Cheatsheet:

Konstanter og variabler

Når man programmerer så er det en fordel at kunne give navne til sine tal, lister osv. Så slipper man for at huske værdier og kan nemt ændre i dem senere. I *python* gøres dette ved at skrive et navn efterfulgt af et `=` og så den værdi man ønsker.
Det kunne f.eks være:

```
numberOfBooks = 10
title = "Lord of the flies"
```

I *python* kan en variabel f.eks være: tal, tekst eller sandhedsværdier.

- En sandhedsværdi har to muligheder `True` eller `False`
- Et tal kan være $-1, 5$ eller 0.5
- Tekst er en samling af bogstaver i anførselstegn som `'Hello'`.

Samlinger af variabler

Hvis man har flere variabler der hører sammen kan man med fordel samle den i en liste eller i en såkaldt `dictionary`.

- **Lister:**
  En liste i *python* er den simpleste samling af variabler, en liste kan bestå af tal, tekst, sandhedsværdier eller endda flere lister.

  ```
  salesNumbers = [100, 230, 76, 103, 300, 279]
  words = ['coffee', 'monkey', 'dinner']
  languages = [['kaffe', 'abe', 'middag'], ['coffee', 'monkey', 'dinner']]
  ```

  Nu peger variablen `salesNumbers` på en liste af tal, vi kan så tilgå tallene i listen ved at skrive `salesNumbers[4]` hvilket giver os et tal fra listen.
  Vigtigt: af historiske årsager så starter en liste fra 0, altså skriver vi `salesNumbers[0]` så får vi tallet 100 og skriver vi `salesNumbers[2]` får vi 76
- **Dictionaries:**
  En af ulemperne ved en liste er at man skal huske hvilken placering i listen de

forskellige variabler har. En `dictionary` er en liste hvor man istedet giver pladserne navne, så værdier består af `'navn': værdi`. Et eksempel er

```
book = {'title': "Lord of the flies", 'year' : 1954, 'author' : 'Willia
m Golding'}
```

Man tilgår en værdi således `book['year']` så vi i dette tilfælde får 1954.
Man kan indsætte nye værdier ved at skrive `book['language'] = 'english'`.

Indbyggede funktioner:

I *python* findes der en stor samling af indbyggede funktioner der kan bruges til at behandle ens data. Man bruger en funktion ved at skrive dens navn efterfulgt af af en paratens hvor man skriver den værdi man ønsker at udregne fra.
Her er nogle eksempler på indbygget funktioner. Efterfulgt af en kommentar om hvad den gør

```
salesNumbers = [100, 230, 76] # En liste vi kan kalde funktioner på.
len(salesNumbers) # Giver længden af listen, 3 i det her tilfælde
sum(salesNumbers) # Alle tallene lagt sammen, 406 i det her tilfælde
print(salesNumbers) # printer dens input
```

`print()` funktionen er specielt brugbar bliver brugt ofte, det er den nemmeste måde at se resultater fra ens kode.

Boolske udtryk:

Ligesom at tal kan kombineres med operationer som $1+2, 1-1, 10\cdot 4, 20/2$ så kan man bruge følgende udtryk til at kombinere forskellige sandhedsudtryk.

```
2 == 1  # Er falsk
a and b # Hvis a og b begge er sande bliver hele udtrykket sandt.
a or b  # Hvis enten b eller a er sand bliver hele udtrykket sandt
not(a)  # Gør sandt til falsk og falsk til sandt
```

For løkker og gentagelser

Ofte vil man gerne gentage noget et bestemt antal gange eller gøre noget for alle elementer i en liste. Hvis man ønsker at gøre noget 10 gange kan det gøres således

```
    for i in range(10):
        print(i) # Printer tallene fra 1 til 10
```

Ønsker man at gøre noget for f.eks alle ting i en liste så kan man skrive

```
    sales = [100, 132, 123, 312, 123]
    for sale in sales:
        print(sale) # Printer alle tal i listen
```

If sætninger:

Hvis man ønsker at give computeren en betingelse, kan det gøres således

```
    if book['year'] < 2000:
        print("Bogen udkom i sidste århundrede")

    else:
        print("Bogen udkom i dette århundrede")
```

Vi får altså et forskelligt svar alt efter om den bog vi kigger på er fra dette år 100 eller ej.

Brugerfunktioner:

Når man har skrevet en bid kode man gerne vil genbruge så kan man gemme den som en funktion. Lad os f.eks sige at vi vil lave en funktion der givet en liste af tekst skal tælle hvor mange gange bogstavet `e` fremgår.

```
sentences = ['I like books', 'Food is good', 'Python is nice']
def numberOfEs(lines):
    e_counter = 0
    for line in lines:
        if 'e' in line:
            e_counter += 1
    return e_counter

print(numberOfEs(sentences))
```

Linjen der starter med `def` fortæller python at vi ønsker at definere en ny funktion og hvad den skal hedde og at den tager et input vi kalder lines. Linjen med `return` fortæller python hvad vi ønsker at give som svar på funktionen.