Gruppetime uke 5
IN1000

Ny sjanse til å levere innlevering 3 og 4!

- Gjelder for de som ikke allerede har levert
- Fristen for å levere er 25. September
- Innlevering 5 beholder samme frist (2.oktober)

Plan for timen

- Repetisjon
  - for in range()
  - prosedyrer uten parametere
- Prosedyrer med parametere
- Funksjoner
- Lese fra filer
- Skrive til filer
- Egenjobbing

Repetisjon

For in range() - noen oppklaringer

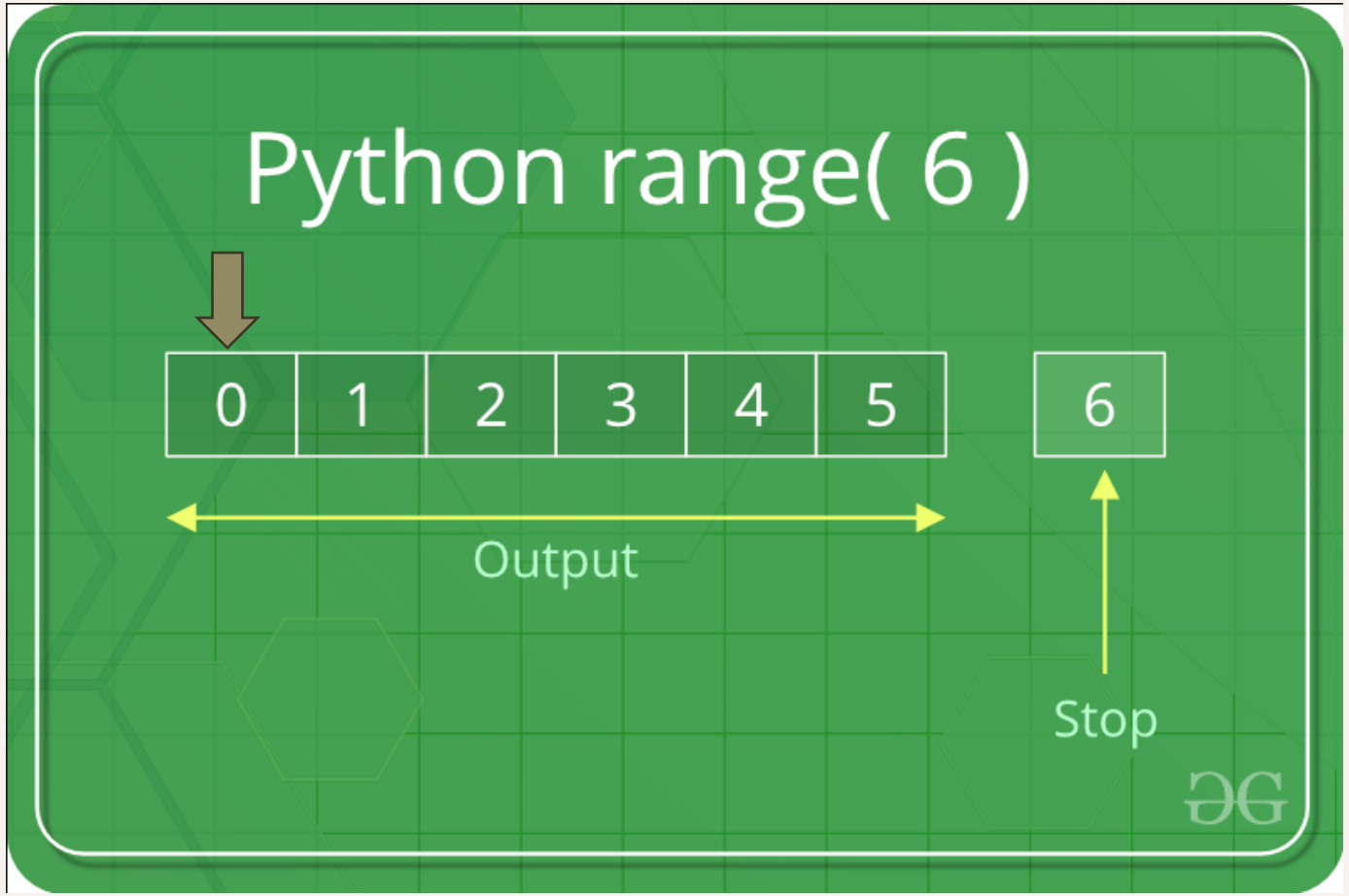
Det er helt vilkårlig hva navnet til denne variabelen er. Det er vanlig å bruke i for indeks, men man kan kalle den hva man vil.

range() lager en sekvens med tall. i vil starte som det første tallet i denne sekvensen og for hver gang løkken kjører vil i bli det neste tallet i sekvensen

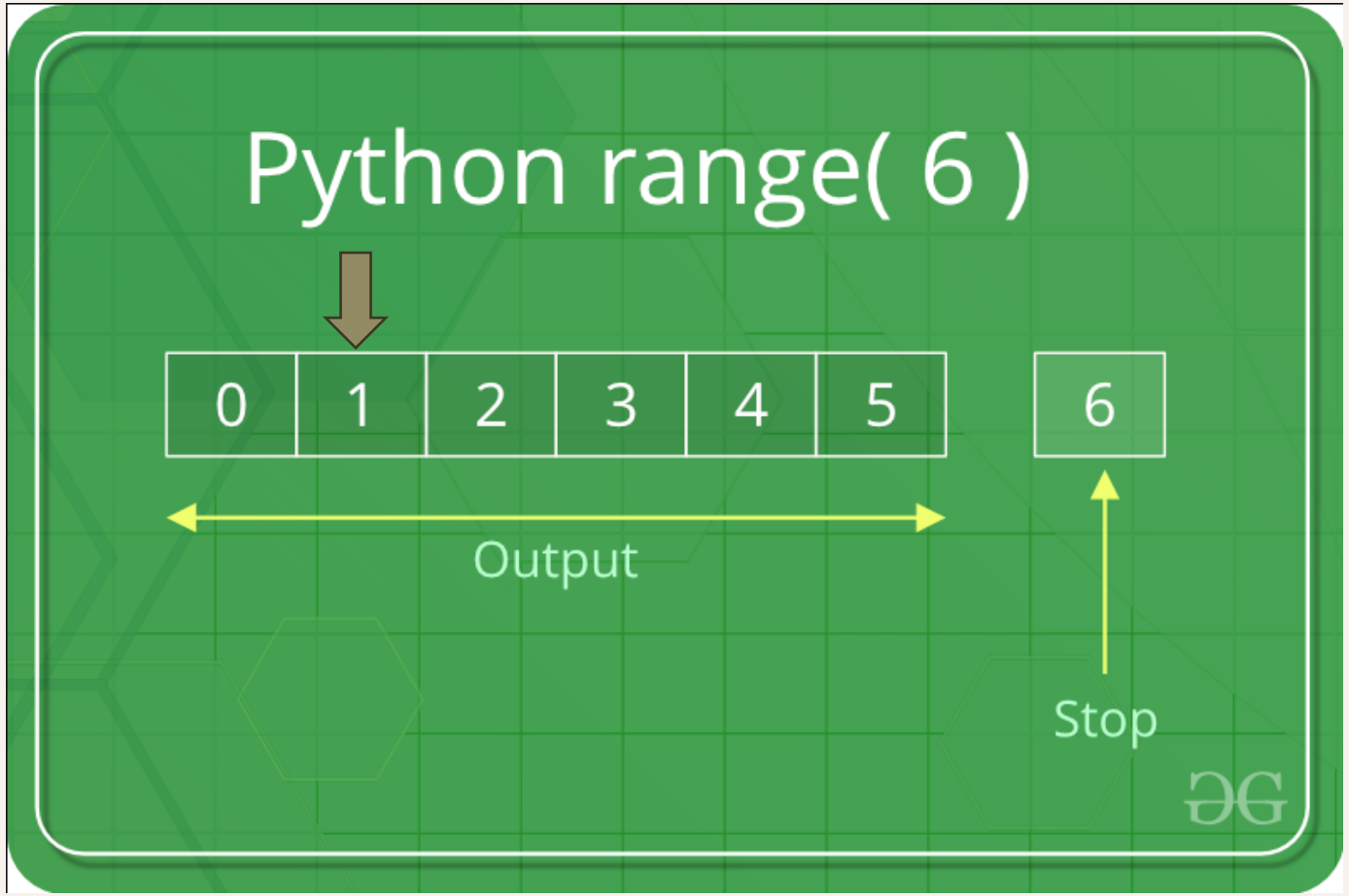


Illustrasjon av for-løkke med sluttverdi

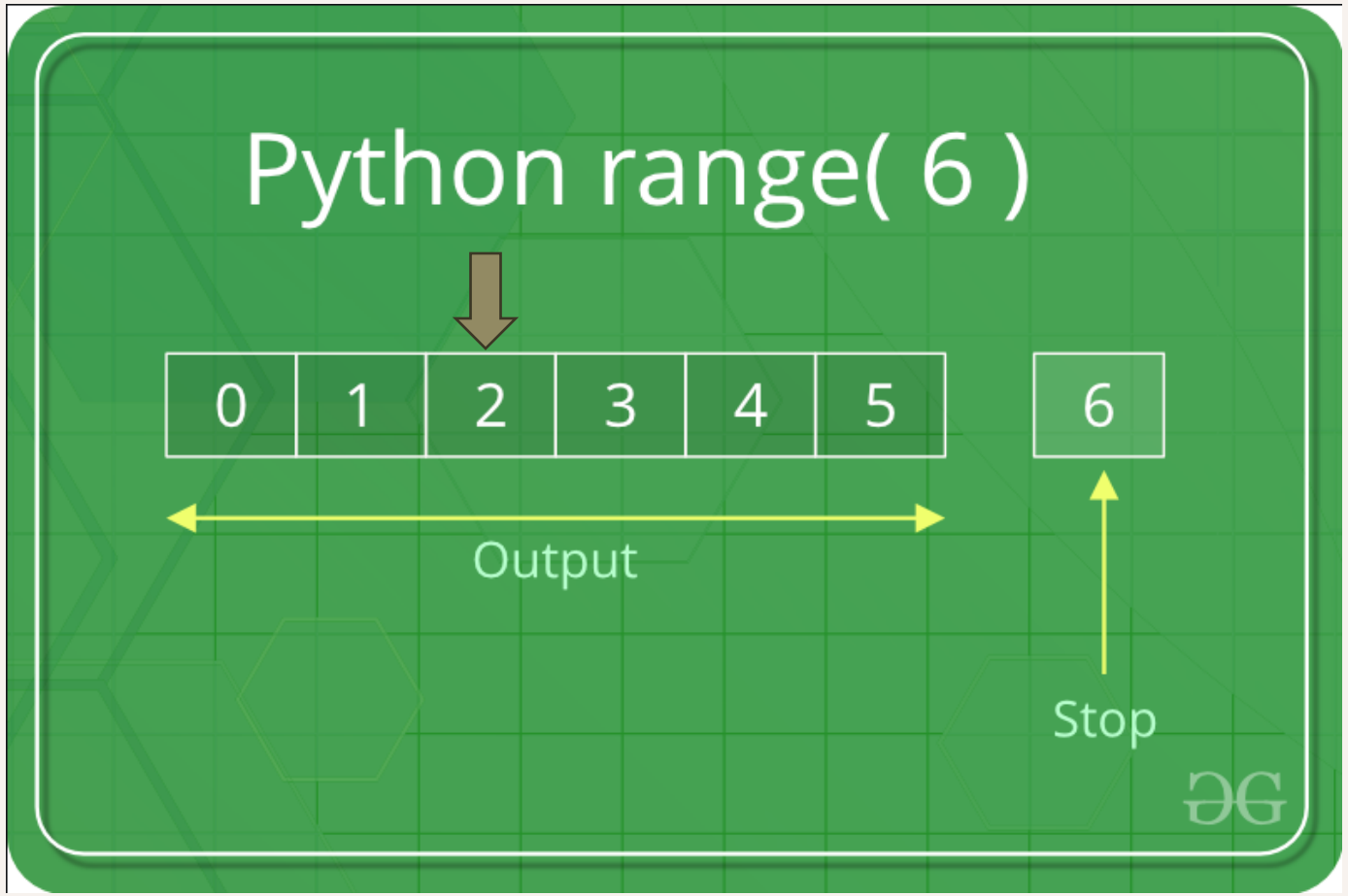
Første iterasjon: i = 0



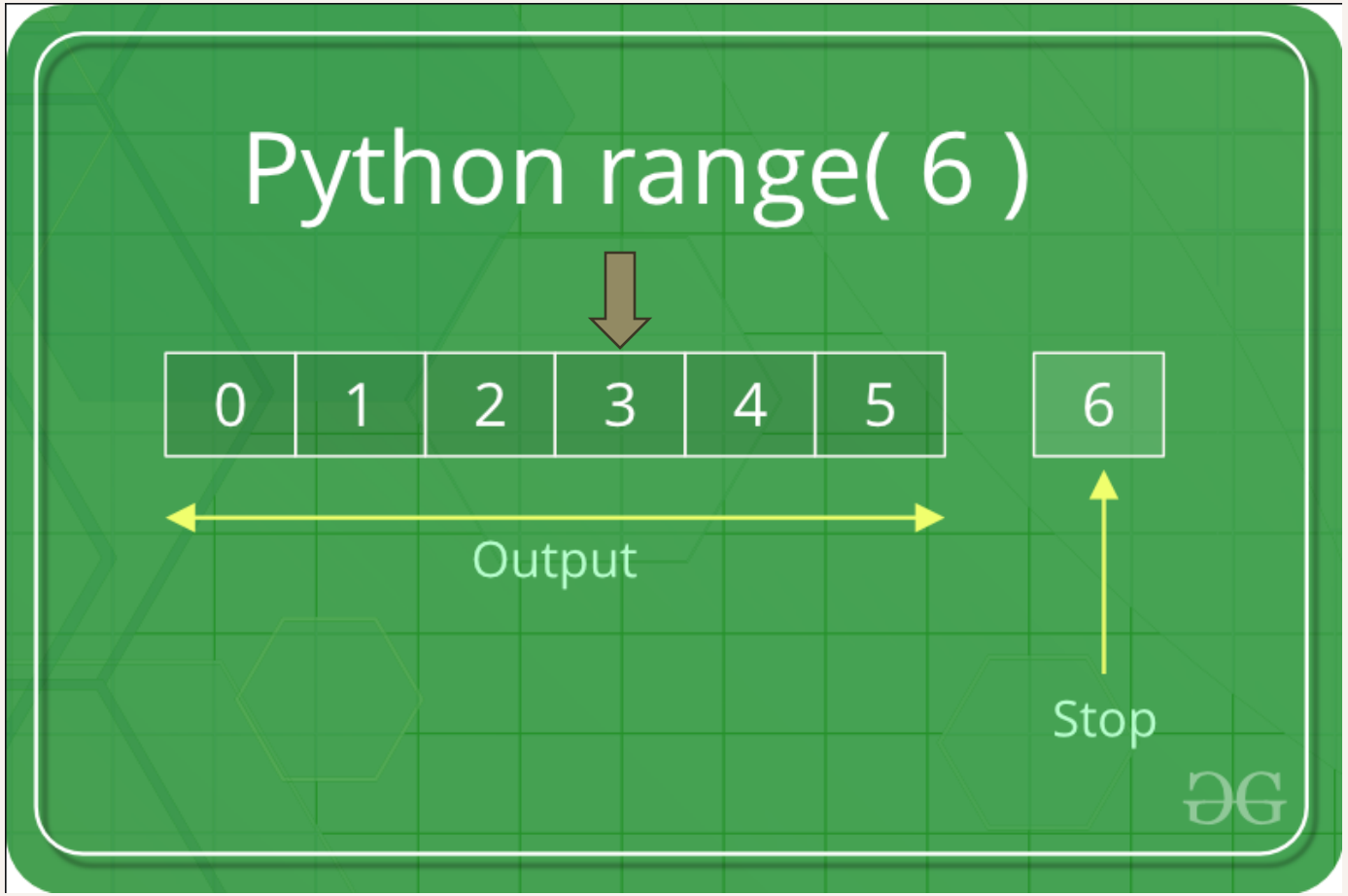
Andre iterasjon: i = 1



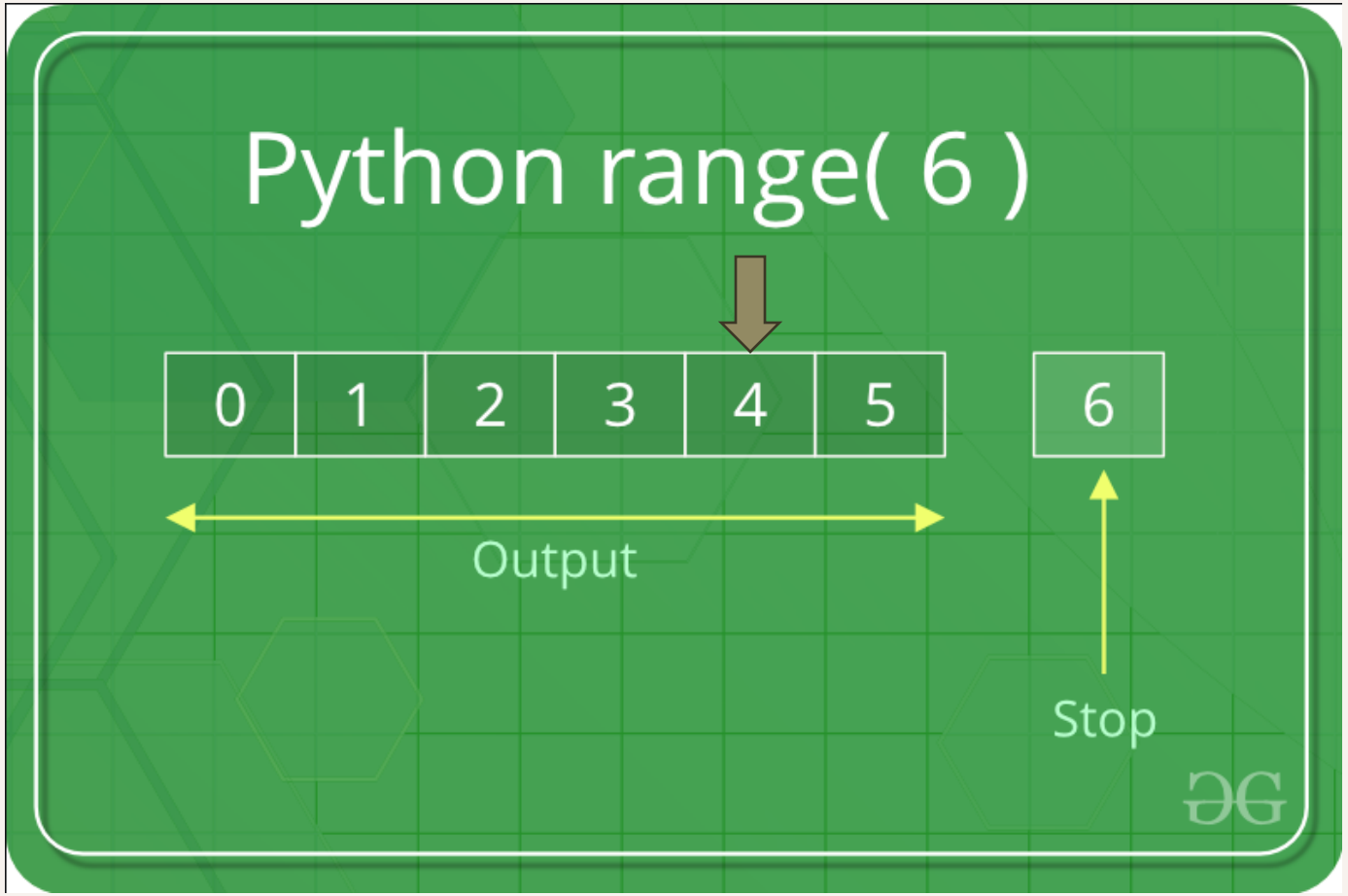
Tredje iterasjon: i = 2



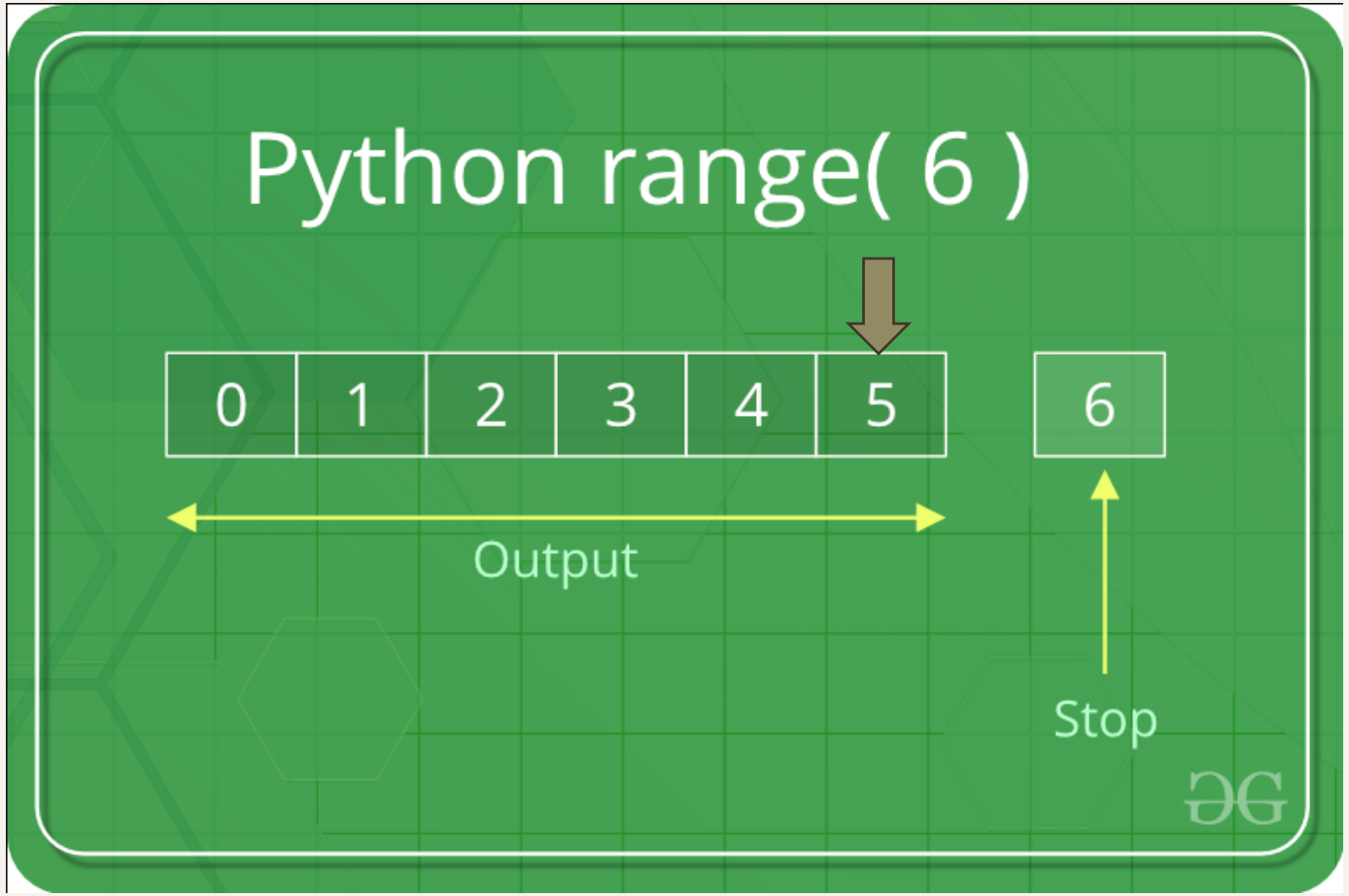
Fjerde iterasjon: i = 3



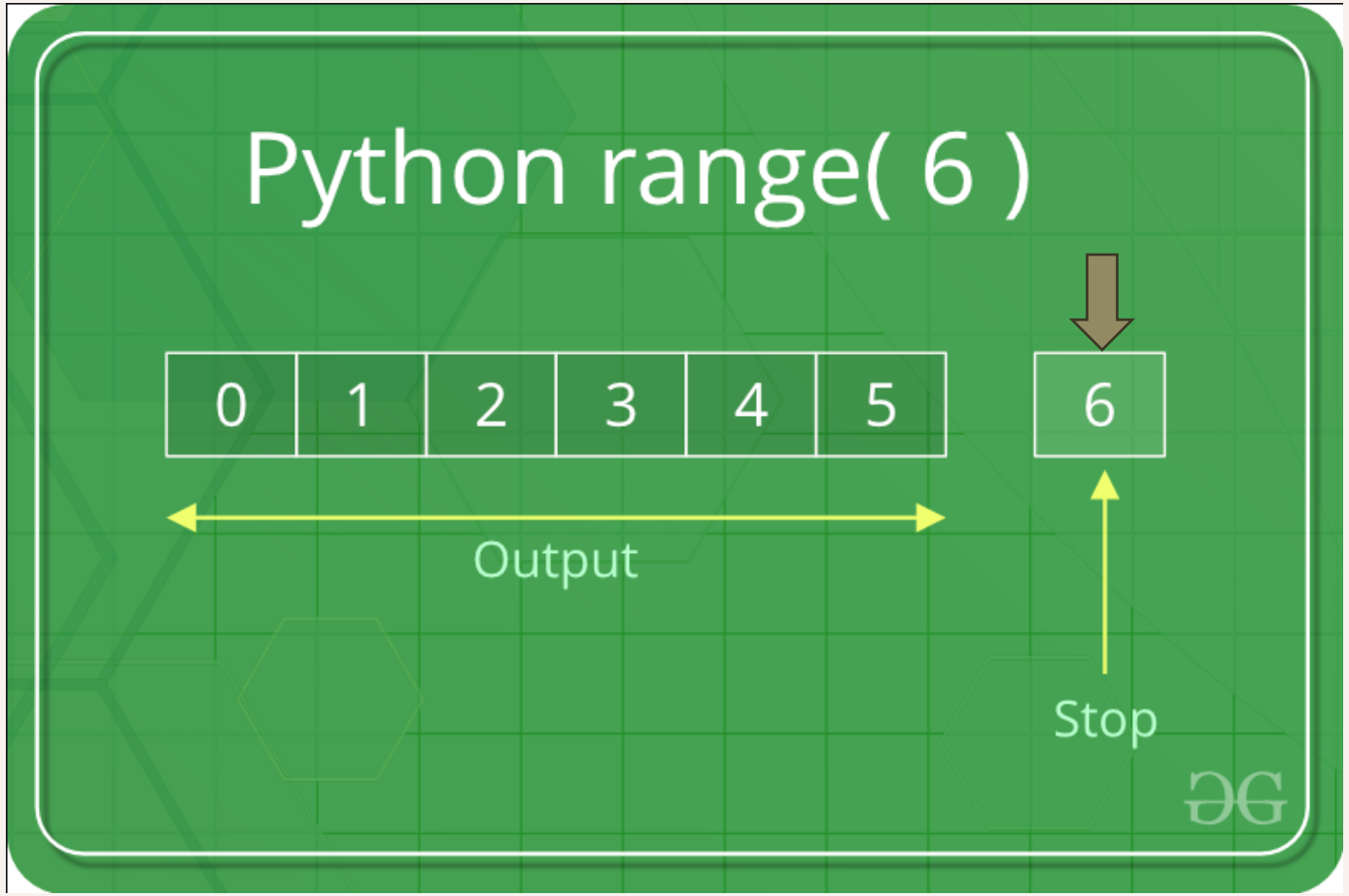
Femte iterasjon: i = 4



Sjette iterasjon: i = 5

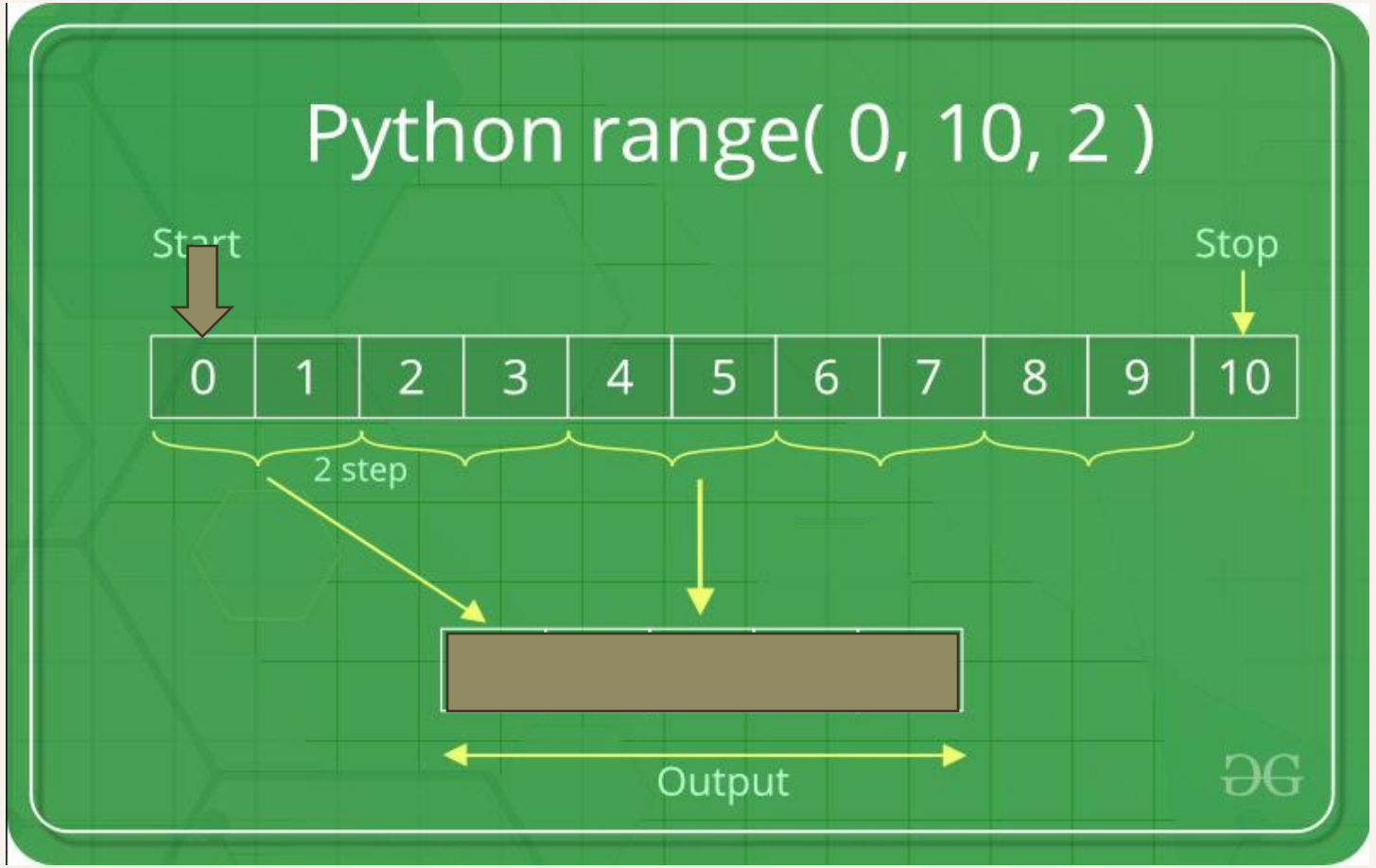


i >= 6 - Stopp!

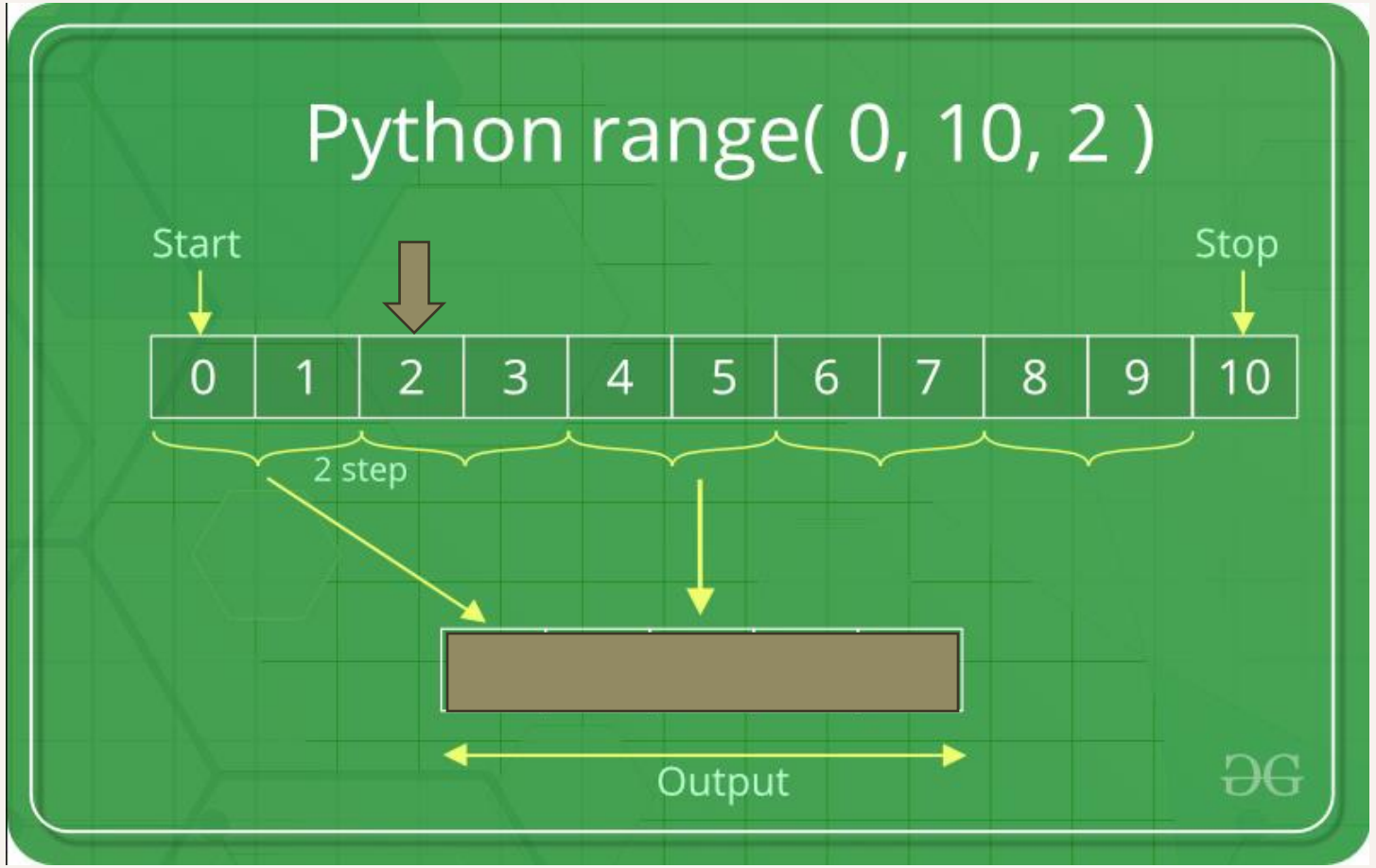


Illustrasjon av for-løkke med startverdi, sluttverdi og økning

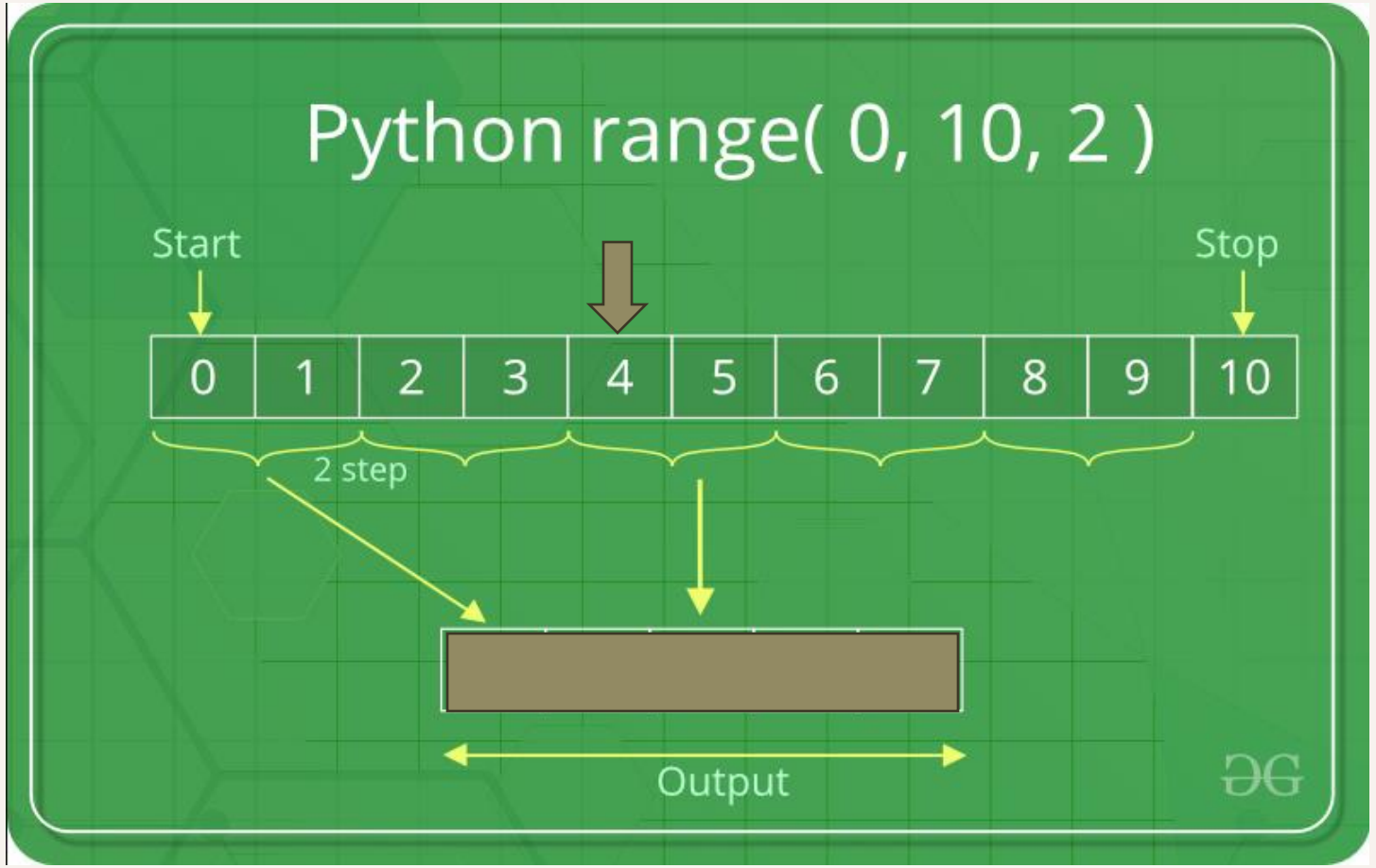
Første iterasjon: i = 0



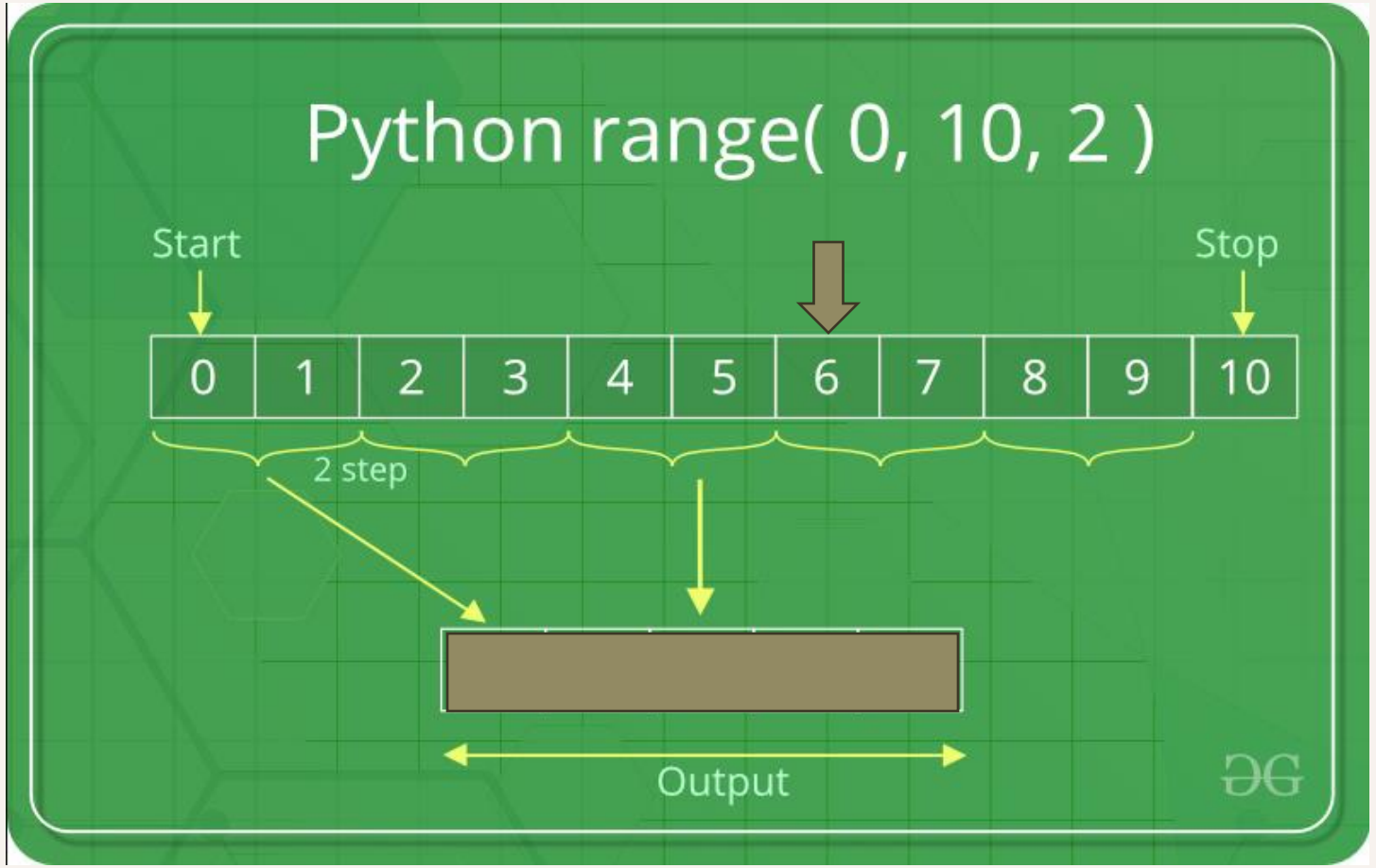
Andre iterasjon: i = 2



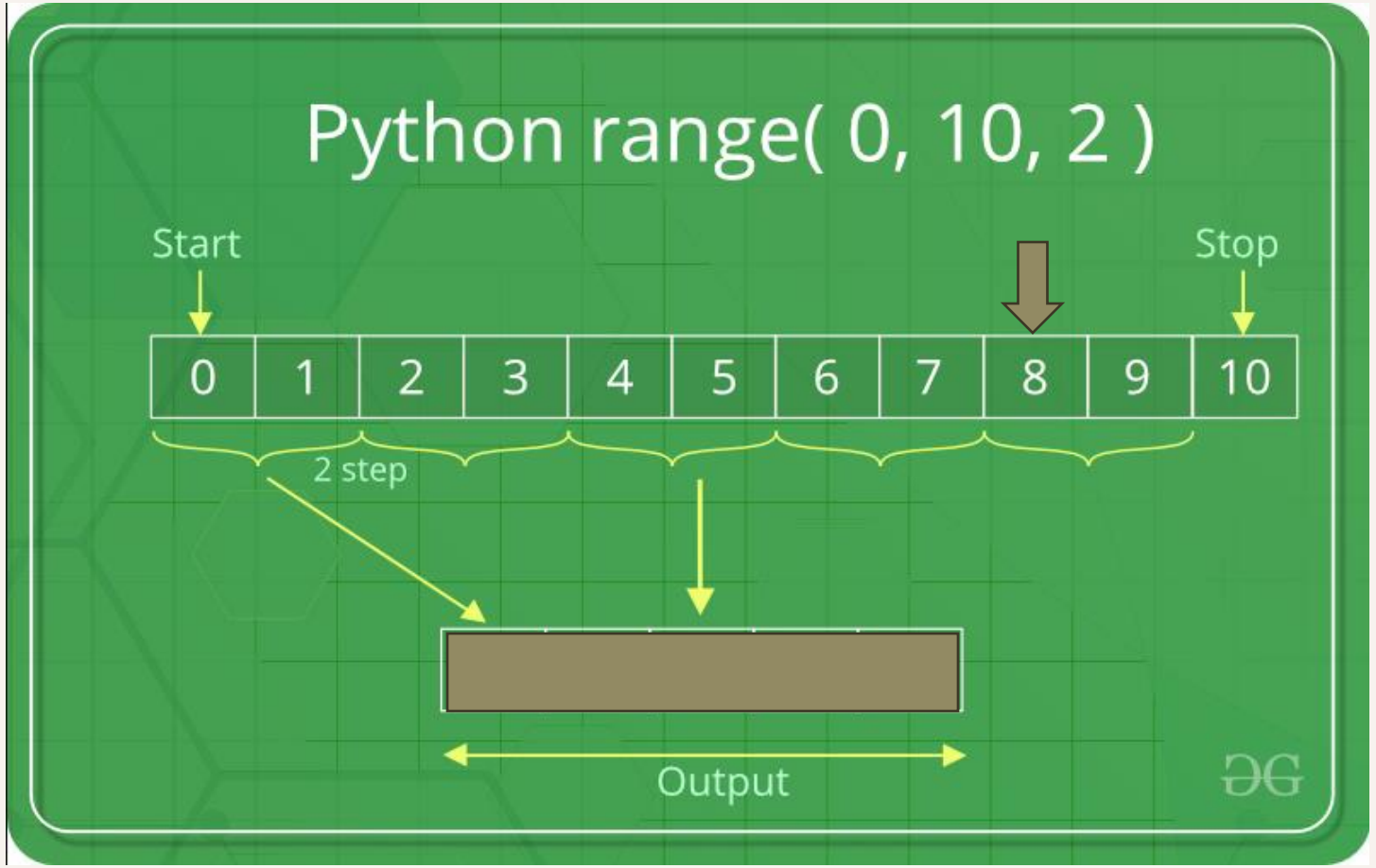
Tredje iterasjon: i = 4



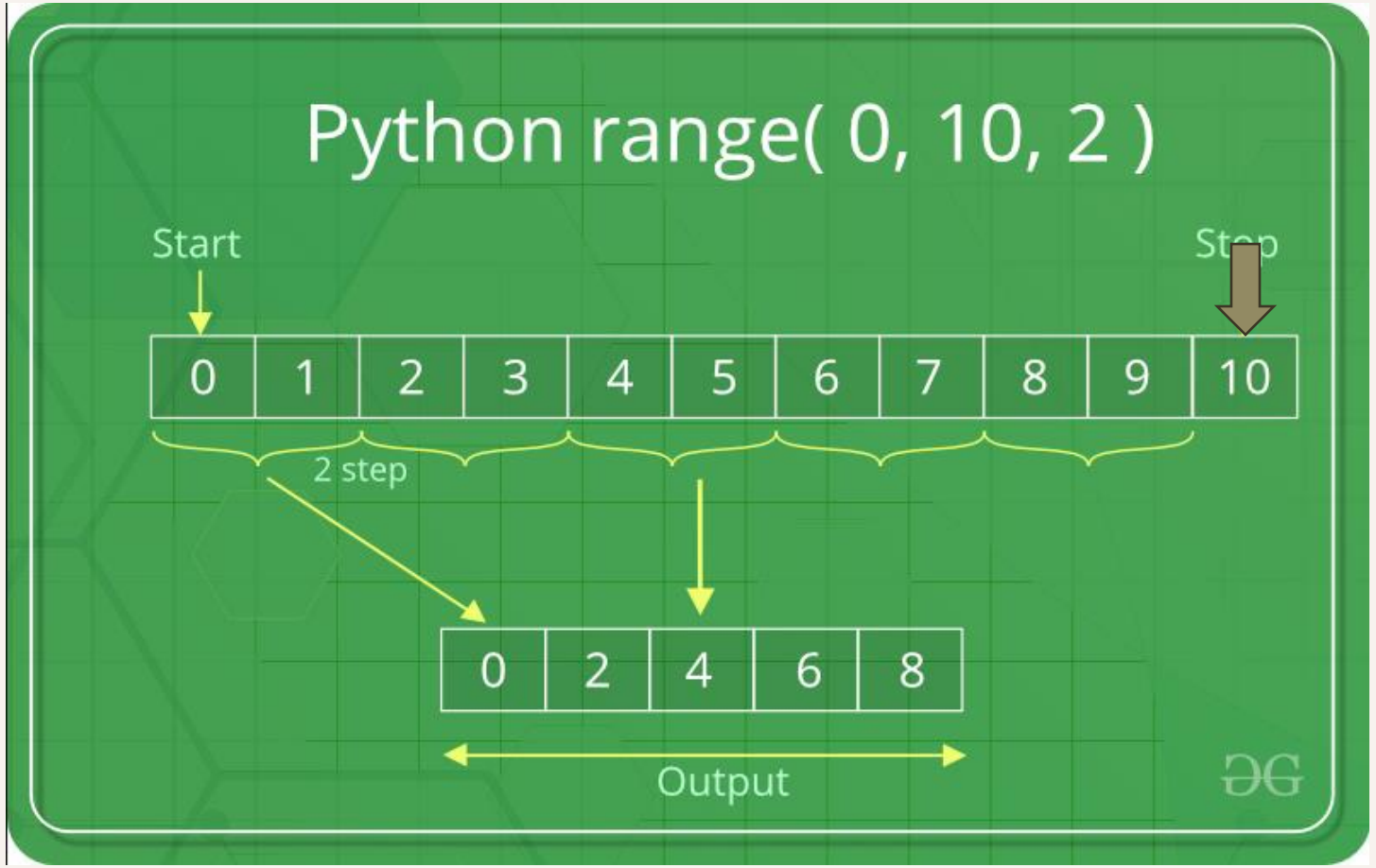
Fjerde iterasjon: i = 6



Femte iterasjon: i = 8



i>= 10 – Stopp!


Python range(0, 10, 2)
Start
Stop
0
1
2
3
4
5
6
7
8
9
10
2 step
0
2
4
6
8
Output

Prosedyrer uten parametere - repetisjon

Prosedyrer kort forklart

- En gjenbrukbar kodeblokk som vi kan kalle på hvor som helst i koden etter prosedyren er definert
- Nyttig hvis vi ønsker å gjøre det samme flere ganger uten å skrive samme kode på nytt
- Syntaks:

```
def funksjonsnavn() :
        <det prosedyren skal gjore>
```

Kodeflyten i en prosedyre

```
1. def si_hei():
3.     print("Hei")
4.     print("Dette er en prosedyre")


2. si_hei()
```

Prosedyrer med parametere

Hva er parametere?

- Informasjon som kan tilpasse prosedyren til å gjøre forskjellige ting
- Skrives som variabler inni parentesene etter prosedyrenavnet der prosedyren blir definert
- Kan kalle disse variablene for hva du vil, akkurat som med andre variabler
- Syntaks:

```
def prosedyre_navn(parameter1, parameter2):
    # gjør noe
```

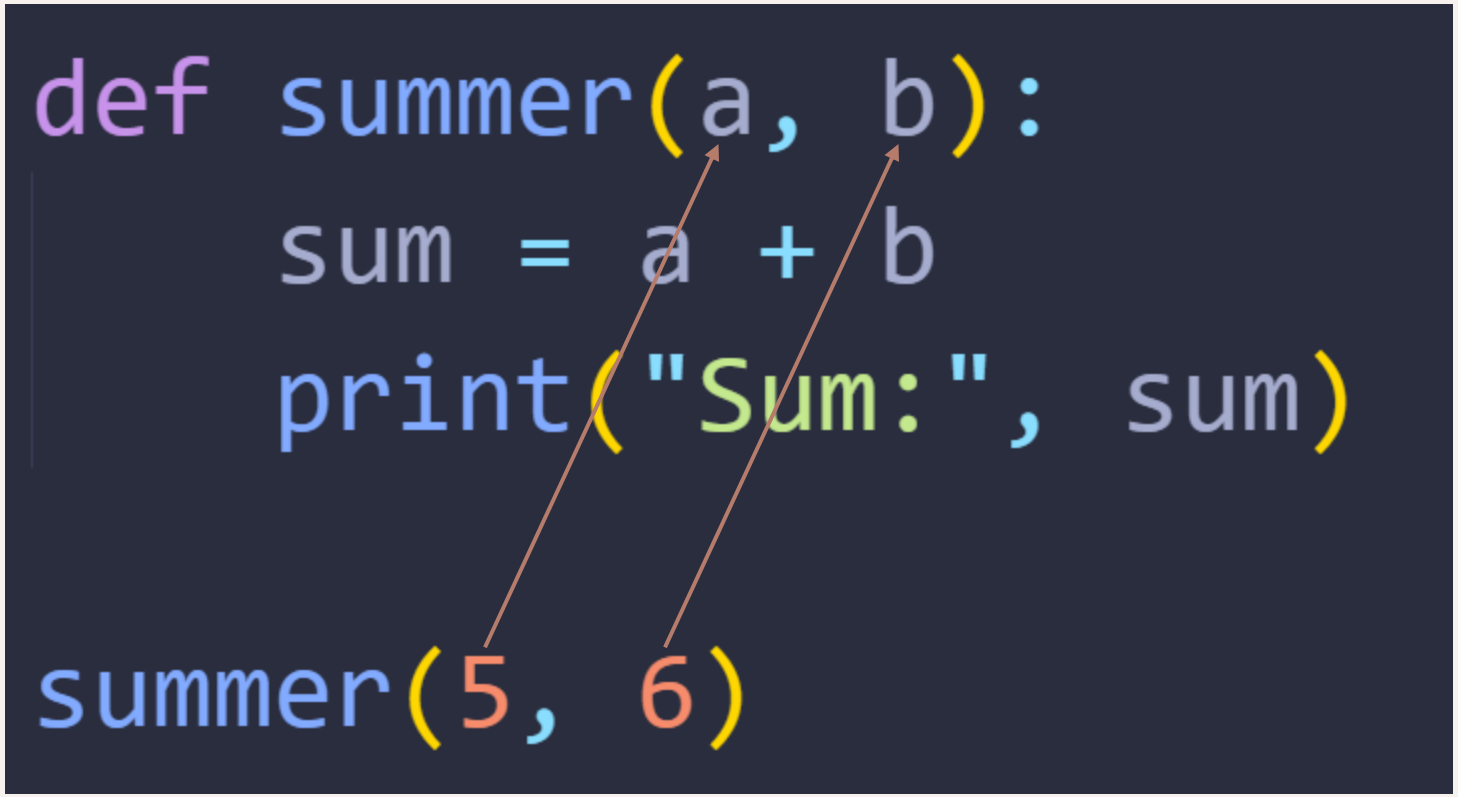
Prosedyre med parametere eksempel

```
def summer(a, b):
    sum = a + b
    print("Sum:", sum)

summer(5, 6)
```

Prosedyre med parametere eksempel

- Hva denne metoden printer er basert på hva som skrives inn i parentesene når metoden kalles
- Verdiene som står i parentes når prosedyren kalles heter **argumenter**, variablene som står i parentes når prosedyren defineres kalles **parametere**



Prosedyre med parametere eksempel

```
def summer(a, b):
    sum = a + b
    print("Sum:", sum)


summer(5, 6)
```

a og b er parametere til prosedyren, og blir til verdien satt av argumentene

5 og 6 er argumenter til prosedyren

Funksjoner

Forskjellen på prosedyrer og funksjoner

PROSEDYRER	FUNKSJONER

• Har **ingen** returverdi
`def min_prosedyre(ord):`
`  print("Ordet er", ord)`
	• Har **alltid** en returverdi
	`def min_funksjon(ord):`
	`  return "Ordet er", ord`

Hva betyr å returnere noe?

- Noen ganger ønsker vi å returnere noe istedenfor å skrive ut noe
- Grunnen til dette er ofte at vi ønsker å bruke det vi returnerer senere i programmet
- Når vi returnerer noe i Python bruker man *return*
- Returverdien er verdien vi returnerer fra funksjonen

Hva er en funksjon?

- Funksjoner er som prosedyrer men med returverdier
- Man kan bare returnere en ting
- Dersom man ønsker å returnere mer enn en ting må man lage en liste og returnere listen
- Som regel når man kaller på en funksjon tar man vare på det den returnerer i en variabel
- Syntaks:

```
def funksjons_navn(parameter1, parameter2):
    # gjør noe
    return # sett inn returverdi her
```

Eksempel på en funksjon

Hva får vi hvis vi skriver print(verdier_fra_summer)?

```
def summer(a, b):
    sum = a + b
    return sum # returnerer verdien til sum

verdi_fra_summer = summer(2, 3)
```

Eksempel på en funksjon

Hva får vi hvis vi skriver print(verdier_fra_summer)?

Svar:
5

```
def summer(a, b):
    sum = a + b
    return sum # returnerer verdien til sum

verdi_fra_summer = summer(2, 3)
```

Oppgave

Oppgave 1

Skriv en funksjon som tar i mot to tall og returnerer det største av de to tallene (ikke skrive det ut).

Løsning

- Den alternative løsningen funker fordi funksjonen avsluttes etter den har returnert noe

Løsning

```
def storst_av_to(tall1, tall2) :
        if tall1 > tall2 :
                return tall1
        else :
                return tall2
```

Alternativ:

```
def storst_av_to(tall1, tall2) :
        if tall1 > tall2 :
                return tall1
        return tall2
```

Lese fra filer

Filer i Python

- Med filer kan vi jobbe med større input og større mengder data enn når vi får input fra tastatur – vi slipper å skrive inn input i terminalen og får denne inputen heller fra en fil
  - Betyr også at vi kan lage mer spennende programmer
- En fil består av linjer og hver linje kan være delt opp videre med for eksempel mellomrom eller komma

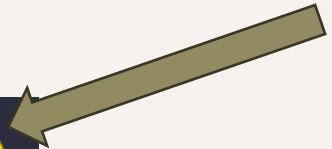
Hvordan åpne en fil

- Før vi kan skrive til en fil må vi åpne filen
- Bruker biblioteksfunksjonen **open** for å åpne filer
- Syntaks:

```
innfil = open("filnavn.filtype", "r")
```

- Eksempel – gitt en fil som heter input.txt:

```
innfil = open("input.txt", "r")
```



"r" står for read, altså vi vil lese fra denne filen

Hvordan lese fra fil

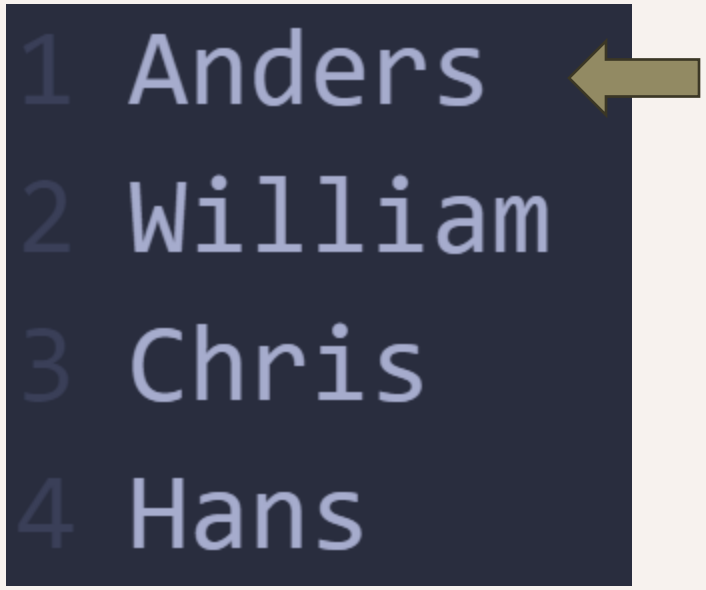
- En måte å lese innholdet i en fil er å gå gjennom hver linje med en for-løkke, slik:

```
fil = open("filnavn.txt", "r")
for linje in fil:
    # gjør noe med linje
```

Hvordan går denne løkken gjennom filen?

linje = Anders

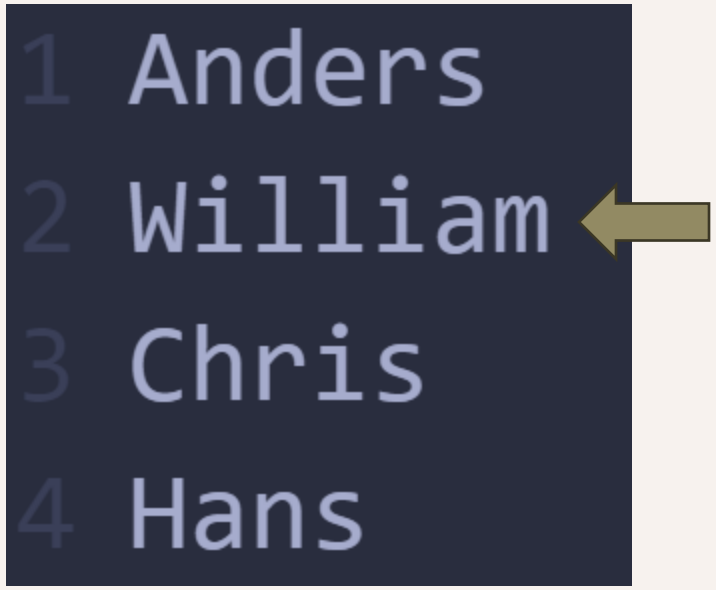
filnavn.txt



Hvordan går denne løkken gjennom filen?

linje = William

filnavn.txt



Hvordan går denne løkken gjennom filen?

linje = Chris

filnavn.txt



Hvordan går denne løkken gjennom filen?

linje = Hans

filnavn.txt



Hvordan lese fra fil

```
fil = open("filnavn.txt", "r")
for linje in fil:
    print(linje)
```

Output:
Anders
William
Chris
Hans

En linje kan bestå av flere verdier

- En linje kan være delt opp med flere verdier
- Da kan vi lagre hver av disse verdiene i en liste
- For å gjøre dette bruker vi split()
- Split() tar *seperatortegnet* som argument – for eksempel komma eller mellomrom

Eksempel på hvordan man kan lese fra en fil med split()

```
fil = open("filnavn.txt", "r")
for linje in fil:
    biter = linje.split(",")
    navn = biter[0]
    alder = biter[1]
    print(navn)
    print(alder)
```

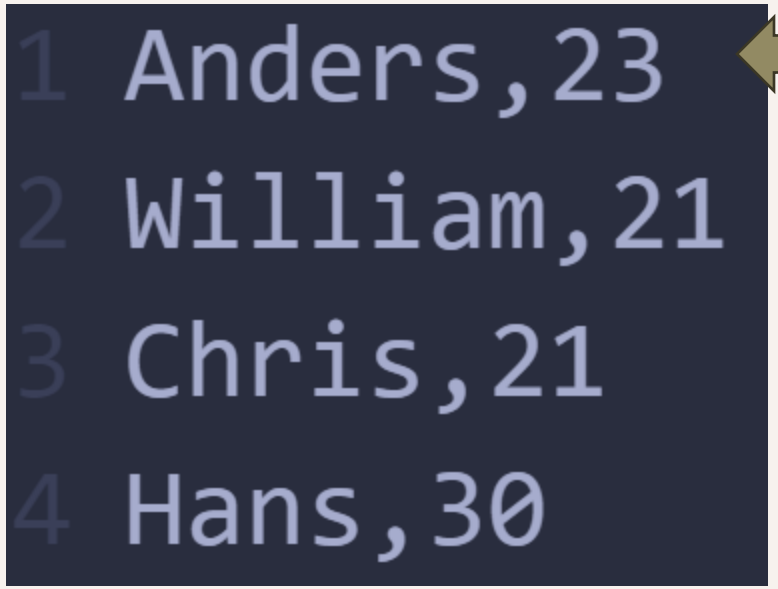
Hvordan går denne løkken gjennom filen?

biter = [Anders, 23]

biter[0] = Anders

biter[1] = 23

filnavn.txt




```
fil = open("filnavn.txt", "r")
for linje in fil:
    biter = linje.split(",")
    navn = biter[0]
    alder = biter[1]
    print(navn)
    print(alder)
```

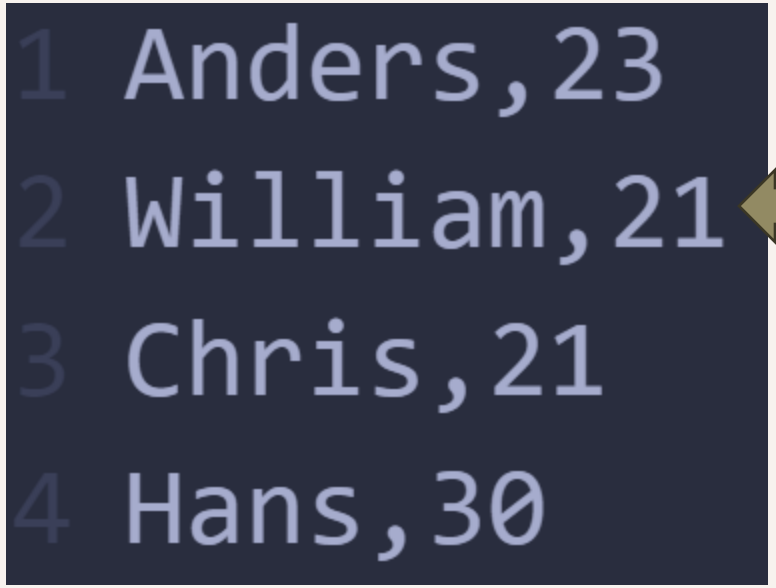
Hvordan går denne løkken gjennom filen?

biter = [William, 21]

biter[0] = William

biter[1] = 21

filnavn.txt




```
fil = open("filnavn.txt", "r")
for linje in fil:
    biter = linje.split(",")
    navn = biter[0]
    alder = biter[1]
    print(navn)
    print(alder)
```

Hvordan går denne løkken gjennom filen?

biter = [Chris, 21]

biter[0] = Chris

biter[1] = 21

filnavn.txt

```
1 Anders,23
2 William,21
3 Chris,21
4 Hans,30
```



```
fil = open("filnavn.txt", "r")
for linje in fil:
    biter = linje.split(",")
    navn = biter[0]
    alder = biter[1]
    print(navn)
    print(alder)
```

Hvordan går denne løkken gjennom filen?

biter = [Hans, 30]

biter[0] = Hans

biter[1] = 30

filnavn.txt

```
1 Anders,23
2 William,21
3 Chris,21
4 Hans,30
```



```
fil = open("filnavn.txt", "r")
for linje in fil:
    biter = linje.split(",")
    navn = biter[0]
    alder = biter[1]
    print(navn)
    print(alder)
```

Skrive til filer

Hvordan åpne en fil som du skal skrive til

- Bruker "w" for write istedenfor "r"
- Ellers er det helt likt
- Syntaks:

```
utfil = open("filnavn.filtype", "w")
```

Hvordan skrive til fil

- Bruker write for å skrive tekst til en fil
- Syntaks:
  - filnavn.write("Denne teksten blir skrevet til filen filnavn\n")
- Man setter inn \n på slutten for å få hver string man skriver på en ny linje
- \n = newline
- Write tar bare strings som argument, så hvis du vil for eksempel skrive inn et tall til en fil må du gjøre dette om til en string først

Skrive til fil eksempel

```
fil = open("nyFil.txt","w")
for x in range(1, 10):
    fil.write(f"Dette blir linje nr {x}\n")
```

nyFil vil da se sånn ut er løkka er ferdig med å kjøre

```
1 Dette blir linje nr 1
2 Dette blir linje nr 2
3 Dette blir linje nr 3
4 Dette blir linje nr 4
5 Dette blir linje nr 5
6 Dette blir linje nr 6
7 Dette blir linje nr 7
8 Dette blir linje nr 8
9 Dette blir linje nr 9
```

Lukk filen med .close()

- God kodeskikk å lukke filen når du er ferdige med å lese eller skrive til den
- Syntaks:
  - filnavn.close()

```
fil = open("filnavn.txt", "r")
for linje in fil:
    print(linje)

fil.close()
```

Livekode

HVORDAN LESE INN DATA FRA FIL OG LAGRE DETTE I EN ORDBOK + FUNKSJON SOM FINNER BILLETTPRIS BASERT PÅ GITT ARGUMENT

Oppgave

Oppgave 2

Anta at du har filen “historie.txt”. Les inn alle linjene og lagre dem i en liste. Første linje skal ligge først i listen, osv. (`Tips`: husk å åpne og lukke filen..)

Løsning

Løsning

```
innfil = open("historie.txt", "r")
historie_liste = []
for linje in innfil:
    historie_liste.append(linje)

innfil.close()
```

Oppgave

Oppgave 3

Skriv en funksjon som tar i mot to sannhetsverdier (True/False) og returnerer 1 hvis begge verdier er sanne, 0 hvis begge verdier ikke er sanne.

Gruble:
Hva tar ikke funksjonen høyde for? Hva kan gå "galt"?

Løsning

Løsning

```
def is_true(a, b) :
 if a and b:
    return 1
 if not a and not b:
    return 0
 return -1
```

Funksjonen tar ikke høyde for tilfellet hvor den ene sannhetsverdien er sann og den andre er usann. Kan legge inn en egen verdi for dette, feks. returnere -1.

Jobbe med innlevering/Trix

- Ukens pensum:
  - Trix oppgave 05.01 - 05.26
    - https://trix.ifi.uio.no/course/8?tags=uke5
  - Innlevering 5, evt. innlevering 3 og 4 for de som ikke allerede har levert

