Lær Python dag 1 - modul 2

Conditionals, funktioner

Jonas Bamse Andersen

Institut for Matematik og Datalogi - IMADA
Syddansk Universitet

Indhold

1. Conditionals
2. Funktioner

Conditionals

Boolske udtryk - sandt og falsk

Vi har ofte brug for at sammenligne ting, for at tage beslutninger. For at sammenligne om ting er lig med hinanden bruges dobbelt lig med, ==.

```
print(4 == 4)
print(4 == 2)
print(type(True))
```

Boolske udtryk - sandt og falsk

Vi har ofte brug for at sammenligne ting, for at tage beslutninger. For at sammenligne om ting er lig med hinanden bruges dobbelt lig med, ==.

```
print(4 == 4)
print(4 == 2)
print(type(True))
```

```
True
False
<type 'bool'>
```

Boolske udtryk - sandt og falsk

Her er en liste af sammenligningsoperatorer.

```
x == y  # er x lig med y?
x != y  # er x forskellig fra y?
x < y   # er x mindre end y?
x <= y  # er x mindre end eller lig med y?
x > y
x >= y
```

Eksempler, alle giver True.

```
4 > 2
2+2 == 4
3 != 4
"hej" == "h" + "ej"
```

Boolske udtryk - sandt og falsk

Man kan også bruge `not` til at skifte `True` til `False` og omvendt:

not $(3 < 4)$
not (True **and** False)
not True **and** False

Boolske udtryk - sandt og falsk

Man kan også bruge `not` til at skifte `True` til `False` og omvendt:

```
not (3 < 4)
not (True and False)
not True and False
```

Output

```
False
True
False
```

If-sætninger - at tage et valg

Meget ofte vil man gerne tage en beslutning, hvor man gør forskellige ting alt efter situationen, f.eks. kan man ikke bruge 10 kr, **hvis** man har ikke har så mange penge i sin pung.

Til dette har programmering if-sætninger, f.eks. i følgende eksempel, **(bemærk indrykningen!)**:

```
money = 52
if (money >= 10):
    money = money - 10
    print("You bought an expensive banana")

print("You have " + str(money) + " money left")
```

If-sætninger - at tage et valg

Man kan også tilføje noget man vil gøre hvis det første ikke var tilfældet:

```
money = 52
if (money >= 10):
    money = money - 10
    print("You bought an expensive banana")
else:
    print("Go earn some more money!")

print("You have " + str(money) + " money left")
```

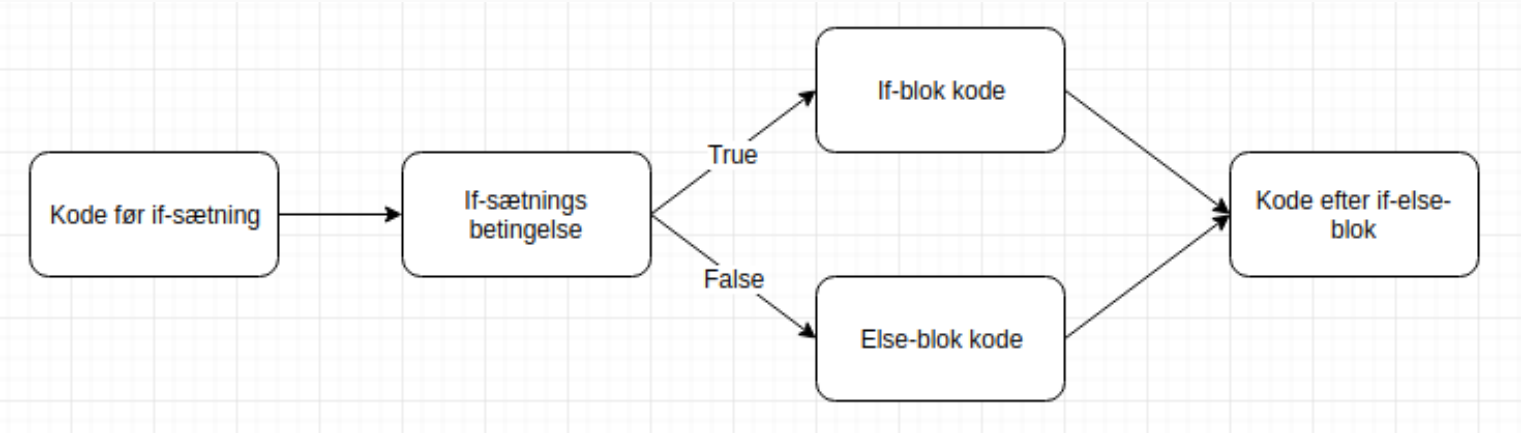
If-sætninger - at tage et valg

En if-sætning er opbygget af en eller flere betingelser og en eller flere "kroppe" af kode.

```
if (<betingelse>):
    #kode
elif (<betingelse):
    #kode
else:
    #kode
```

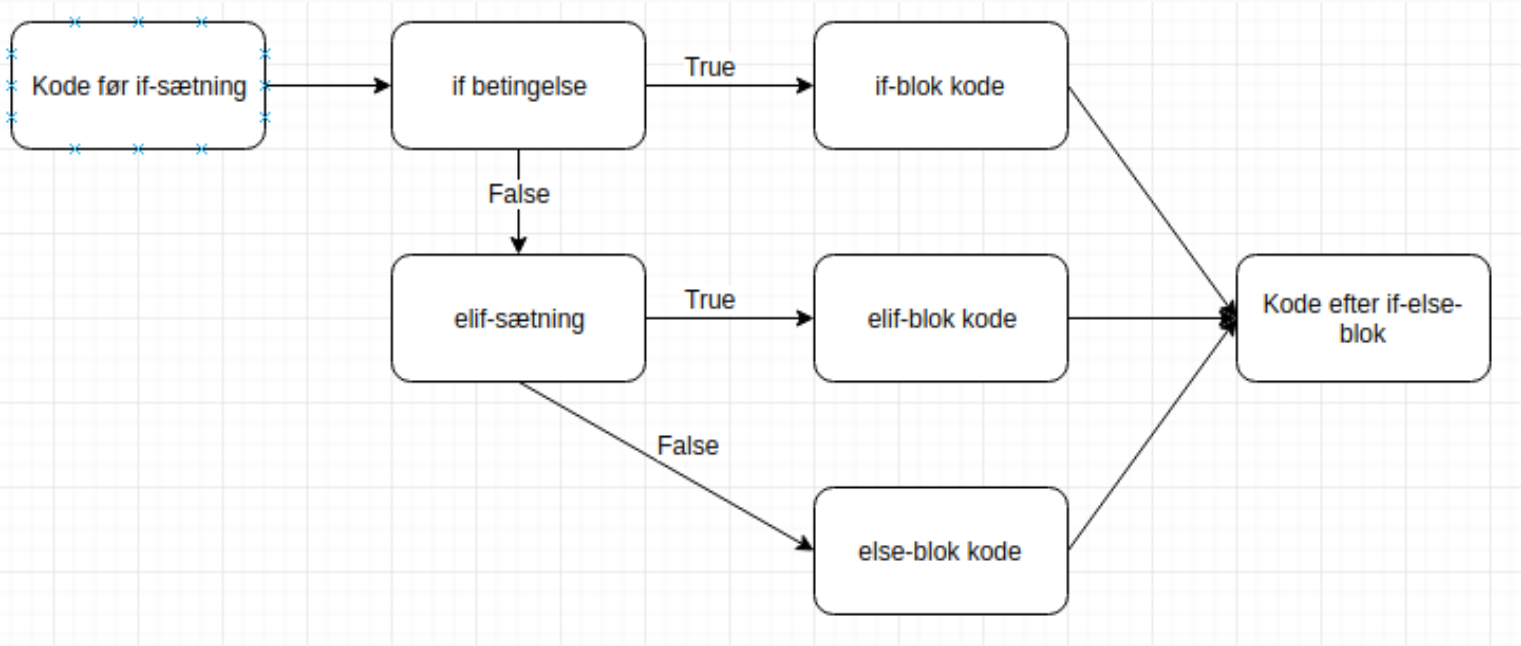
If-sætninger - at tage et valg

Man kan også tænke på det som et flow diagram.



If-sætninger - at tage et valg

Og med `elif` ser det således ud:



Husk at der kan være lige så mange `elif`'s man har lyst, men den første der er sand vil blive valgt og de andre vil så ikke blive tjekket.

If-sætninger - at tage et valg

Måske skal der gælde mere end én ting før man vil gøre noget. F.eks. køber man kun bananen, hvis man både er over 18 og har penge nok.

$$\text{age} = 20$$
$$\text{money} = 42$$

If-sætninger - at tage et valg

Måske skal der gælde mere end én ting før man vil gøre noget. F.eks. køber man kun bananen, hvis man både er over 18 og har penge nok. Her er en måde at løse det på:

```
age = 20
money = 42
if (age >= 18):
    if (money >= 10):
        money = money - 10
        print("You bought an expensive banana")
```

If-sætninger - at tage et valg

Der kan også være situationer, hvor bare én af flere ting er nok, for at udføre noget. F.eks. kan enten en fødselsdag eller en bommert være anledning til kage. Hvordan vil man så skrive sit program?

```
birthday = False
blunder = True
```

If-sætninger - at tage et valg

Der kan også være situationer, hvor bare én af flere ting er nok, for at udføre noget. F.eks. kan enten en fødselsdag eller en bommert være anledning til kage. Hvordan vil man så skrive sit program? Måske sådan her:

```
birthday = False
blunder = True
caketime = False
if (birthday):
  caketime = True
if (blunder):
  caketime = True

if (caketime):
  print("It's cake time!")
```

If-sætninger - at tage et valg

Heldigvis findes der en bedre måde. Man kan sammensætte boolske udtryk med `and` og `or`. Med `and` skal begge sider være opfyldt før det giver True, med `or` skal mindst én side være True før det giver True.

```
age = 20
money = 42
if (age >= 18 and money >= 42):
    money = money - 10
    print("You bought an expensive banana")
```

```
birthday = False
blunder = True
if (birthday or blunder):
    print("It's cake time!")
```

Funktioner

Funktioner - undgå gentagelser

I har faktisk allerede brugt flere funktioner mange gange, f.eks. `print` og `type` funktionerne. At bruge en funktion hedder at man kalder funktionen, vi snakker altså om funktionskald. Man kan kende en funktion på at der er parenteser efter navnet.

```
print("Hej")
type("Hej")
```

Det der står inde i parenteserne kaldes for parametre eller argumenter. Man kan også kalde det input til funktionen.

Funktioner - undgå gentagelser

Husk, man kan kombinere funktionskald. Så i stedet for:

```
a = 3.14/2
a = int(a)
print(a)
```

Så kan man skrive:

```
print(int(3.14/2))
```

Funktioner - undgå gentagelser

Forestil jer I har skrevet noget lækkert kode, men I skal hele tiden trække 1 fra og så printe værdien.

```
a = 7
b = a - 1
print(b)
c = (3**2 + 4**2)**0.5
d = c - 1
print(d)
e = 17 // 3
f = e - 1
print(f)
```

Funktioner - undgå gentagelser

Forestil jer nu at I finder ud af at I skulle lægge 1 til i stedet for, så skal I finde alle de steder hvor I havde skrevet det og ændre det.

```
a = 7
b = a + 1
print(b)
c = (3**2 + 4**2)**0.5
d = c + 1
print(d)
e = 17 // 3
f = e + 1
print(f)
```

Output

```
8
6.0
6
```

Funktioner - undgå gentagelser

Men programmører er dovne, vi kan gøre det smartere med funktioner:

```
def newPrint(x):
    print(x+1)


a = 7
newPrint(a)
c = (3**2 + 4**2)**0.5
newPrint(c)
e = 17 // 3
newPrint(e)
```

Output

```
8
6.0
6
```

Funktioner - undgå gentagelser

En funktion er opbygget således:

```
def <funktions navn>(<param. 1>, <param. 2>, ...):
  #kode
  return <værdi>
```

Hvis man har et `return` kan man sende et resultat tilbage og gemme/bruge til noget andet.

Funktioner - undgå gentagelser

Her er et eksempel på noget kode der udregner Pythagoras men som er copy-pastet rundt:

```
a = 4
b = 2
c = (a**2 + b**2)**0.5
a = 3
b = 4
d = (a**2 + b**2)**0.5
```

Funktioner - undgå gentagelser

Her er den samme kode, men hvor der er lavet en funktion istedet:

```
def pyth(a, b):
    x = (a**2 + b**2)**0.5
    return x
c = pyth(4, 2)
d = pyth(3, 4)
```

Her var der to parametre i stedet for en, og vi gemte retur-værdien.

Funktioner - undgå gentagelser

En funktion behøver hverken have en retur værdi eller nogle parametre, den kan godt bare gøre det samme hver gang. F.eks.:

```
def print_twice():
    s = "Hej med dig!"
    print(s)
    print(s)

print_twice()
```

Recap

Hvad har vi set på i dette modul?

- Sammenligningsoperatorer
- if-else-sætninger
-