Lær Python, dag 1 - modul 2

Institut for Matematik og Datalogi, Syddansk Universitet

Mikkel Juul Vestergaard

https://moggel12.github.io/PythonWorkshop2021/

14. november, 2021

Indhold

Indentering/Indrykning

Vi kommer nu til at arbejde med kode hvor indentering/indrykning af koden har en effekt på hvordan Python forstår koden.

Til dem af jer der har programmeret i et c-like sprog er indentering Pythons måde at lave { og }

Conditionals

Boolske udtryk — sandt og falsk

Nogle gange vil vi gerne kunne køre noget kode hvis én ting gælder, mens vi kører andet kode hvis en anden gælder. Fx vil vi i et login system vise brugerens profil hvis den er logget ind, ellers vil vi vise en standard login-side.

Boolske udtryk — sandt og falsk

Typisk arbejder vi med sammenligninger når vi bruger condittionals. For at tjekke om noget er lig noget andet bruges dobbelt lig med, ==. (Husk, en enkelt = er forbeholdt til at give variabler en værdi)

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

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

Boolske udtryk — sandt og falsk

Her er en liste af sammenlignings operatorer.

```
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

Altså er et boolsk udtryk en måde at teste om noget er sandt eller falsk, hvilket også kan siges som at det giver os en *sandhedsværdi*.

Men hvad nu hvis vi både vil tjekke om brugeren er logget ind **og** om den er en del af vores *premium* service?

Boolske operatorer

Følgende operatorer arbejder på sandhedsværdier og giver en sandhedsværdi tilbage:
Ved **and** skal begge udtryk være sande, for at give sand.
Ved **or** skal et af udtrykkene være sande, for at give sand.
not-operatoren vender sandhedsværdien.

```
print(not False)
print(True and False)
print(True or  False)
```

Boolske operatorer

Følgende operatorer arbejder på sandhedsværdier og giver en sandhedsværdi tilbage:
Ved **and** skal begge udtryk være sande, for at give sand.
Ved **or** skal et af udtrykkene være sande, for at give sand.
not-operatoren vender sandhedsværdien.

```
print(not False)
print(True and False)
print(True or  False)
```

```
True
False
True
```

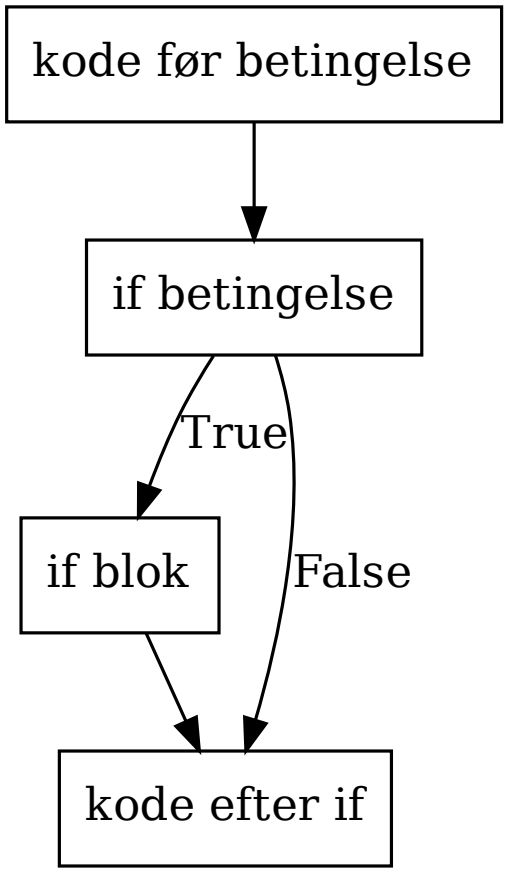
If-sætninger — at tage et valg

Generelt:

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

Eksempel:

```
money = 100
if (money >= 1000):
   print("You are a rich")
print("man")
```



Bemærk indrykningen.

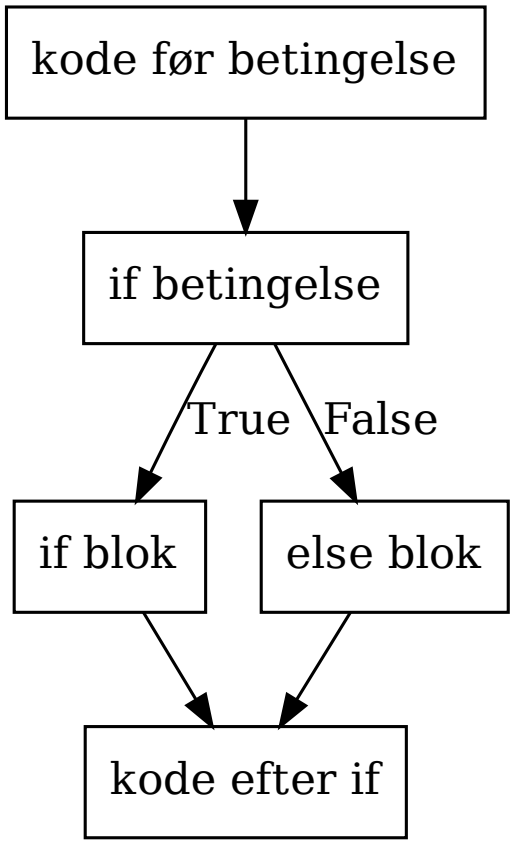
If-sætninger — at tage et valg

Generelt:

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

Eksempel:

```
money = 100
if (money >= 1000):
  print("You are a rich")
else :
  print("You are a poor")
print("man")
```



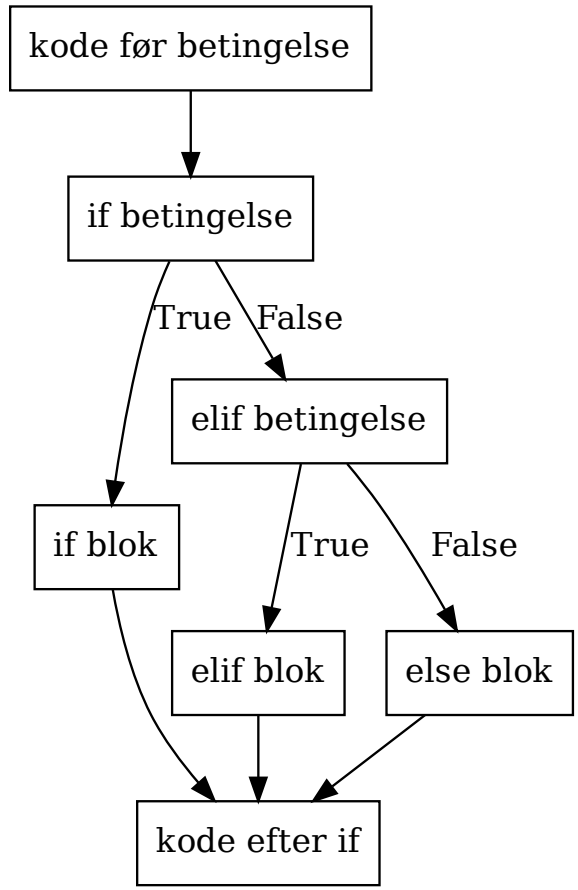
If-sætninger — at tage et valg

Generelt:

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

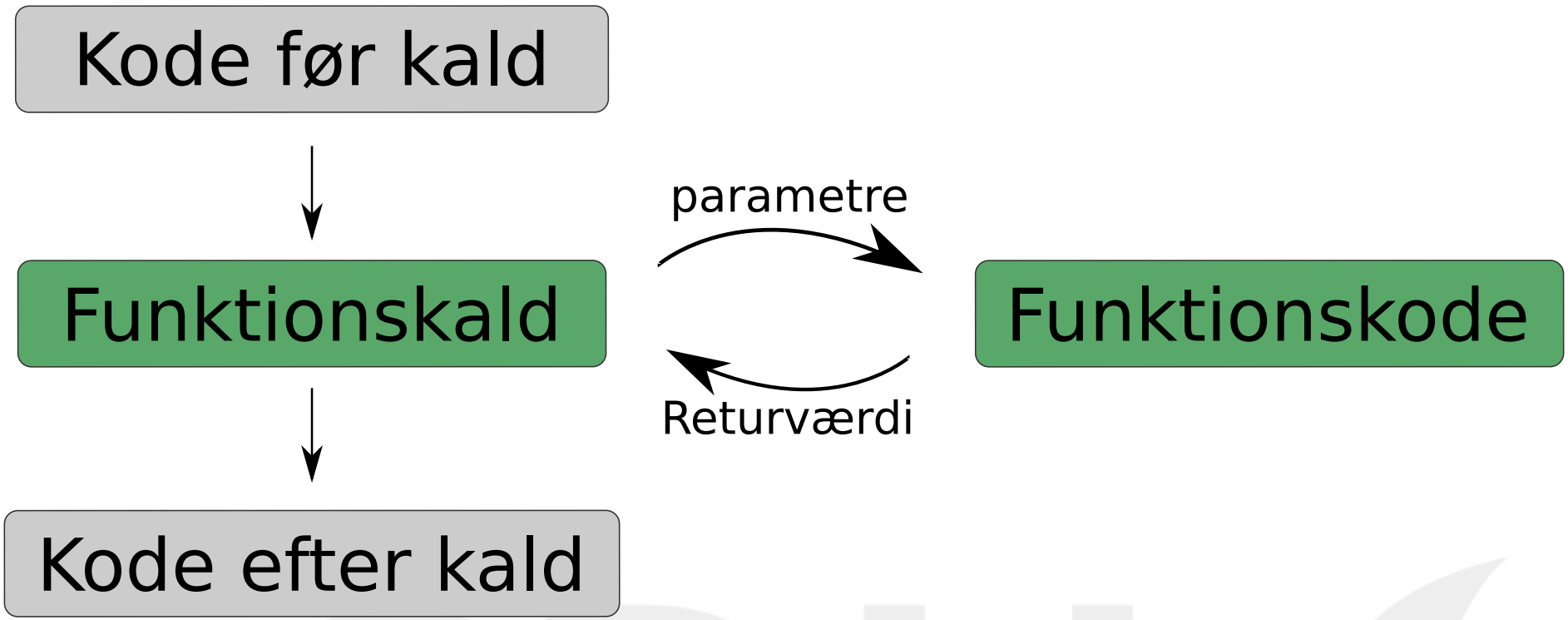
Eksempel:

```
money = 100
if (money >= 1000):
  print("You are a rich")
elif (money >= 100):
  print("You are kind of a rich")
else:
  print("You are a poor")
print("man")
```



Funktioner

Funktioner — Don't Repeat Yourself!



Funktioner — Don't Repeat Yourself!

I har faktisk allerede brugt flere funktioner mange gange, f.eks. `print(_)` og `type(_)` funktionerne.
At bruge en funktion = at kalde en funktion (et funktionskald).

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

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

Funktioner — Don't Repeat Yourself!

Man kan kombinere funktionskald.
I stedet for:

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

Kan man skrive:

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

Funktioner — Don't Repeat Yourself!

En funktion er opbygget således:

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

Hvis man har et return kan man sende et resultat tilbage og gemme/bruge dette til noget andet. Dette er dog ikke et krav.

Funktioner — Don't Repeat Yourself!

Forestil jer I har skrevet noget lækkert kode, f.eks. noget der udregner Pythagoras, og I bruger det igen og igen, måske ser det sådan her ud fordi I copy-paster:

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

Funktioner — Don't Repeat Yourself!

Det er ikke optimalt at copy-paste det til alle stederne, hvor det skal bruges, f.eks. hvis man skal ændre noget i det, så skal man ændre det alle stederne. Heldigvis kan man lave sine egne funktioner, så det bliver meget pænere og lettere at rette.

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

First steps — reloaded

Vi har nu kigget på variabler, datatyper, udtryk, boolske udtryk + operatorer, conditionals (`if`-sætninger) og funktioner.
I skal nu kombinere disse til jeres næste opgavesæt.