk her
```

[3]:
```
print("Med range(5)")
# Syntaksen: range(slutt)
# (sluttverdien er ikke inkludert, sekvensen stopper rett før)
for x in range(5):
    print(x)

print("Med range(3, 6)")
# Syntaksen: range(start, slutt)
for x in range(3, 6):
    print(x)

print("Med range(10,20,2)")
# Syntaksen: range(start, slutt, steg)
for x in range(10,20,2):
    print(x)
```

```
Med range(5)
0
1
2
3
4
Med range(3, 6)
3
4
5
Med range(10,20,2)
10
12
14
16
18
```

Vi kan bruke `range()` for å få tilgang til indeksene i sekvensen

[4]:
```
min_streng = "Hallo, verden!"
min_liste = [2024, 0.1, "DIGI", True, None]
print(f"{min_liste = }")
print(f"{min_streng = }")

def iterere_gjennom_sekvens_range(sekvens):
    # Hvor lang er sekvensen? Bruk "len()" funksjon
    lengde = len(sekvens)
    # Denne koden fungerer med strenger og lister!
    for i in range(lengde):
        print(f"Indeks {i} : {sekvens[i]}")
```

```
iterere_gjennom_sekvens_range(min_streng)
iterere_gjennom_sekvens_range(min_liste)
```

```
min_liste = [2024, 0.1, 'DIGI', True, None]
min_streng = 'Hallo, verden!'
Indeks 0 : H
Indeks 1 : a
Indeks 2 : l
Indeks 3 : l
Indeks 4 : o
Indeks 5 : ,
Indeks 6 :
Indeks 7 : v
Indeks 8 : e
Indeks 9 : r
Indeks 10 : d
Indeks 11 : e
Indeks 12 : n
Indeks 13 : !
Indeks 0 : 2024
Indeks 1 : 0.1
Indeks 2 : DIGI
Indeks 3 : True
Indeks 4 : None
```

1.1.3 `For`-løkke: Eksempel

Vi vil beregne gjennomsnittet av en liste av karakterer med en `for`-løkke. Som alltid i programmering er det viktig å skrive alle instruksjonene på norsk før vi skriver i python, og gå fra hovedmålet til mer og mer små og presise instruksjoner. Når vi har beskrevet hele algoritmen på norsk, kan vi prøve å identifisere verktøyene vi har i Python som kan hjelpe oss med å implementere hver instruksjon.

---

1. Hovedmålet

Beregne gjennomsnittet av en liste av karakterer.

2. Litt mer presise instruksjoner

For å beregne gjennomsnittet må vi:

1. Beregne den totale summen
2. Dele summen på hvor mange karakterer det var i summen

3. Enda mer presise instruksjoner

1. For å beregne summen kan vi:
    1. Iterere over listen for å få tak i karakterene én etter én
    2. Legge den nåværende karakteren til summen av alle de foregående karakterene så langt.
    3. … Men vi må *huske* hva summen var så langt når vi legge til den nåværende karakteren til summen av alle de foregående elementene: Vi trenger en variabel som skal oppdateres
2. For å dele summen på hvor mange elementene det var i summen, må vi:
    1. Beregne hvor mange elementer det er i listen (og *huske* det)
    2. Dele summen vi har allerede beregnet på antallet karakterer

---

Hint med `for`-løkker

- Når vi jobber med `for`-løkker, trenger vi veldig ofte å initialisere en variabel før løkken som vi skal oppdatere inni `for`-løkken for å beregne resultatet.
- Generelt i programmering, hver gang vi må *huske* noe, bruker vi en variabel.

[5]:
```
karakterene = [12, 8, 15, 6, 10, 12, 3, 0, 19, 36]

# -------- 1. Beregne den totale summen -----------

# 1.3: Initialisere variabelen som skal
# - hjelpe oss å huske summen så langt i for-løkken
# - gi oss den totale summen etter løkken
total_sum = 0
# 1.1:  Iterere
for karakter in karakterene:
    # 1.2 Legge til
    total_sum = total_sum + karakter

# ---- 2. Dele summen på antallet karakterer -------

# 2.1 Beregne antallet karakterer
lengde = len(karakterene)
# 2.2 Beregne antallet karakterer
gjennomsnittet = total_sum / lengde

print(gjennomsnittet)
```

```
12.1
```

1.2 `While`-løkker og `for`-løkker

- `for`-løkken brukes når vi vet antall ganger vi ønsker å gjenta koden på forhånd eller nå vi vet at vi vil iterere over en hel sekvens.
- `while`-løkken brukes når vi ikke kjenner antallet gjentakelser på forhånd, men vil fortsette å gjenta så lenge en betingelse er sann.

1.2.1 Iterere gjennom sekvens med en `while`-løkke

[6]:
```
min_streng = "Hallo, verden!"
min_liste = [2024, 0.1, "DIGI", True, None]
print(f"{min_liste = }")
print(f"{min_streng = }")

def iterere_gjennom_sekvens_while(sekvens):
    # Hvor lang er sekvensen
    lengde = len(sekvens)

    # Initialiser indeksvariabel for å *huske* hvor vi er i sekvensen
    i_element = 0
```

```
    while i_element < lengde:
        print(f"Indeks {i_element} : {sekvens[i_element]}")

        # Oppdater indeksvariabelen
        i_element = i_element + 1

iterere_gjennom_sekvens_while(min_streng)
iterere_gjennom_sekvens_while(min_liste)
```

```
min_liste = [2024, 0.1, 'DIGI', True, None]
min_streng = 'Hallo, verden!'
Indeks 0 : H
Indeks 1 : a
Indeks 2 : l
Indeks 3 : l
Indeks 4 : o
Indeks 5 : ,
Indeks 6 :
Indeks 7 : v
Indeks 8 : e
Indeks 9 : r
Indeks 10 : d
Indeks 11 : e
Indeks 12 : n
Indeks 13 : !
Indeks 0 : 2024
Indeks 1 : 0.1
Indeks 2 : DIGI
Indeks 3 : True
Indeks 4 : None
```