Funksjoner, prosedyrer og skop

...

Hva er en prosedyre og hvorfor er det lurt å bruke ?

- Strukturer kode i blokker
- Hvordan lage en prosedyrer?
    a. Start med prosedyre definisjonen, du må selv gi den et navn (i eksemplet nedenfor er det minProsedyre. Du kan legge til så mange parametere som du ønsker (i eksempelet nedenfor har vi lagd to parameter1 og parameter2

```
2
3  def minProsedyre(parameter1, parameter2):
4
```

Hva er en prosedyre og hvorfor er det lurt å bruke ?

- Strukturer kode i blokker
- Hvordan lage en prosedyrer?
  a. Start med prosedyre definisjonen, du må selv gi den et navn (i eksemplet nedenfor er det minProsedyre. Du kan legge til så mange parametere som du ønsker (i eksempelet nedenfor har vi lagd to parameter1 og parameter2
  b. Videre legger vi til den kode blokken vi ønsker at skal være inne i prosedyren (i dette eksempelet slår vi sammen de to parametrene, før vi legger til parameter1 en gang til bakerst og printer det ut)

```
2
3  def minProsedyre(parameter1, parameter2):
4      tekst = parameter1 + parameter2
5      tekst += paramter1
6      print(tekst)
```

Ordforklaringer:

- Parameter: variabel i prosedyren som tar i mot en verdi

Prosedyrer

- Strukturer kode i blokker
- Hvordan lage en prosedyrer?
  a. Start med prosedyre definisjonen, du må selv gi den et navn (i eksemplet nedenfor er det minProsedyre. Du kan legge til så mange parametere som du ønsker (i eksempelet nedenfor har vi lagd to parameter1 og parameter2
  b. Videre legger vi til den kode blokken vi ønsker at skal være inne i prosedyren (i dette eksempelet slår vi sammen de to parametrene, før vi legger til parameter1 en gang til bakerst og printer det ut)
  c. Vi må så kalle på prosedyren, (kodeblokken inne i) prosedyren vil kjøre når den blir kalt på :) Husk å gi et argument per parameter

```
2
3  def minProsedyre(parameter1, parameter2):
4      tekst = parameter1 + parameter2
5      tekst += paramter1
6      print(tekst)
7
8  minProsedyre("argument1", "argument2")
```

Ordforklaringer:

- Parameter: variabel i prosedyren som tar i mot en verdi
- Argument: verdi sendt inn når prosedyren kalles

Fordeler med prosedyrer

- Hjelper oss å strukturere koden
- Hvis vi har relativ lik kode flere steder burde dette lages til en prosedyre slik at vi kan kalle på den, istedenfor å ha duplikater i koden

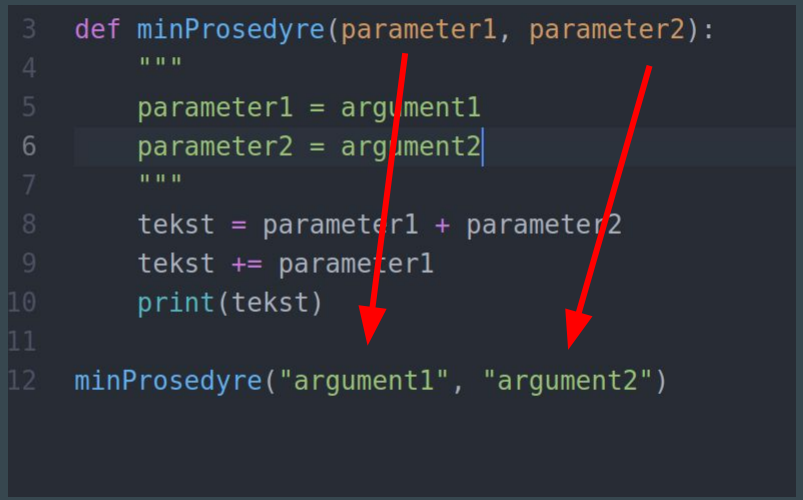
Parameteroverføring på 1,2,3

- Fordeler med parameter er at vi kan ha tilpasninger i prosedyren

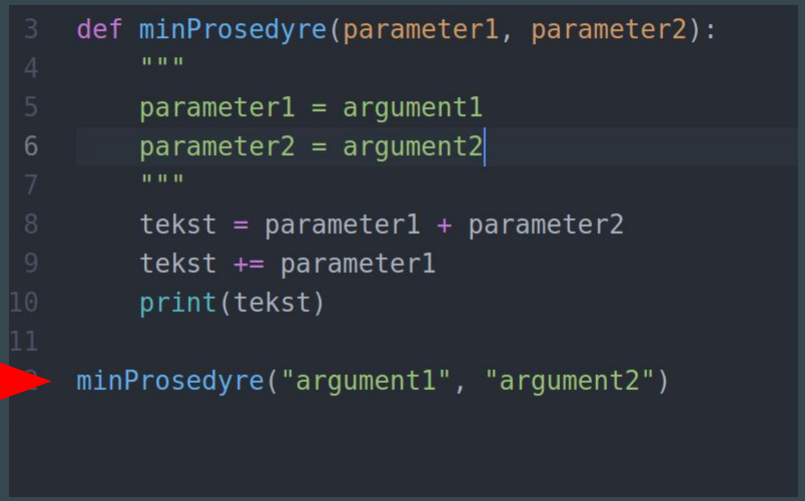
```
3  def minProsedyre(parameter1, parameter2):
4      """
5      parameter1 = argument1
6      parameter2 = argument2
7      """
8      tekst = parameter1 + parameter2
9      tekst += parameter1
10     print(tekst)
11
12 minProsedyre("argument1", "argument2")
```

Parameteroverføring på 1,2,3

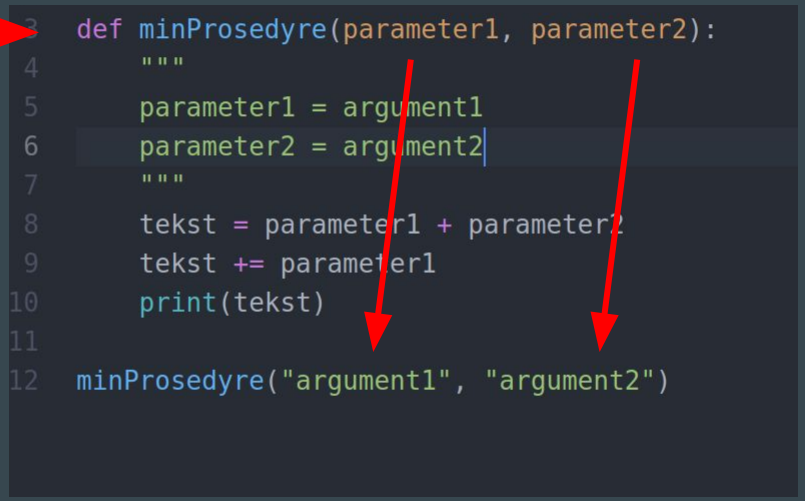
- Fordeler med parameter er at vi kan ha tilpasninger i prosedyren



Kodeflyt


3 def minProsedyre(parameter1, parameter2):
4 """
5 parameter1 = argument1
6 parameter2 = argument2
7 """
8 tekst = parameter1 + parameter2
9 tekst += parameter1
10 print(tekst)
11
minProsedyre("argument1", "argument2")

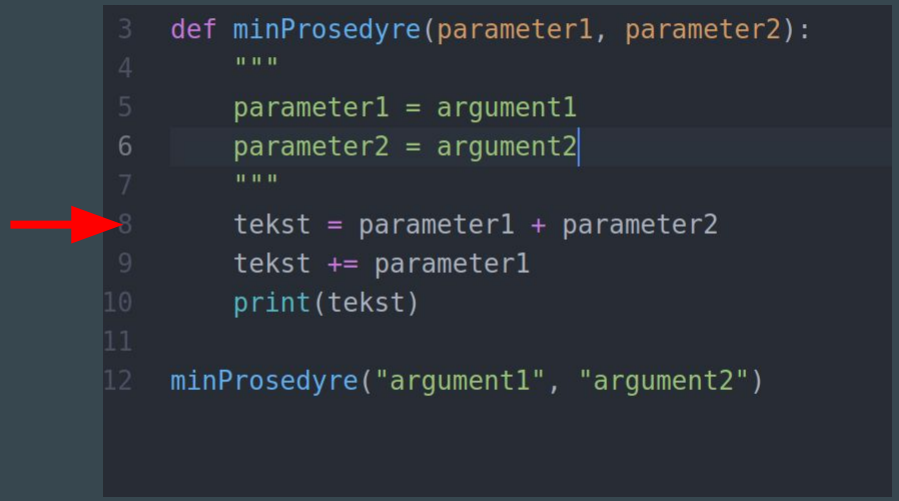
Kodeflyt



Variabler i minProsedyre

parameter1	“argument1”
parameter2	“argument2”

Kodeflyt



```
3  def minProsedyre(parameter1, parameter2):
4      """
5      parameter1 = argument1
6      parameter2 = argument2
7      """
8      tekst = parameter1 + parameter2
9      tekst += parameter1
10     print(tekst)
11
12 minProsedyre("argument1", "argument2")
```

Variabler i minProsedyre

parameter1	“argument1”
parameter2	“argument2”
tekst	“argument1argument2”

Kodeflyt

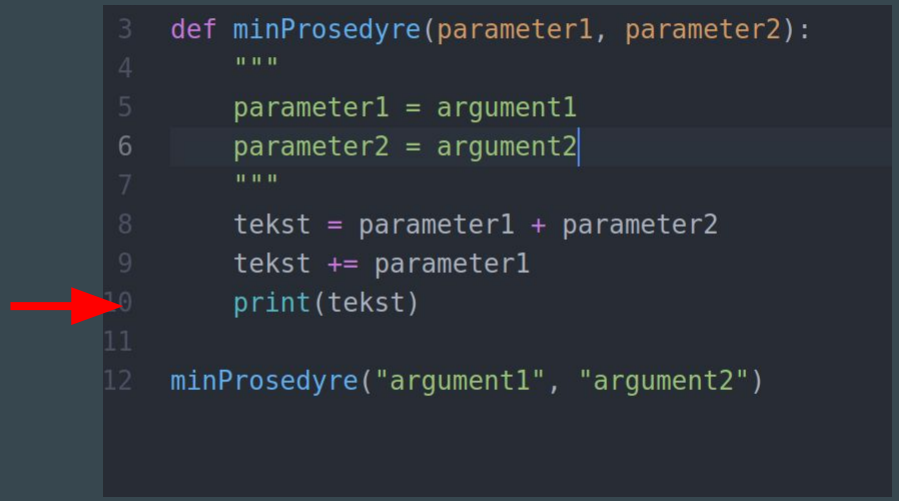
```
3  def minProsedyre(parameter1, parameter2):
4      """
5      parameter1 = argument1
6      parameter2 = argument2
7      """
8      tekst = parameter1 + parameter2
9      tekst += parameter1
10     print(tekst)
11
12 minProsedyre("argument1", "argument2")
```



Variabler i minProsedyre

parameter1	“argument1”
parameter2	“argument2”
tekst	“argument1argument2argument1”

Kodeflyt



```
3  def minProsedyre(parameter1, parameter2):
4      """
5      parameter1 = argument1
6      parameter2 = argument2
7      """
8      tekst = parameter1 + parameter2
9      tekst += parameter1
10     print(tekst)
11
12 minProsedyre("argument1", "argument2")
```

Variabler i minProsedyre

parameter1	“argument1”
parameter2	“argument2”
tekst	“argument1argument2argument1”

På terminalen:

“argument1argument2argument1”

Hva er en funksjon ?

- En funksjon lar deg “outsource” en beregning til en separat kodeblokk
- Forskjellen på en funksjon og en prosedyre er altså at en funksjon returnerer noe det gjør ikke en prosedyre
- Defineres på samme måte som en prosedyre bortsett fra at at vi må ha return med

```
3  def storstTall(tall1, tall2):
4      if tall1 > tall2
5          return tall1
6      return tall2
```

Kodeflyt

```
3  def storstTall(tall1, tall2):
4      if tall1 > tall2
5          return tall1
6      return tall2
7
8  storstEn = storstTall(1, 2)
9  print("Det største tallet av 1 og 2 er ", storstEn)
10
11 storstTo = storstTall(6, 2)
12 print("Det største tallet av 6 og 2 er ", storstTo)
```



Kodeflyt

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

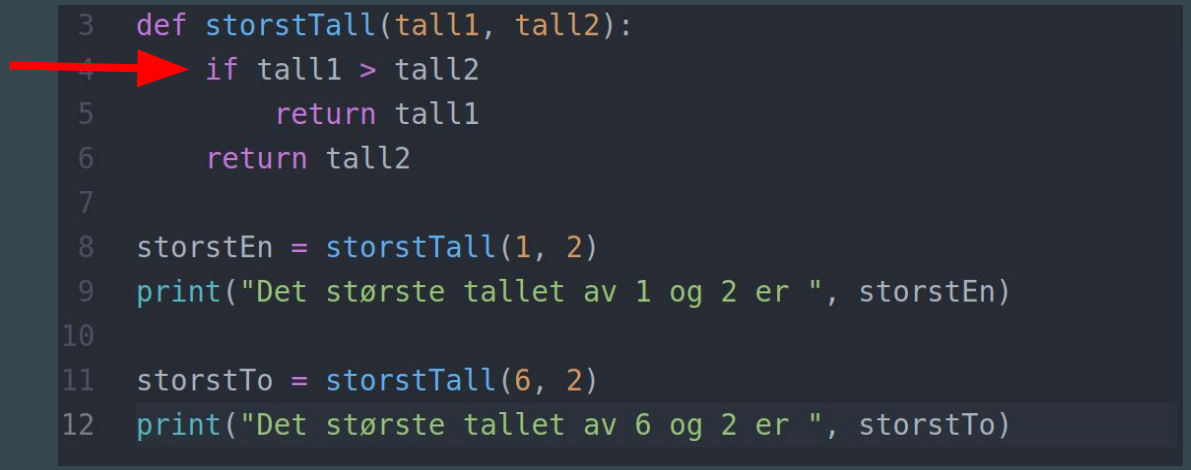
storstEn = storstTall(1, 2)
print("Det største tallet av 1 og 2 er ", storstEn)

storstTo = storstTall(6, 2)
print("Det største tallet av 6 og 2 er ", storstTo)
```

Variabler i storstTall

tall1	1
tall2	2

Kodeflyt



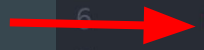
```
3  def storstTall(tall1, tall2):
4      if tall1 > tall2
5          return tall1
6      return tall2
7
8  storstEn = storstTall(1, 2)
9  print("Det største tallet av 1 og 2 er ", storstEn)
10
11 storstTo = storstTall(6, 2)
12 print("Det største tallet av 6 og 2 er ", storstTo)
```

Variabler i storstTall

tall1	1
tall2	2

Kodeflyt

```
3  def storstTall(tall1, tall2):
4      if tall1 > tall2
5          return tall1
6      return tall2
7
8  storstEn = storstTall(1, 2)
9  print("Det største tallet av 1 og 2 er ", storstEn)
10
11 storstTo = storstTall(6, 2)
12 print("Det største tallet av 6 og 2 er ", storstTo)
```



Variabler i storstTall

tall1	1
tall2	2

Kodeflyt

```
3  def storstTall(tall1, tall2):
4      if tall1 > tall2
5          return tall1
6      return tall2
7
8  storstEn = storstTall(1, 2)
9  print("Det største tallet av 1 og 2 er ", storstEn)
10
11 storstTo = storstTall(6, 2)
12 print("Det største tallet av 6 og 2 er ", storstTo)
```

2

Variabler i det globale skop

storstEn	2

Kodeflyt

```
3  def storstTall(tall1, tall2):
4      if tall1 > tall2
5          return tall1
6      return tall2
7
8  storstEn = storstTall(1, 2)
9  print("Det største tallet av 1 og 2 er ", storstEn)
10
11 storstTo = storstTall(6, 2)
12 print("Det største tallet av 6 og 2 er ", storstTo)
```

2

På terminalen:

Det største tallet av 1 og 2 er 2

Variabler i det globale skop

storstEn	2

Kodeflyt

```
3  def storstTall(tall1, tall2):
4      if tall1 > tall2
5          return tall1
6      return tall2
7
8  storstEn = storstTall(1, 2)
9  print("Det største tallet av 1 og 2 er ", storstEn)
10
11 storstTo = storstTall(6, 2)
12 print("Det største tallet av 6 og 2 er ", storstTo)
```

2

Variabler i det globale skop

storstEn	2

På terminalen:

Det største tallet av 1 og 2 er 2

Kodeflyt

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

storstEn = storstTall(1, 2)
print("Det største tallet av 1 og 2 er ", storstEn)

storstTo = storstTall(6, 2)
print("Det største tallet av 6 og 2 er ", storstTo)
```

2

På terminalen:

Det største tallet av 1 og 2 er 2

Variabler i storstTall

tall1	6
tall2	2

Variabler i det globale skop

storstEn	2

Kodeflyt

```
3  def storstTall(tall1, tall2):
4      if tall1 > tall2
5          return tall1
6      return tall2
7
8  storstEn = storstTall(1, 2)
9  print("Det største tallet av 1 og 2 er ", storstEn)
10
11 storstTo = storstTall(6, 2)
12 print("Det største tallet av 6 og 2 er ", storstTo)
```

2

På terminalen:

Det største tallet av 1 og 2 er 2

Variabler i storstTall

tall1	6
tall2	2

Variabler i det globale skop

storstEn	2

Kodeflyt

```
3  def storstTall(tall1, tall2):
4      if tall1 > tall2
5          return tall1
6      return tall2
7
8  storstEn = storstTall(1, 2)
9  print("Det største tallet av 1 og 2 er ", storstEn)
10
11 storstTo = storstTall(6, 2)
12 print("Det største tallet av 6 og 2 er ", storstTo)
```

2

På terminalen:

Det største tallet av 1 og 2 er 2

Variabler i storstTall

tall1	6
tall2	2

Variabler i det globale skop

storstEn	2

Kodeflyt

```
3  def storstTall(tall1, tall2):
4      if tall1 > tall2
5          return tall1
6      return tall2
7
8  storstEn = storstTall(1, 2)
9  print("Det største tallet av 1 og 2 er ", storstEn)
10
11 storstTo = storstTall(6, 2)
12 print("Det største tallet av 6 og 2 er ", storstTo)
```

2

6

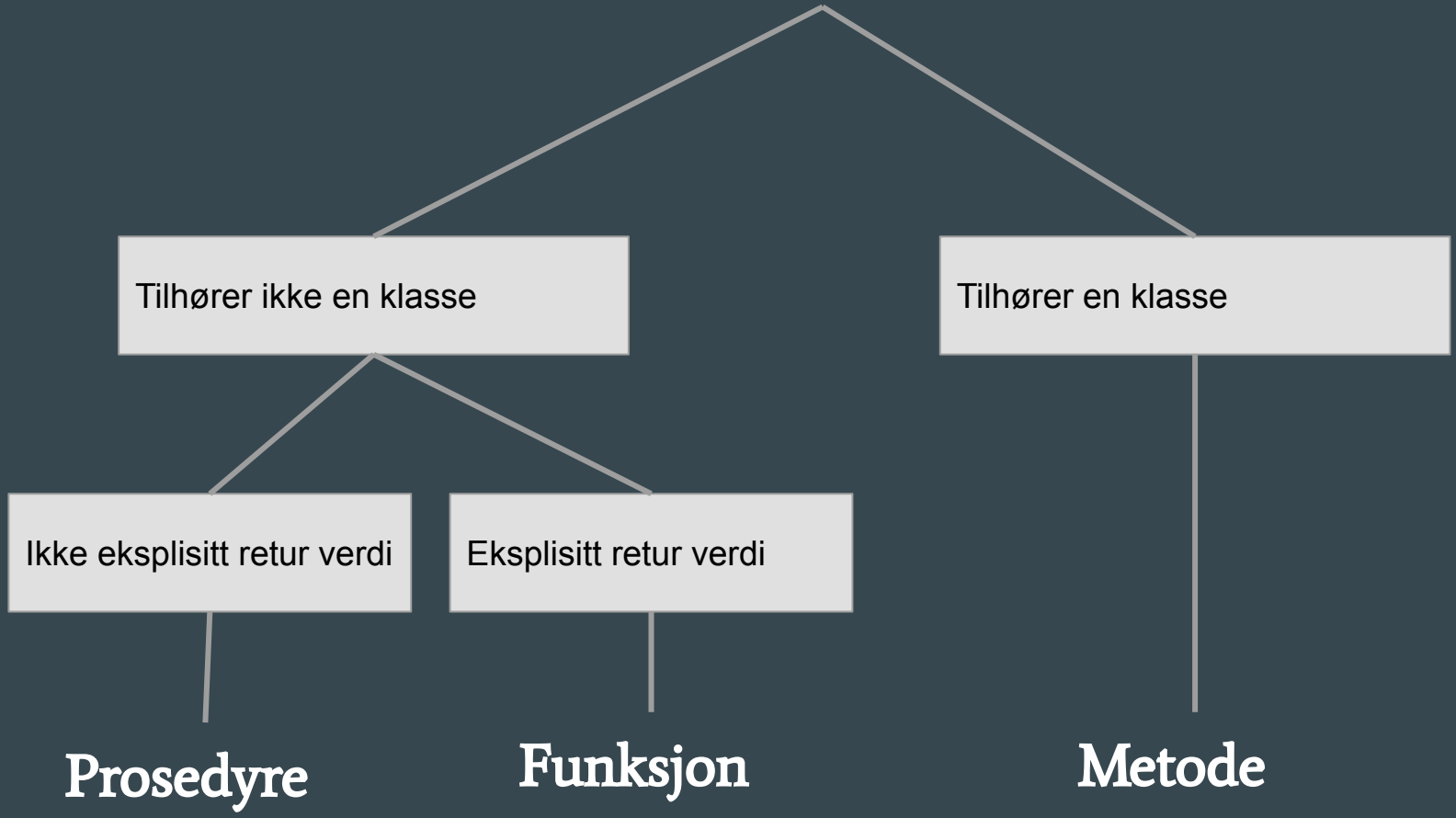
På terminalen:

Det største tallet av 1 og 2 er 2

Det største tallet av 6 og 2 er 6

Variabler i det globale skop

storstEn	2
storstTo	6

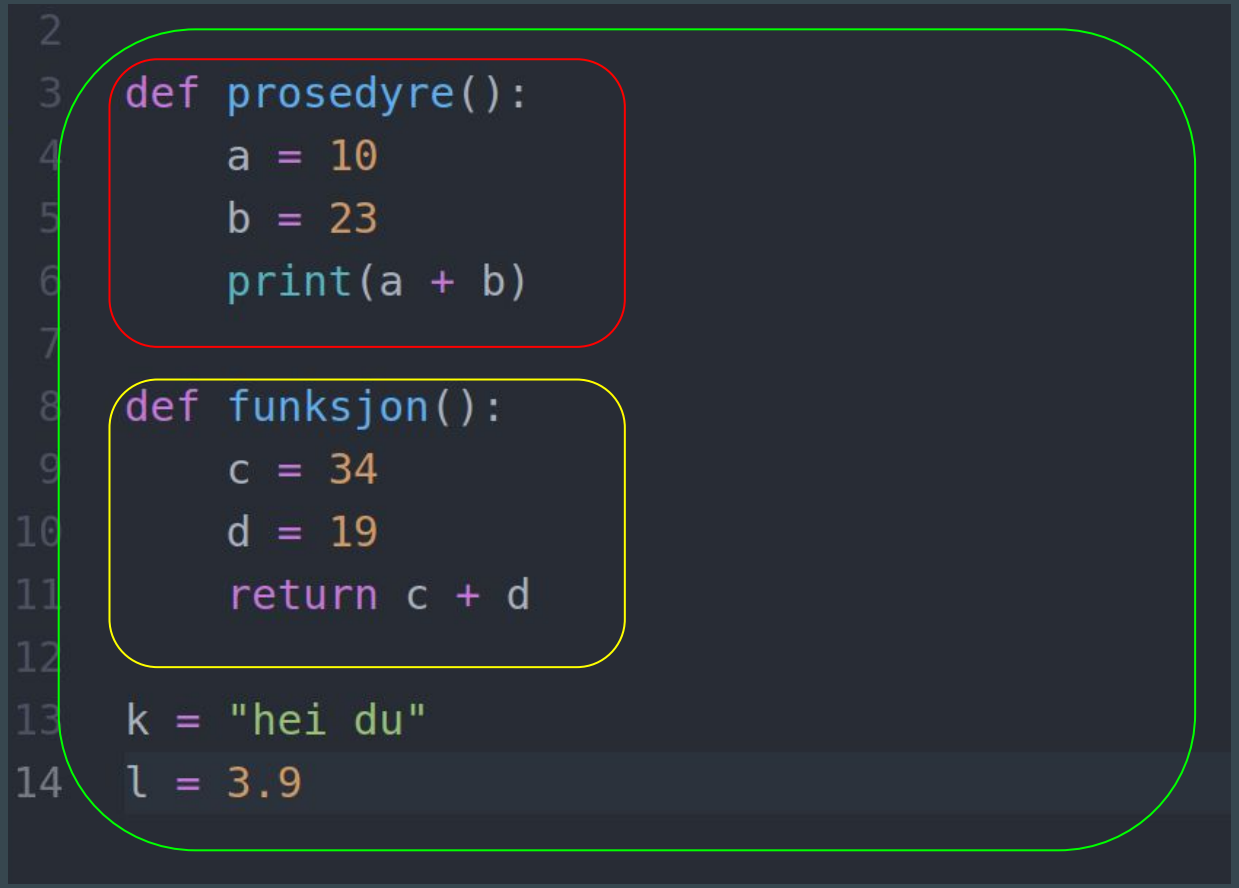

Tilhører ikke en klasse
Tilhører en klasse
Ikke eksplisitt retur verdi
Eksplisitt retur verdi
Prosedyre
Funksjon
Metode

Funksjon	En funksjon som defineres med *def*, som ikke er del av en klasse (ordet *self* brukes ikke). Funksjoner har alltid en returverdi. Eks: *def sum(a, b):* *c = a + b* *return c* På engelsk kalles dette *function*.
Prosedyre	Tilsvarende funksjon, men uten returverdi. Bruker aldri ordet *self*, og er ikke del av en klasse. Eks: *def superprint(ord):* *print("ordet er", ord)* På engelsk kalles dette *procedure*.
Metode	Tilsvarende funksjon, men som del av en klasse. Har alltid *self* som første parameter. Kan, men må ikke, ha en returverdi. Eks: *def areal(self):* *firkant_areal = self.lengde * self.bredde* *return firkant_areal* På engelsk kalles dette *method*.

Skop

Kort sagt har alle tilgang på variablene som ligger i det globale skopet, mens det er en variabel inne i en prosedyre/funksjon er det kunne denne prosedyren/funksjonen, som har tilgang og kan bruke denne variabelen

Skop


def prosedyre():
a = 10
b = 23
print(a + b)
def funksjon():
c = 34
d = 19
return c + d
k = "hei du"
l = 3.9

Skop

prosedyre kan bruke variablene: a,b,k,l

funksjon kan bruke variablene: c,d,k,l

det globale skopet kan kun bruke variablene: k og l

