

# Outline

- Løkker
- Kombinere løkker og samlinger
- For-løkker
- **Prosedyrer med parametre**
- Funksjoner (returverdier)

# Flere typer subrutiner

- **Subrutine** (fra uke 2): en **navngitt blokk** med kodelinjer, som kan **kalles** og **tilpasses**
- Vi vil i dag introdusere flere aspekter
  - Uke 2: Prosedyre - **uten** parametre og returverdi
  - I dag: Prosedyre - med **parametre**
  - I dag: Funksjon - med **returverdi**
  - Om tre uker: Instans-**metode** (OO)

# Prosedyrer med parametre

- Prosedyren vi så på i tidligere uke gjorde alltid eksakt det samme når den ble kallet
  - Det er sjelden av nytte!
- For å være nyttig må en slik prosedyre kunne **tilpasses**
  - Det gjør vi ved å sende inn **parametre**

# Din første prosedyre med parametre

- **print** er en prosedyre hvor utfallet tilpasses!
- `print(text)`:  
skriver verdien i `text` til skjermen†
  - Variabelen `text` er en **parameter**
  - Verdien vi gir inn (f.eks. "hallo IN1000") når vi kaller `print` er et **argument**
- Parameter og argument er to sider av det samme
  - Parameter: **variabel** i prosedyre som tar i mot verdi
  - Argument: **verdi** sendt inn når prosedyren kalles

# Prosedyre med parametre

```
def mittProsedyreNavn(parameter1, parameter2, ...):  
    kode1linje1  
    kode1linje2  
    ...
```


For å kjøre alle kodelinjene i prosedyren ("*kalle*  
prosedyren"):

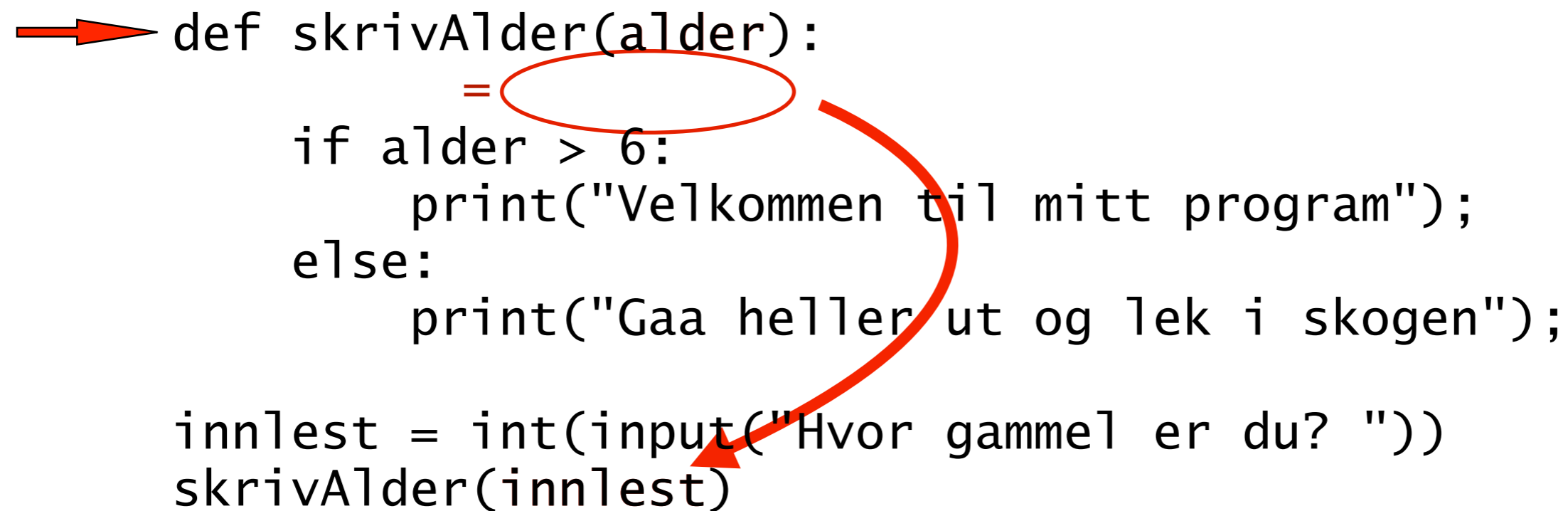
```
mittProsedyreNavn(argument1, argument2, ...)
```

# Prosedyre med parametre

- {prosedyre\_med\_parameter\_v1.py-  
prosedyre\_med\_parameter\_v3.py}

# Parameteren tilordnes verdien av argumentet!

```
→ def skrivAlder(alder):  
    =   
    if alder > 6:  
        print("Velkommen til mitt program");  
    else:  
        print("Gaa heller ut og lek i skogen");  
  
innlest = int(input("Hvor gammel er du? "))  
skrivAlder(innlest)
```



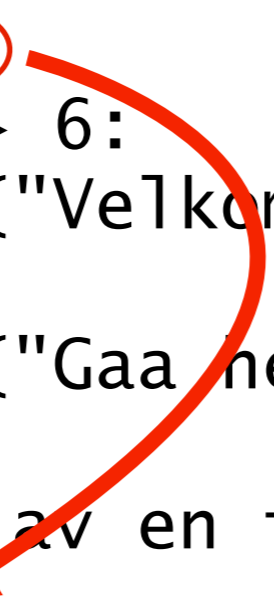
# Parameteren tilordnes verdien av argumentet!

```
def skrivAlder(alder):  
    if alder > 6:  
        print("Velkommen til mitt program");  
    else:  
        print("Gaa heller ut og lek i skogen");  
  
print("Hacket av en toaaring: ")  
skrivAlder(2)
```



# Parameteren tilordnes verdien av argumentet!

```
def skrivAlder(alder):  
    alder = 2  
    if alder > 6:  
        print("Velkommen til mitt program");  
    else:  
        print("Gaa heller ut og lek i skogen");  
  
print("Hacket av en toaaring: ")  
skrivAlder(2)
```



# Oppgave:

*Hva skrives ut på skjermen?*

```
def pros1(a):  
    print(a*2)
```

```
def pros2(b):  
    print(b)  
    pros1(b+2)
```

```
pros1(5)  
pros2(4)
```

# Løsning:


*Hva skrives ut på skjermen?*

```
def pros1(a):  
    print(a*2)
```

```
def pros2(b):  
    print(b)  
    pros1(b+2)
```

```
pros1(5)  
pros2(4)
```

På skjermen:



# Løsning:

*Hva skrives ut på skjermen?*

```
def pros1(a):  
    a=5  
    print(a*2) #10
```

```
def pros2(b):  
    print(b)  
    pros1(b+2)
```

```
pros1(5)  
pros2(4)
```

På skjermen:



10

# Løsning:

*Hva skrives ut på skjermen?*

```
def pros1(a):  
    print(a*2)
```

```
def pros2(b):  
    b=4  
    print(b)  
    pros1(b+2) #6
```

```
pros1(5)  
pros2(4)
```

På skjermen:



10  
4

# Løsning:

*Hva skrives ut på skjermen?*

```
def pros1(a):  
    a=6  
    print(a*2) #12
```

```
def pros2(b):  
    b=4  
    print(b)  
    pros1(b+2) #6
```

```
pros1(5)  
pros2(4)
```

På skjermen:

```
10  
4  
12
```

# Prosedyrer outsourcer detaljer og håndterer redundans

- Man har ofte behov for (omtrent) samme funksjonalitet ulike steder i en kode
- Man ønsker da ikke å duplisere koden
  - Minsker oversiktighet av kode
  - Endringer og rettinger må utføres mange steder
- Man samler i stedet funksjonaliteten i en prosedyren og kaller metoden der den trengs
  - Dersom det er noe variasjon i hva man trenger, representerer man det som varierer med en parameter