Jupyter Notebooks som lommeregner

Table of contents

Simple beregninger 1
Opgave 1 1
Variable 1
Opgave 2 2
Opgave 3 2
Funktioner 3
Opgave 4 3
Specielle funktioner 3
Opgave 5 4


```
try:
    import fysisk_biokemi
    print("Already installed")
except ImportError:
    %pip install -q "fysisk_biokemi[colab] @ git+https://github.com/au-mbg/fysisk-biokemi.
git"
```

Simple beregninger

Vi kan bruge Jupyter notebooks som en lommeregner med alle de operationer vi er vant til fra en almindelig lommeregner.

```
1 + 1       # Addition med +
3 - 1       # Substraktion med -
4 * 4       # Multiplikation med *
20 / 5      # Division med /
```

Opgave 1

Brug Jupyter notebook til at beregne følgende:

1. $21 + 21$
2. $53 - 11$
3. 6×7
4. $\frac{546}{13}$

```
# Skriv din kode i denne celle
```

Variable

Som også nævnt i "Introduktion til livets Molekyler" er det oftest hjælpsomt at definere størrelser i variabler, hvilket giver dem et navn der kan bruges senere. En variable defineres ved at bruge =, f.eks. kan vi definere

```
a = 21 + 21
```

Som gør at resultat af beregningen er gemt i `a`.

Opgave 2

Gentag beregningerne fra opgave 1, men gem nu hvert del i en variable

```
a = 21 + 21
b = ... # Erstat ... med din kode
# Lav variablerne c og d.
```

Vi kan så bruge de definerede variable i en ny beregning

```
total = (a + b + c + d) / 4
print(total)
```

```
42.0
```

Opgave 3

I "Introduktion til livets molekyler" har vi lavet beregninger om antallet af celler i forskellige dele af kroppen.

```
total_cells = 37_000_000_000_000
brain_cells = 86_000_000_000
liver_cells = 240_000_000_000
skin_cells = 1_600_000_000_000
```

Benyt Jupyter som lommeregner til at beregne procent-delen hver kategori udgør af det total antal celler i kroppen, altså for hver kategori beregn

$$P = \frac{\text{Antal}}{\text{Total}} \times 100$$

```
brain_pct = ... # Erstat ... med din kode.
liver_pct = ... # Erstat ... med din kode.
# Beregn og lav variable for skin_pct
```

```
print(brain_pct)
print(liver_pct)
print(skin_pct)
```

```
0.23243243243243245
0.6486486486486486
4.324324324324325
```

Funktioner

Når vi laver den samme beregning flere gange er det smart ikke at skulle skrive det samme kode igen, det gør koden mere læse-venlig og reducere chancen for at vi laver fejl. Måden at undgå dette er ved at bruge **funktioner**, en funktion i Python definere en opskrift når den bliver kaldt.

Ovenfor har vi f.eks. brugt funktionen `print` til at skrive output. En funktion defineres sådan

```
def min_funktion(a, b):
    resultat = a * b
    return resultat
```

Det er vigtigt at funktionen er indenteret (skubbet til højre) da Python har behov for dette for at læse den. Når en funktionen defineres som ovenfor udføres beregningen ikke, ligesom at skrive en opskrift ned ikke laver et måltid, først når funktionen "kaldes" bliver den udført

```
min_funktion(6, 7)
```

```
42
```

Opgave 4

Færdiggør funktionen nedenfor så den beregner procent-delen af `værdi` af `total`.

```
def procentdel(værdi, total):
    resultat = ... # Erstat ... med din kode.
    return resultat
```

Vi kan så bruge funktionen til at beregne procent delene

```
brain_pct = procentdel(brain_cells, total_cells)
liver_pct = procentdel(liver_cells, total_cells)
skin_pct = procentdel(skin_cells, total_cells)
print(brain_pct, liver_pct, skin_pct)
```

```
0.23243243243243245 0.6486486486486486 4.324324324324325
```

Specielle funktioner

Vi har nu set at vi kan bruge Jupyter notebooks som lommeregner til at lave beregninger med de basale aritmetiske operationer. Vi har dog ofte behov for f.eks. at regne potensfunktioner, kvadratrødder, exponential funktioner eller logaritmer.

Heldigvis er dette også nemt med Python, vi kan nemlig bruge en pakke der implementere disse operationer som funktioner - specifikt `numpy`.

Potensfunktioner, altså x^k kan regnes som

- x**k

De tre andre kan regnes med NumPy som

- `np.sqrt`
- `np.exp`
- `np.log`

Opgave 5

Brug disse til at beregne følgende

- 6.4807^2
- $\sqrt{1764}$
- $e^{3.737669}$
- $\log(66.6863)$

```
import numpy as np # Dette importere NumPy så vi kan bruge funktionerne.

resultat_1 = ... # Din kode her
resultat_2 = ... # Din kode her
resultat_3 = ... # Din kode her

print(resultat_1, resultat_2, resultat_3)
```