Uge 2

Tirsdag

Dagens tekst

- Variabler
- Control flow
- Funktioner

Variabler, assignment og sandt/falsk test

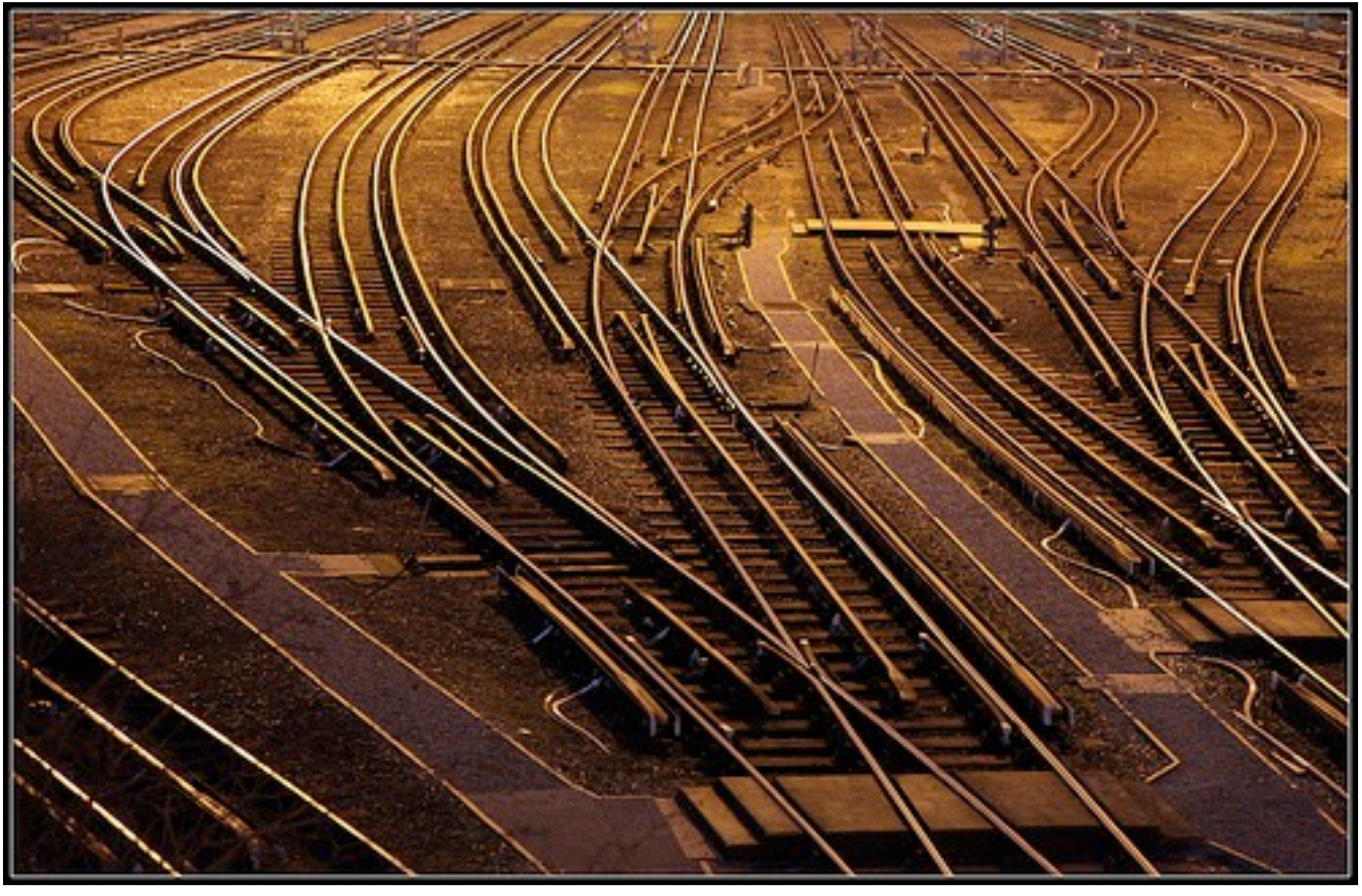
```
x = 3

y = x + 21

z = None

if x > 10:
    print x
```

Hvad er et program?

En serie af simple udtryk/instruktioner evalueret et ad gangen fra start til slut.

Hvilken serie der evalueres kontrolleres dynamisk af sand/falsk tests - “Control flow”

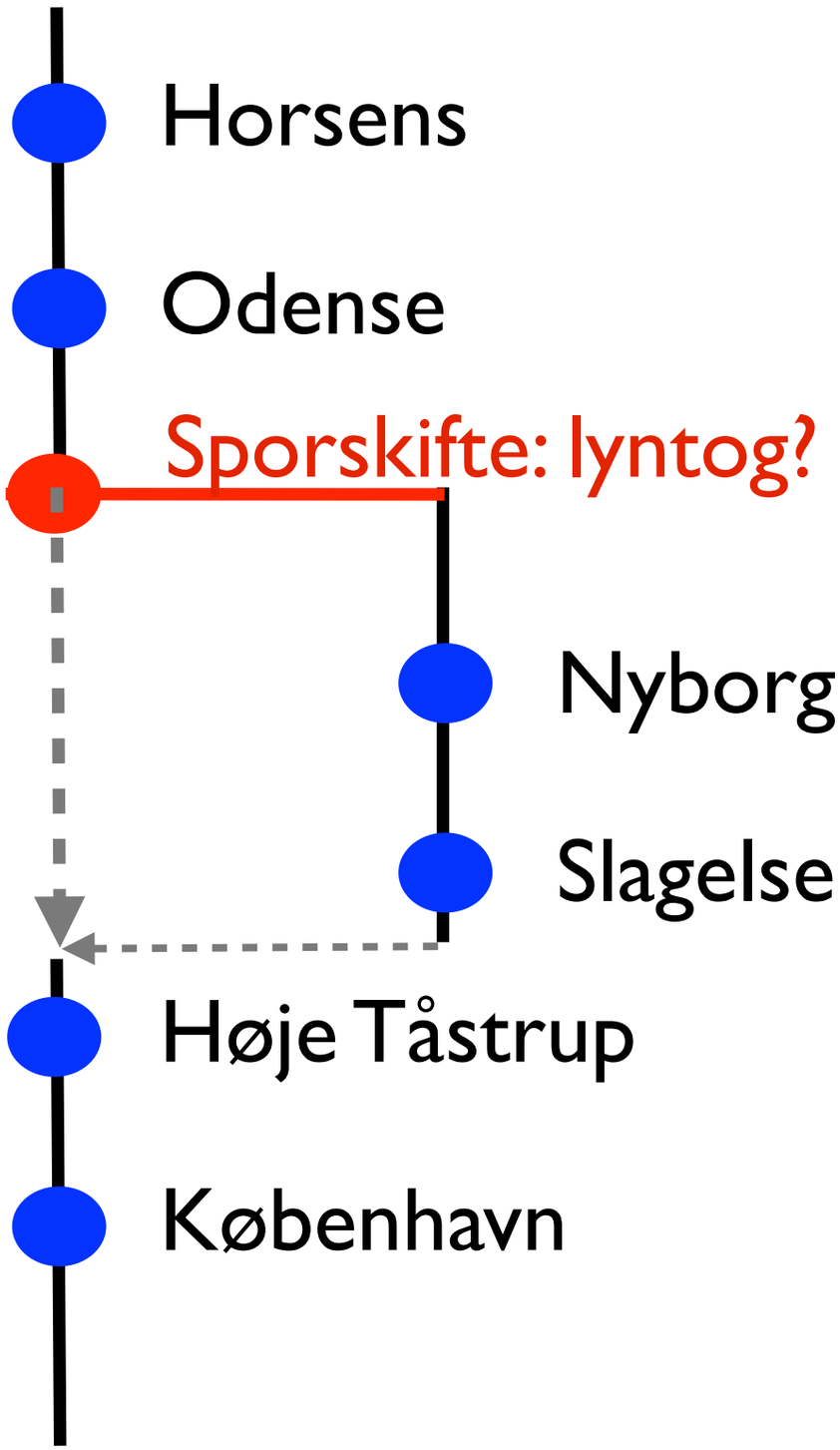
Brug din editor ikke den interaktive prompt!



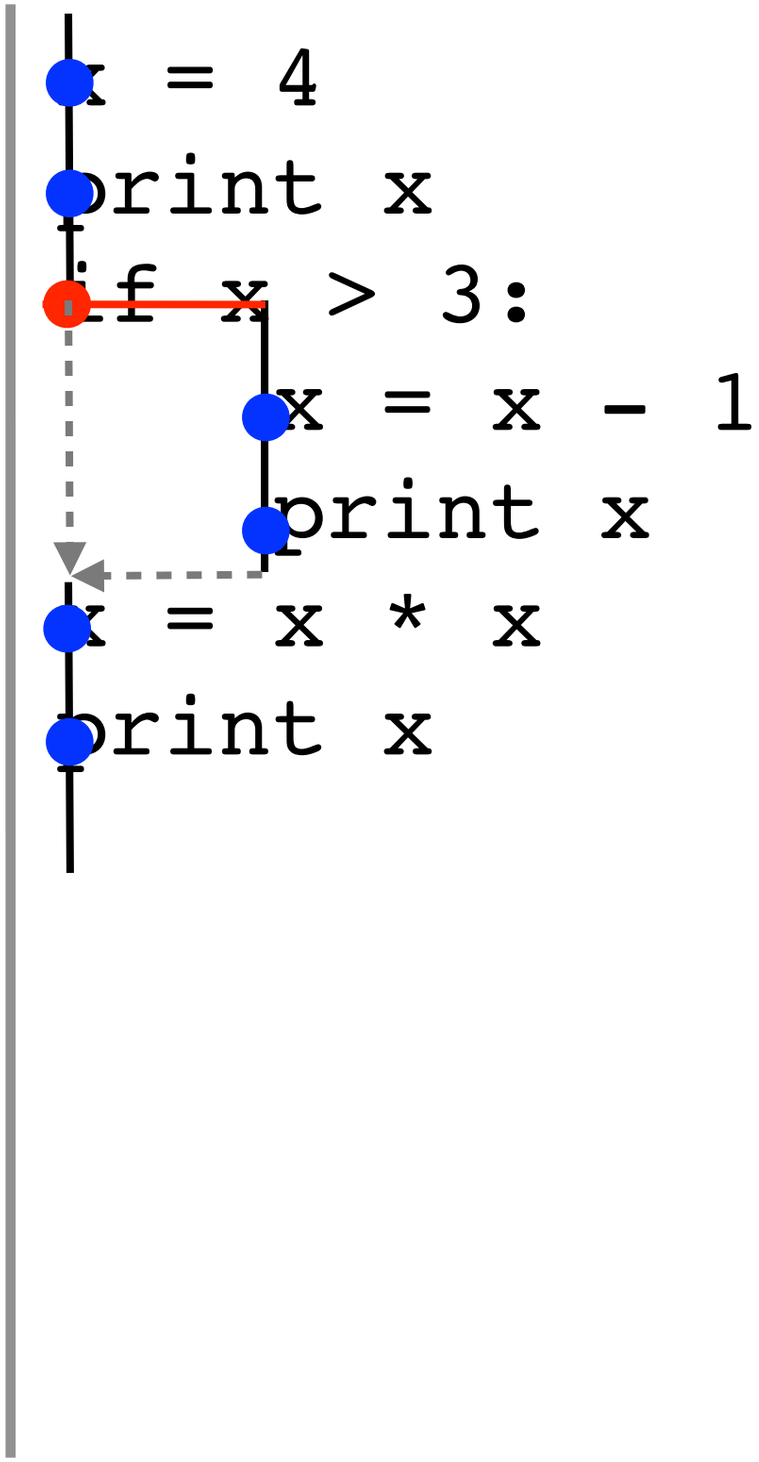
Control flow

Arbejde

Dynamisk kontrol



Control flow



Formalismen for control flow

Keywords

Sandt/falsk udtryk

Kolon

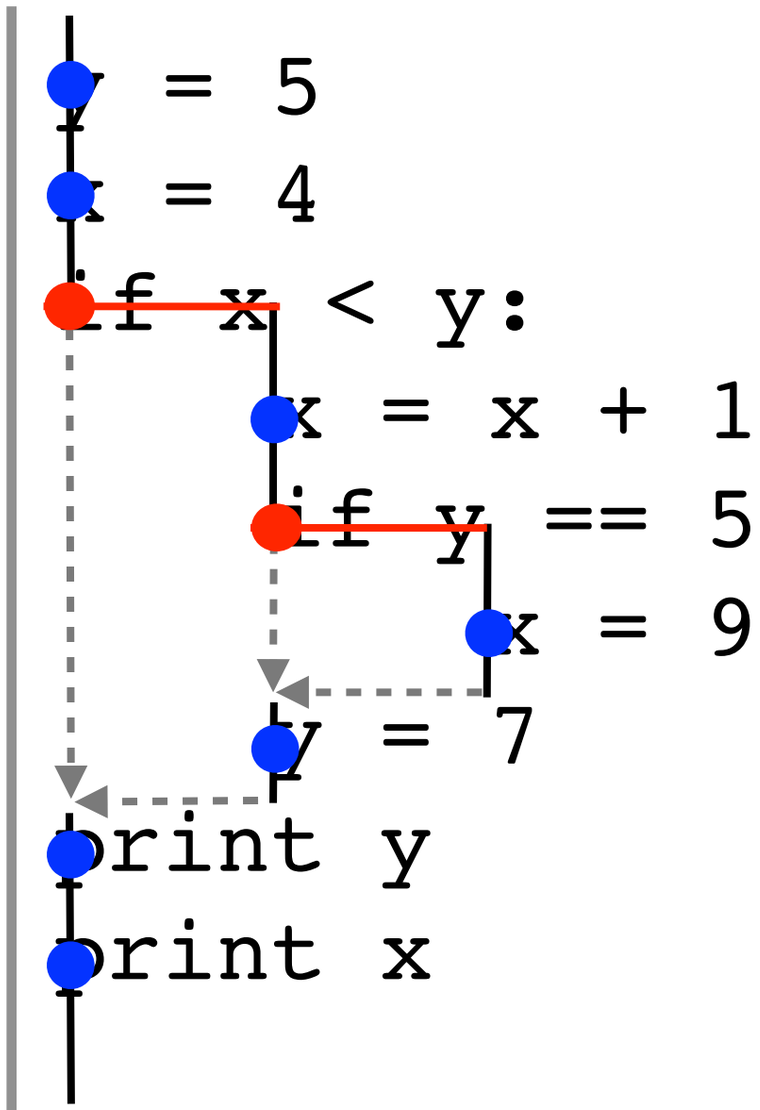
Indentering

```
x = 4
print x
if x > 3:
    x = x - 1
    print x
x = x * x
print x
```

Testudtryk - evalueres til sandt eller falsk

```
x == y
x != y
x > y
x => y


0
1
27
x


True
False
None
```

Flere lag af betingede udtryk



Else keyword

```
y = 5
x = 4
if x < y:
    x = x + 1
else:                  # “if x >= y”.
    x = x - 1
print y
print x
```

Opgave

sunny = True/False

windy = True/False

Skriv noget kode der kan skrive alle kombinationer.

“Sunny and windy”, “Sunny and calm”

“Cloudy and windy” , “Cloudy and calm”

Løsning på opgave

```
sunny = True
windy = False

if sunny:
    if windy:
        print “Sunny and windy”
    else:
        print “Sunny and calm”
else:
    if windy:
        print “Cloudy and windy”
    else:
        print “Cloudy and calm”
```

Kombinere og negere testudtryk
and, or, not

```
if x == y and z > 4:
    print x, y, z


if nr_patients > 1000 or nr_birds == 0:
    print “Hmm - something wrong”

if not sunny:
    print “Appears to be cloudy”
```

elif keyword

```
if x == 1:
    print “x is one”
else:
    if x == 2:
        print “x is two”
    else:
        if x == 3:
            print “x is three”
        else:
            print “x is something else”
```

elif keyword

```
if x == 1:
    print “x is one”
elif x == 2:
    print “x is two”
elif x == 3:
    print “x is three”
else:
    print “x is something else”
```

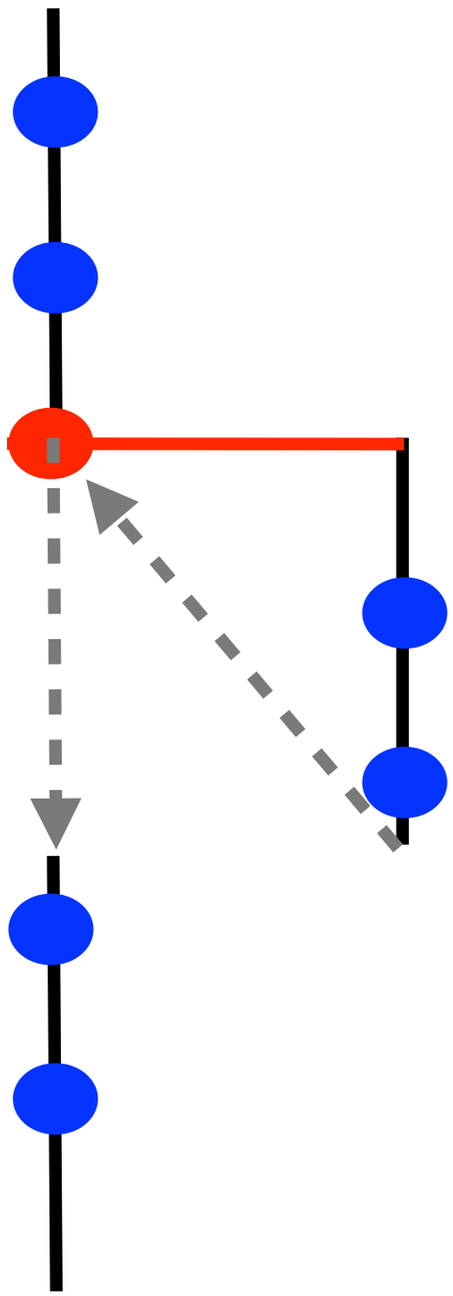
Opgave

Samme som opgave 1, men prøv nu at gøre det ved hjælp af "and", "not" og "elif" i stedet for flere niveauer af if-else inden i hinanden.

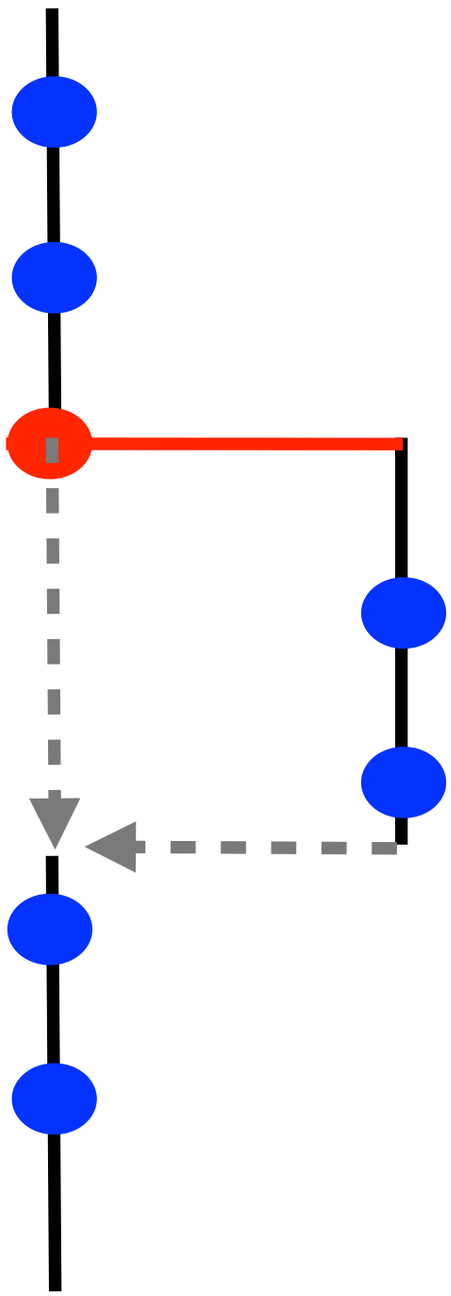
Løsning på opgave

```
if sunny and windy:
    print “Sunny and windy”
elif sunny and calm:
    print “Sunny and calm”
elif not sunny and windy:
    print “Cloudy and windy”
else:
    print “Cloudy and calm”
```

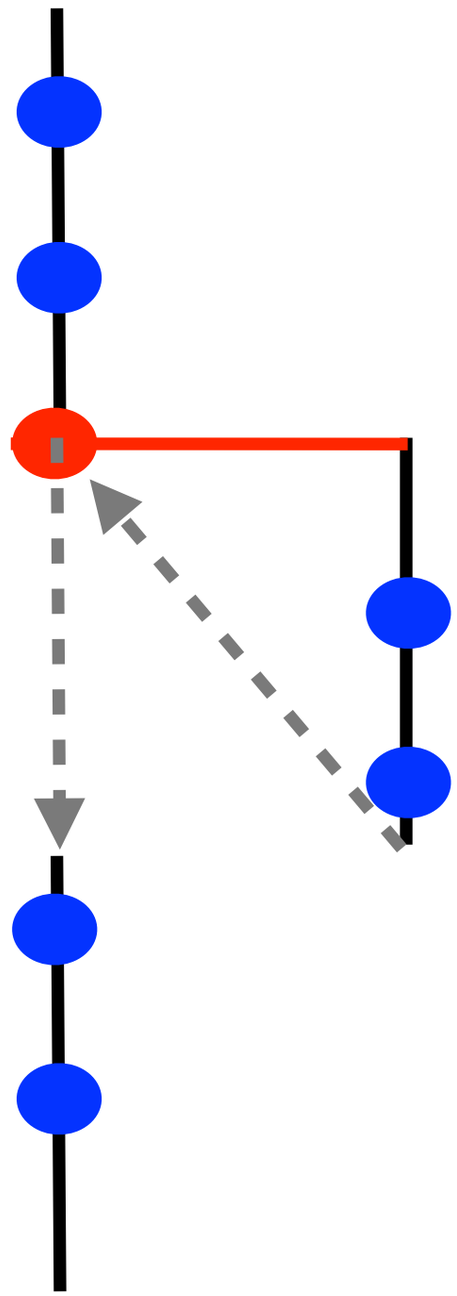
While loops



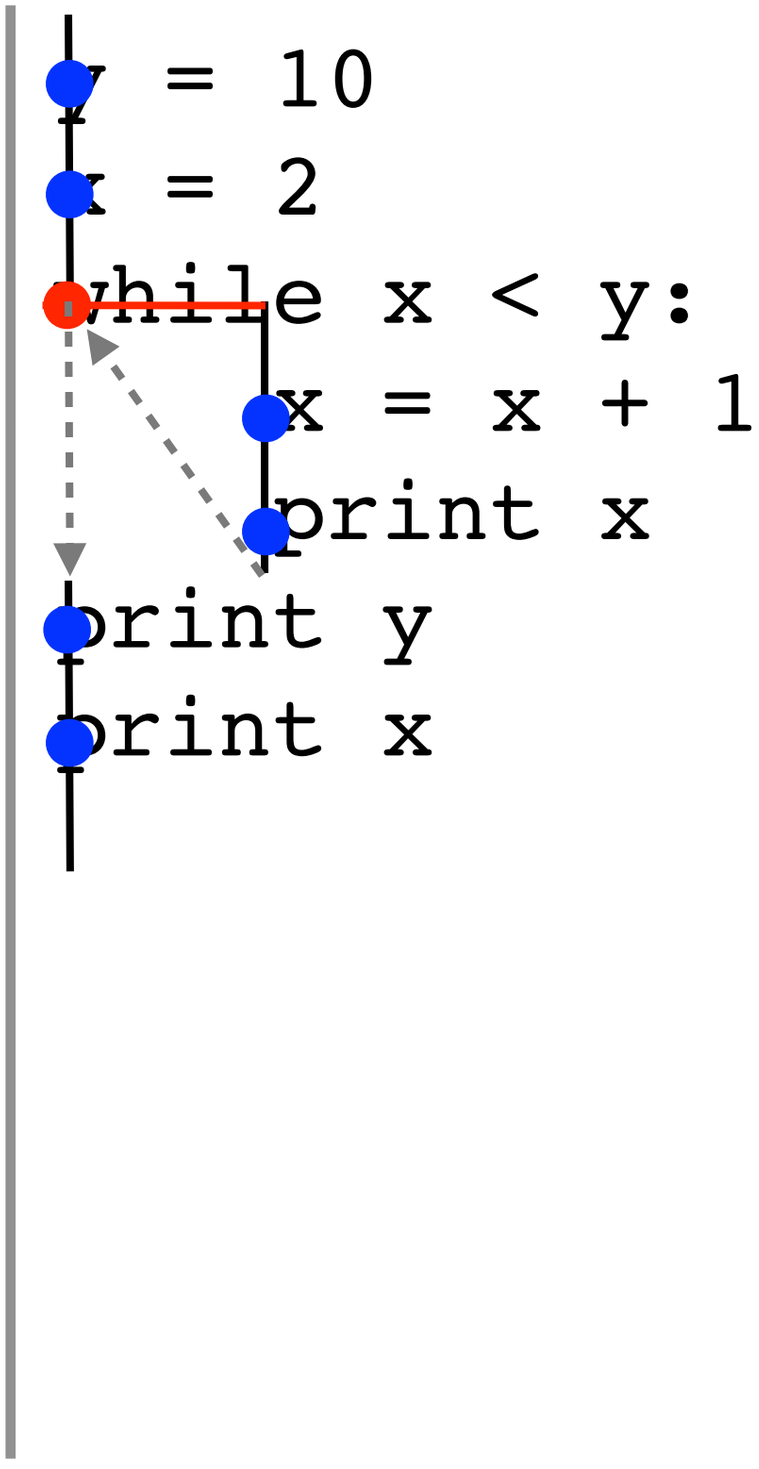
If



While



While loops



Opgave

x = 4

y = 42

Læg en til x ved hjælp af et while loop så x til sidst får samme værdi som y.

Prøv at printe alle x-værdierne på vejen.

Løsning på opgave

```
x = 3
y = 42

while x < y:
      x = x + 1
      print x

# eller lidt mere elegant:
while x < y:
      x += 1
      print x
```

Funktioner - hvad er det, hvad kan de?

- Små programmer i programmet
- Nyttigt til genbrug af kode og til at skabe struktur og orden i koden.

Funktioner inbygget i Python

min, max, help, len, sum

range

exp, log, sqrt, cos, sin, tan

Hvordan bruger man funktioner?

```
y = f(x)

y = log10(100)

absValue = abs(-4)

max_dosis = max(dosis1, dosis2)

print range(5)
```

Hvordan definerer man egne funktioner?

```
def multiplyBySeven(x):
    r = x * 7
    return r


oldValue = 2

newValue = multiplyBySeven(oldValue)
```

Opgave

Skriv en funktion der lægger to tal sammen.

```
r = addNumbers(10, 82)

print r

print addNumbers(10, 82)
```

Løsning på opgave

```
def addNumbers(x, y):
    r = x + y
    return r

# or shorter:
def addNumbers(x, y):
    return x + y
```

Argumenter, parametre og scope

```
def voresFunktion(x)
    z = x
    return z

# x: funktionens parameter
# z og x er private funktionsvariaber

c = voresFunktion(a)

# a: funktionens argument
# a og c kan ses udenfor funktionen
```

Uge 2

Torsdag

Dagens tekst

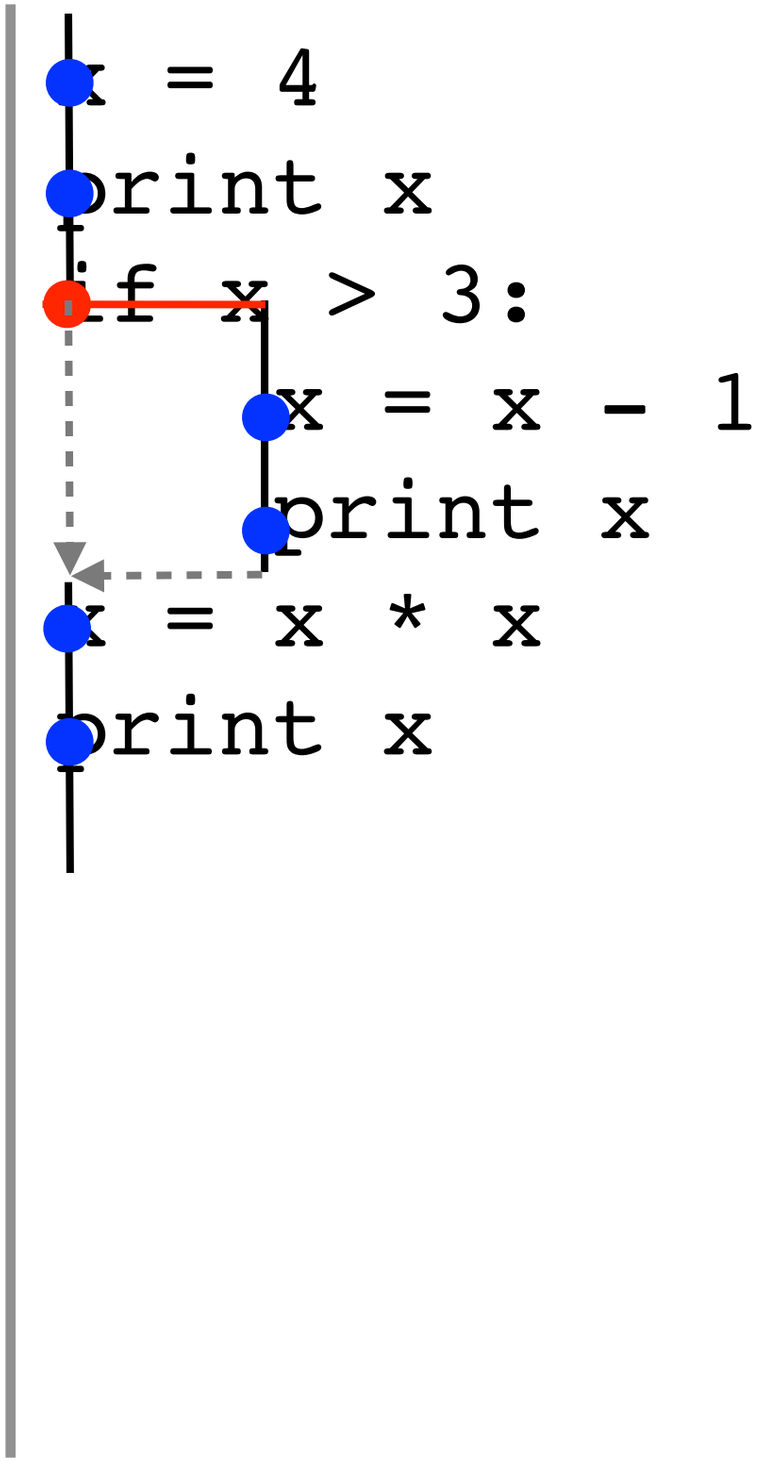
Control flow

Funktioner og control flow

Scope - definitionsområder

Moduler

Control flow



Formalismen for control flow

Keywords

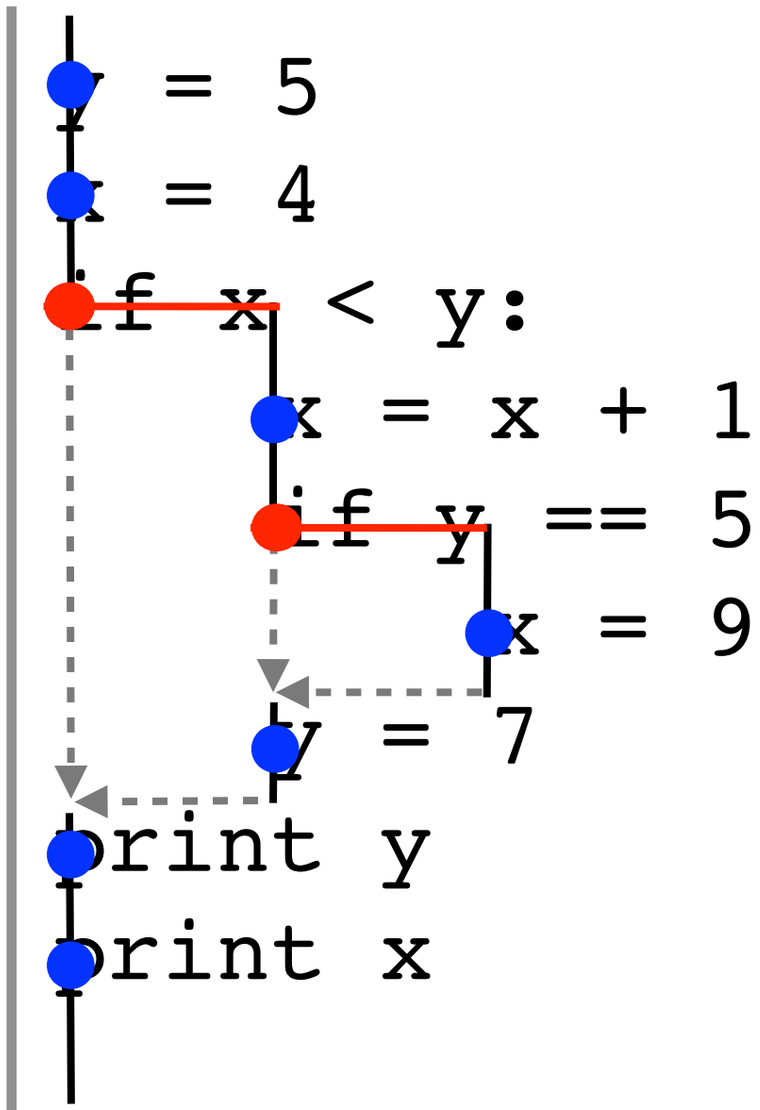
Sandt/falsk udtryk

Kolon

Indentering

```
x = 4
print x
if x > 3:
    x = x - 1
    print x
x = x * x
print x
```

Flere lag af betingede udtryk



Testudtryk

```
x == y
x != y
x > y
x => y


0
1
27
x


True
False
None
```

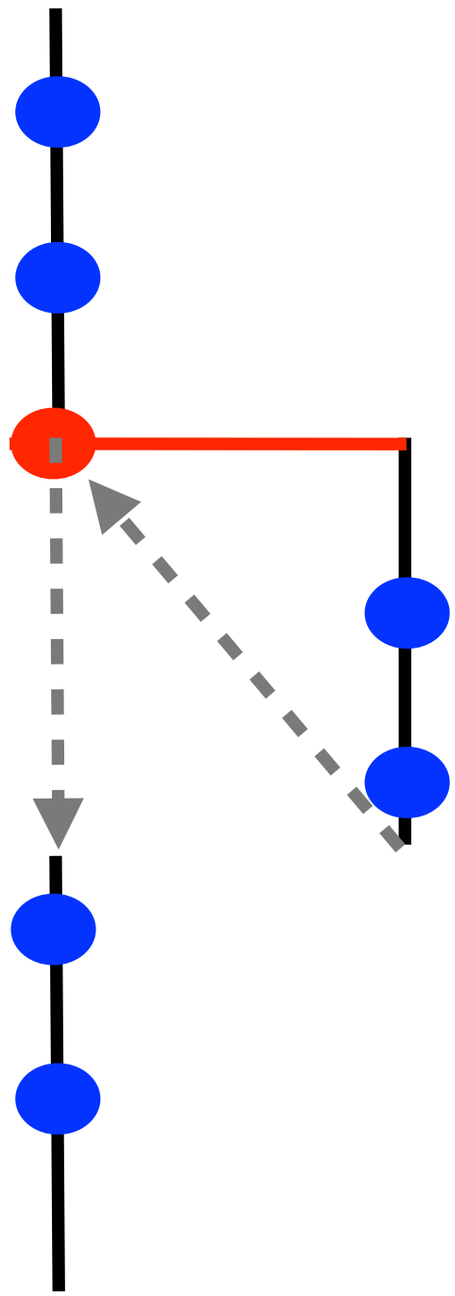
Boolske udtryk

```
if x > y:
    print x, y

if x:
    print “Hvad er værdien af x?”

if not x:
    print “Hvad er værdien af x?

if x and y:
    print “Hvad kan vi sige om x og y?”

if x or y:
    print “Hvad kan vi sige om x og y?”
```

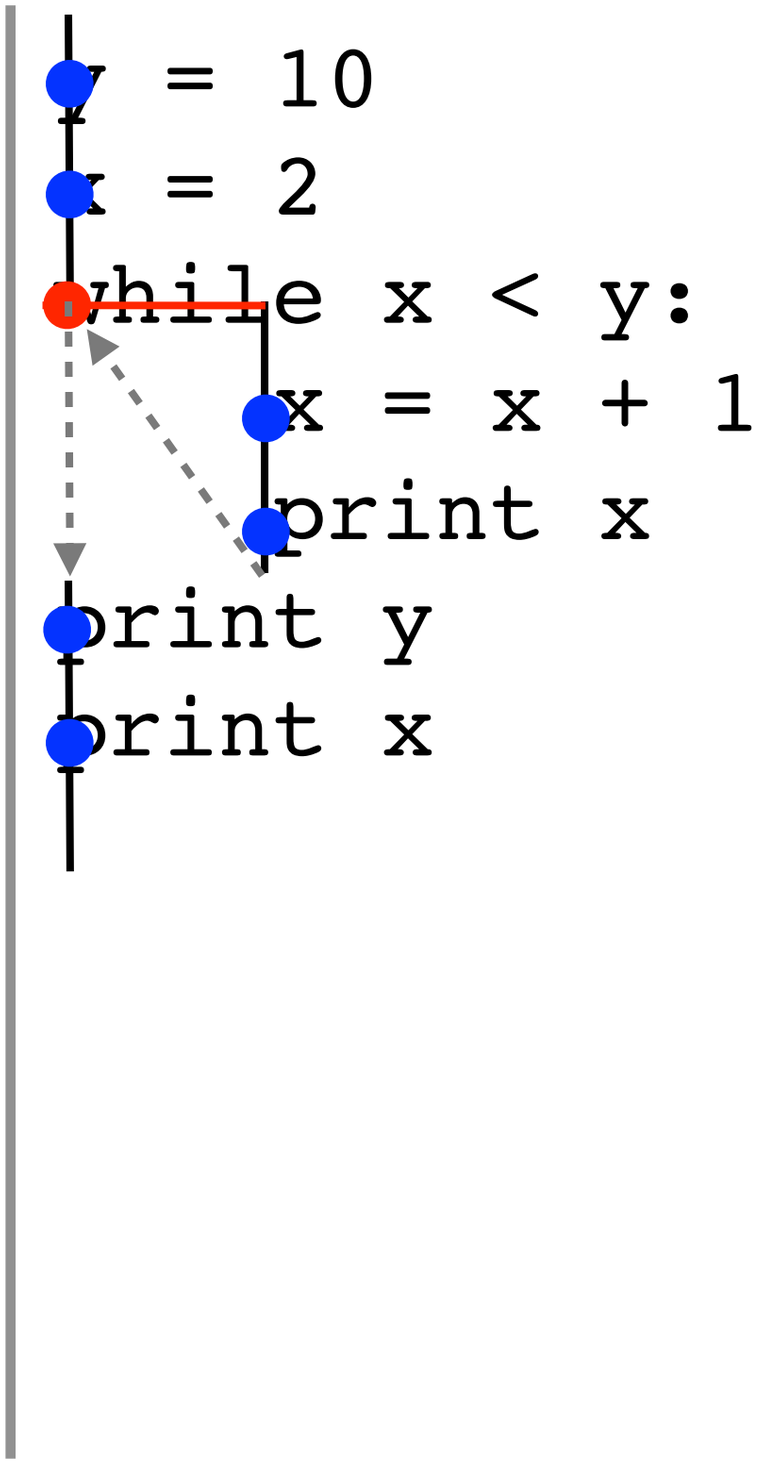
Else keyword

```
y = 5
x = 4
if x < y:
    x = x + 1
else:                  # “if x >= y”.
    x = x - 1
print y
print x
```

While loops



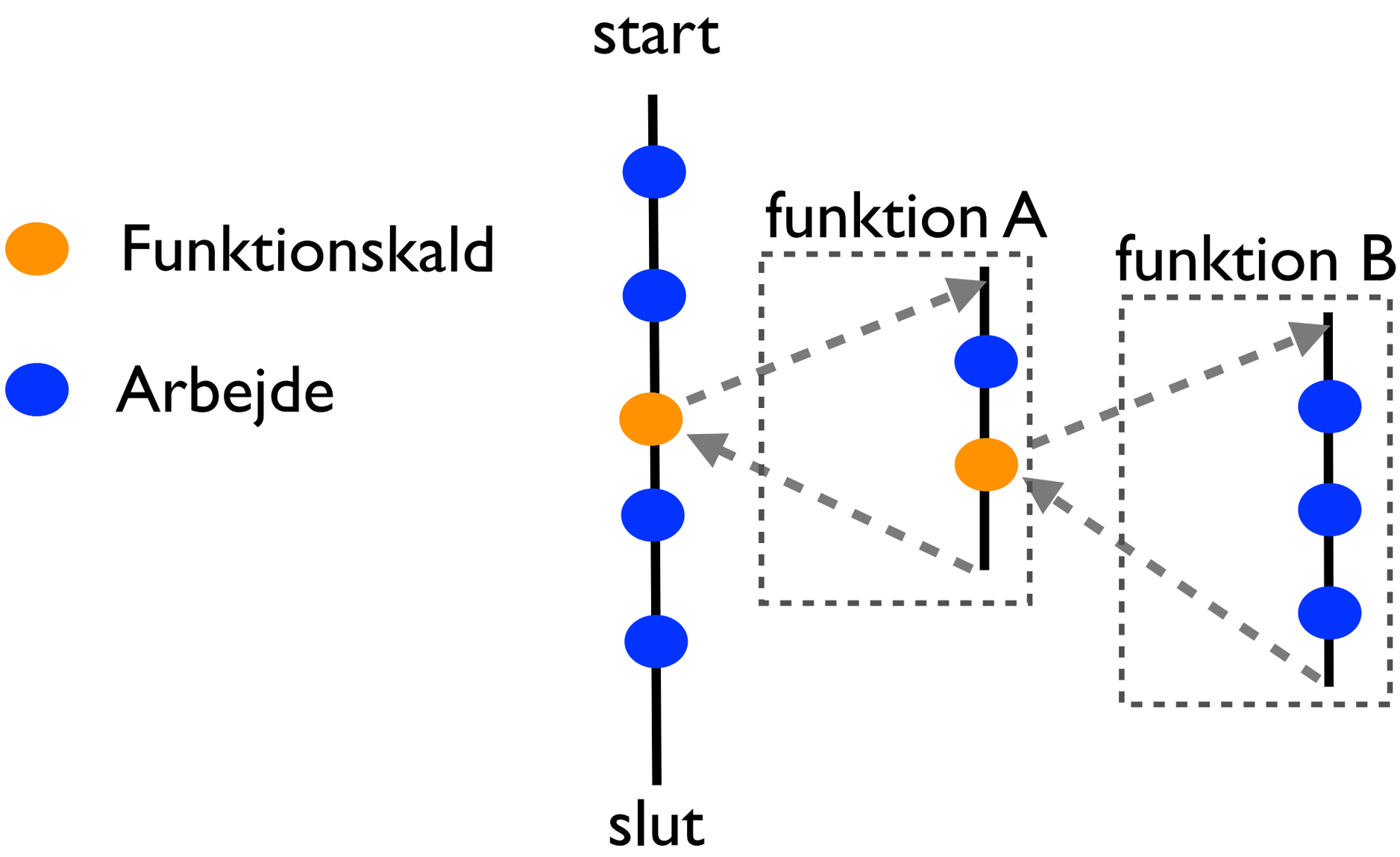
While loops



Funktioner - hvad er det, hvad kan de?

- Små programmer i programmet
- Nyttigt til genbrug af kode og til at skabe struktur og orden i koden.

Funktioner og program flow



Tænk på funktioner som små separate programmer
som modtager og returnerer variabler.

Antal argumenter

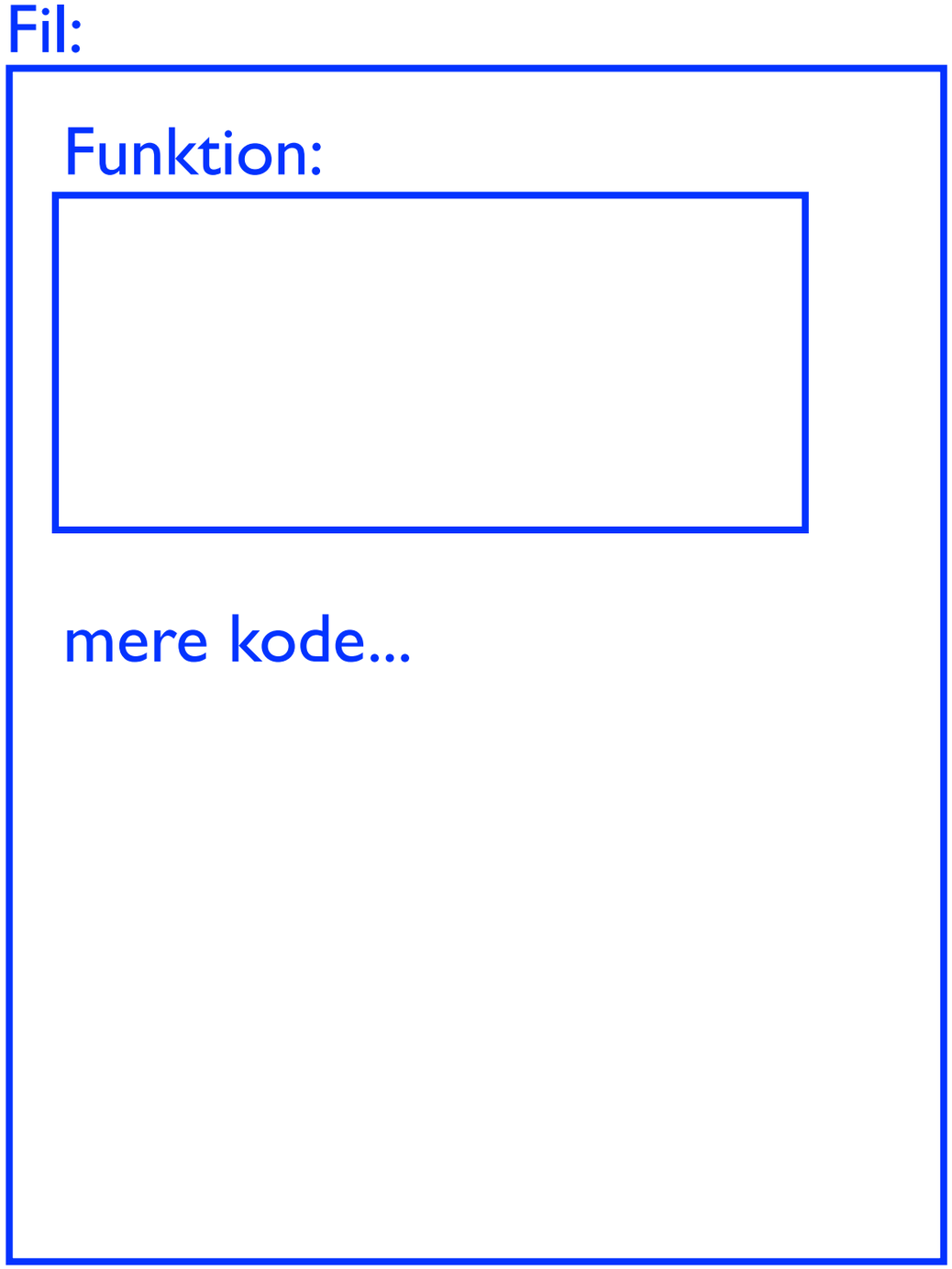
```
def printMyName():
    print “Kasper”

def printName(name):
    print name

def printFullName(firstName, lastName)
    print firstName, lastName
```

Antal retur værdier

```
def addOne(number):
    result = number + 1
    return result
```

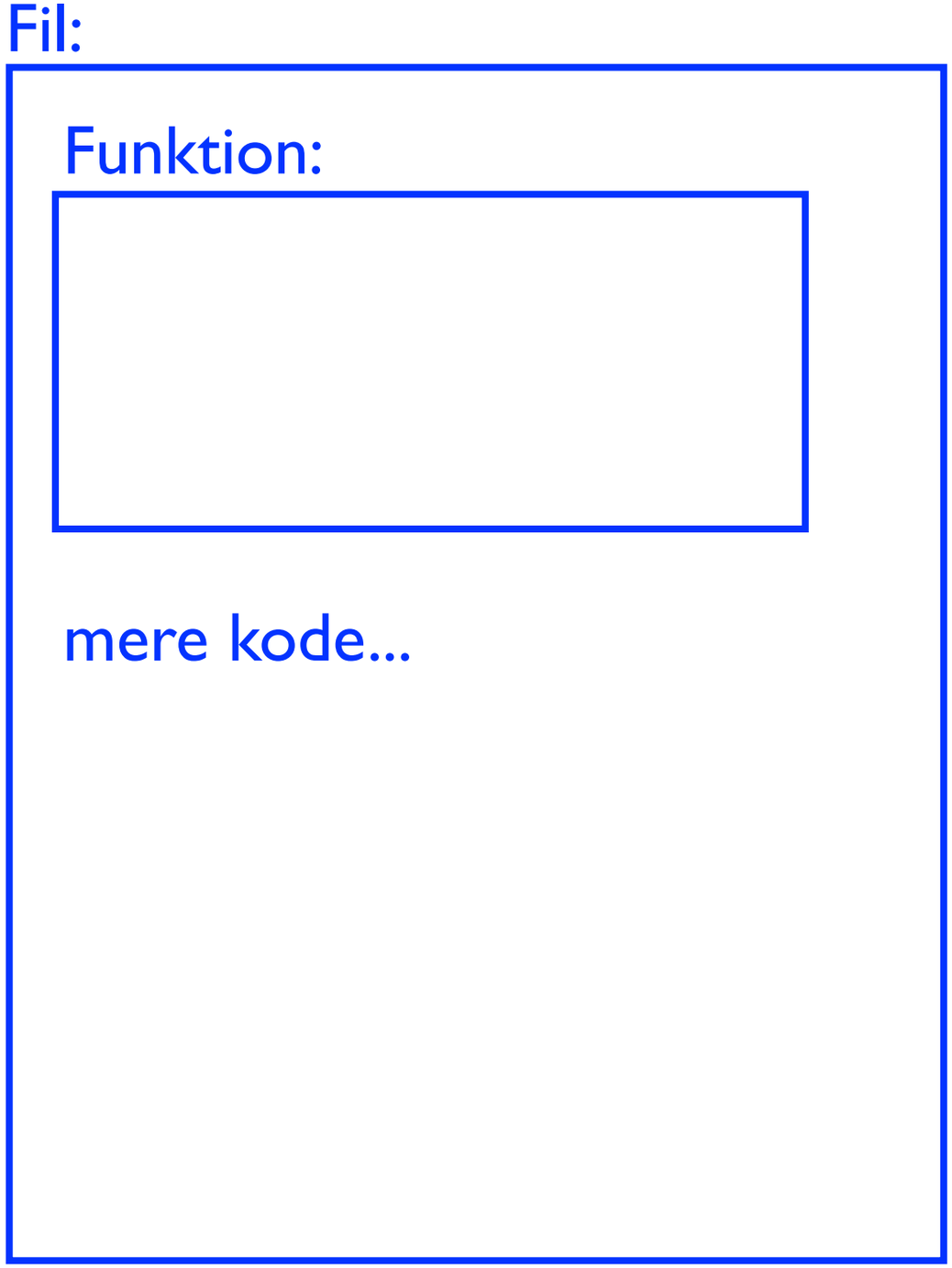
```
def addOne(number):
    return number + 1

def substractValues(number1, number2)
    return number1 - number2
```

Variablers scope eller synlighed



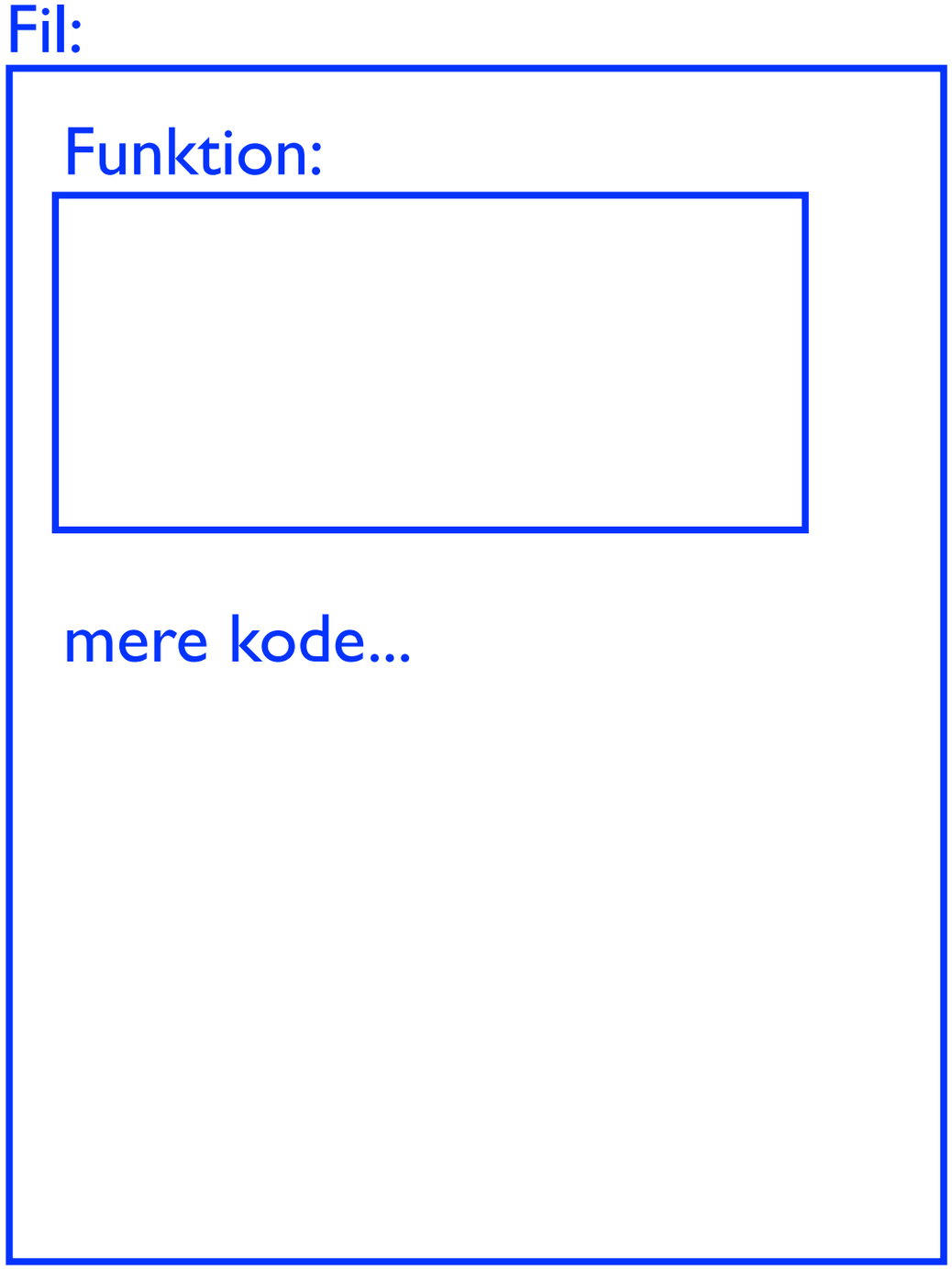

```
def function():
    print x


x = 4
print x

function()
print x
```

Variablers scope eller synlighed




```
def function():
    x = 3
    print x

x = 4
print x

function()
print x
```

Variablers scope eller synlighed




```
def function(x):
    print x


x = 4
print x

function(x)
print x
```

Python moduler

Et modul er en fil med python kode.

Moduler er nyttige til genbrug og strukturering af større mængder kode - lidt analogt til funktioner.

Et modul har sit eget scope ligesom en funktion.

Importering af moduler

```
import math

math.sqrt(4)
```

```
from math import sqrt

sqrt(4)
```

```
dir(math)
```

Moduler og name spaces

Fil (script):

```
import  module
x = 7
print x
print module.x


from module import x
print x
```

Fil (module.py):

```
x = 3
```

Funktion:

```

```

Funktion:

```

```