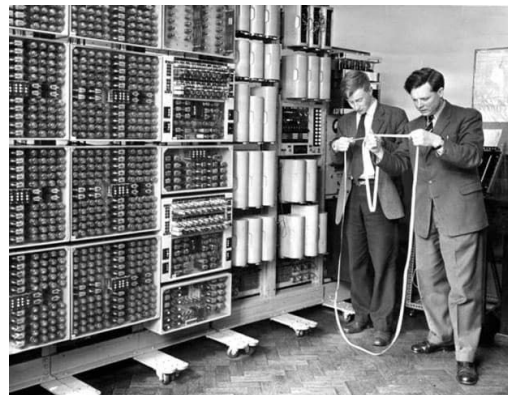


Programmering i Python

Jonas Andersen & Nikolaj Bjerregaard, UNF Naturfagssommercamp

Hvorfor er det smart at kunne programmere?

- Simulationer, problemløsning, AI, hjemmesider, videospil, etc...
- DATABASEHANDLING!!!!!!



Hvorfor er det smart at kunne Python?

- Højniveau programmeringssprog (ikke hvad det lyder som)
- Kan alt hvad man vil have det til (i hvert fald i starten)
- Gratis OG lettilgængeligt!!!!
 - Anaconda, PyCharm, the like

```
import numpy as np
import time

runtime = time.time()

from numpy.random import random, seed
def Ising(a):
    Is = np.zeros(a**2)
    for i in range(a**2):
        r = random()
        if r < 0.5:
            Is[i] = -1
        else:
            Is[i] = 1
    Is = Is.reshape((a,a))
    Is[ 0, :] = 0
    Is[-1, :] = 0
    Is[:, 0] = 0
    Is[:, -1] = 0
    return Is
```

```
import matplotlib.pyplot as plt

def visual(model0):
    fig, ax = plt.subplots()
    cax = ax.imshow(model0, cmap=plt.cm.get_cmap('Blues', 3))
    ax.set_title('Ising model')

    cbar = fig.colorbar(cax, ticks=[-0.66, 0, 0.66], orientation='horizontal')
    cbar.ax.set_xticklabels([r'$\downarrow$', '0', r'$\uparrow$'])

    plt.show()
    return
```

Hvad kan man med python?

- Regning

```
[4]: print(2 + 3, 2*3, 2/3, 2**3)
5 6 0.6666666666666666 8
```

- Variable

```
[8]: a = 2
b = 3
print(a + b)
5
```

- Inputs

```
[6]: a = int(input("Tal 1:"))
b = int(input("Tal 2:"))
print(a + b)
Tal 1: 3
Tal 2: 5
8
```

Lad os få noget python op og køre!

- Colaboratory:

<https://colab.research.google.com/>



“print”

Kode → [1]: `print("Hej Naturfagssommercamp")`

Output → Hej Naturfagssommercamp

[5]: `print("Jeg er Batman")`
`print("Jeg")`
`print("er")`
`print("Batman")`

Jeg er Batman
Jeg
er
Batman


Hvorfor “tekst” ?

Datatyper

Type	Står for	Forklaring	Eksempel
<code>int</code>	Integer	Helt tal	42
<code>float</code>	Floating point	Kommatal	3.14
<code>str</code>	String	Tekst	"Jeg er Batman"
<code>bool</code>	Boolean	Sandhedsværdi	True

Tabel 5.8: De grundlæggende typer i Python

"type()"



```
[10]: print(type(42))
      print(type(3.14))
      print(type("Jeg er Batman"))
      print(type(True))

<class 'int'>
<class 'float'>
<class 'str'>
<class 'bool'>
```

Regning

```
[11]: print(2 + 3)  
      print(3 - 2)  
      print(2 * 3)  
      print(3 / 2)
```

5
1
6
1.5

```
[12]: print(2 + 3)
```

5

VS

```
[13]: print("2 + 3")
```

2 + 3



Regning (med tekst)

Tilladt:

- `str + str`
- `str * int`

```
[14]: print("Bat" + "man")
```

Batman

```
[15]: print("Batman"*3)
```

BatmanBatmanBatman

Ikke tilladt:

- Det meste andet...
- f.eks. `str - str`

```
[18]: print("Batman" - "man")
```

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-18-f8fdc5e43288> in <module>  
----> 1 print("Batman" - "man")  
  
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

Variable

Vi kan gemme værdier

```
[19]: a = "Batman"  
      b = 3  
      print(a * b)
```

BatmanBatmanBatman

Vi kan ændre værdier

```
[20]: a = 4  
      a = a + 2  
      print(a)
```

6



Inputs

```
[*]: tal = input()  
print(tal)
```

```
[*]: tal = input()  
print(tal)
```

```
[22]: tal = input()  
print(tal)
```

5
5



Inputs

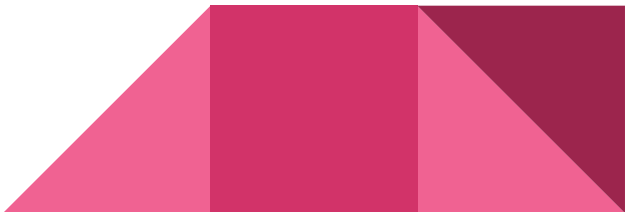
Lad os lave et program som tager et tal og lægger 1 til

```
[25]: tal = input("Skriv et tal:")  
      print("Det næste tal er:", tal + 1)
```

Skriv et tal: 5

```
-----  
TypeError                                 Traceback (most recent call last)  
<ipython-input-25-e0528fee36ce> in <module>  
      1 tal = input("Skriv et tal:")  
----> 2 print("Det næste tal er:", tal + 1)  
  
TypeError: can only concatenate str (not "int") to str
```

Problem!!!!



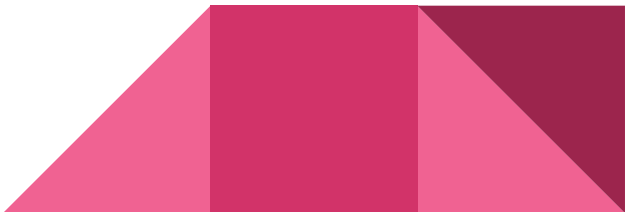
Inputs

Inputs er str type, men vi kan omdanne det til int !

```
[26]: tal = input("Skriv et tal:")  
      tal = int(tal)  
      print("Det næste tal er:", tal + 1)
```

Skriv et tal: 5

Det næste tal er: 6



Sammenligninger

Tager 2 objekter (ofte tal) og giver en boolean (sandt eller falsk)

```
[27]: a = 3
      b = 2
      print( a > b )
      print( a < b )

True
False
```

```
[28]: "Batman" == "Batman"
```

```
[28]: True
```

Python symbol	Matematisk symbol	Forklaring	Eksempel
>	>	Større end	4 > 2
>=	≥	Større end eller lig med	4 >= 4
<	<	Mindre end	2 < 4
<=	≤	Mindre end eller lig med	2 <= 4
==	=	Lig med	3 == 3
!=	≠	Ikke lig med	2 != 4

Tabel 5.9: Sammenligningsoperatorer i Python

If - statements (betinget udførsel)

Kører noget kode *hvis* en betingelse er opfyldt

```
[31]: trussel = 4
      if trussel >= 6:
          print("Trusselsniveau er højt!")
```

Lig mærke til at denne kode ikke giver noget output!

```
[30]: trussel = 4
      if trussel >= 6:
          print("Trusselsniveau er højt!")
      else:
          print("Trusselsniveau er lavt.")
```

Trusselsniveau er lavt.



Opgave!

Lav en quiz-maskine!

- Skriv et program som stiller et spørgsmål, tager et input og kan se om det er rigtigt eller ej



Flere betingelser!

```
[32]: trussel = 4
      if trussel >= 6:
          print("Trusselsniveau er højt!")
      elif trussel > 2:
          print("Trusselsniveau er mellemt.")
      else:
          print("Trusselsniveau er lavt.")
```

Trusselsniveau er mellemt.



Løkker

Vi vil gerne køre den samme kode flere gange:
for-loop!

```
[33]: for i in range(4):  
      print("Hej")
```

```
Hej  
Hej  
Hej  
Hej
```

```
[34]: for i in range(4):  
      print(i)
```

```
0  
1  
2  
3
```



Loops + if-statements

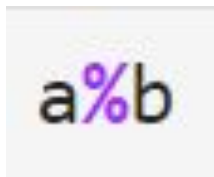
```
[36]: trussel = 1
      for i in range(4):
          if trussel >= 6:
              print("Trusselsniveau er højt!")
          elif trussel > 2:
              print("Trusselsniveau er mellemt.")
          else:
              print("Trusselsniveau er lavt.")
          trussel = trussel + 2
```

```
Trusselsniveau er lavt.
Trusselsniveau er mellemt.
Trusselsniveau er mellemt.
Trusselsniveau er højt!
```



Opgave!

- Skriv et program som finder ud af om et tal kan deles med 10 og kød det for alle tal mellem 1 og 100
- Følgende funktion kan være vigtig:



Hvis a kan divideres med b vil $a \% b = 0$



Funktioner

Hvad putter vi ind?

```
[76]: def f(x):  
      y = 3*x + 5  
      return y
```

Hvad kommer der ud?

```
[65]: print(f(1))
```

8

```
[64]: print(f(5))
```

20

Funktioner

```
[66]: for i in range(10):  
      print(f(i))
```

5
8
11
14
17
20
23
26
29
32

fremfor

```
[67]: for i in range(10):  
      print(3*i + 5)
```

5
8
11
14
17
20
23
26
29
32



Pakker


Giver os flere funktioner at lege med

- Numpy (har ting som pi og gør det nemt at regne ting)
- Matplotlib.pyplot (tillader os at tegne grafer)
- SciPy, Random, etc.

```
[9]: import numpy as np  
import matplotlib.pyplot as plt
```

```
[69]: print(np.pi)
```

```
3.141592653589793
```



Arrays

En måde at gemme en masse ting i én variabel!

```
[9]: a = np.array([1,2,3,4])  
     print('a[2] = ', a[2])  
a[2] = 3
```

Man kan regne på hele arrays!

```
[71]: print(a + 2)  
[3 4 5 6]
```



np.linspace(a, b, c)

Laver et array med c elementer af tal fra a til b

```
[72]: x = np.linspace(1,10, 10)  
print(x)
```

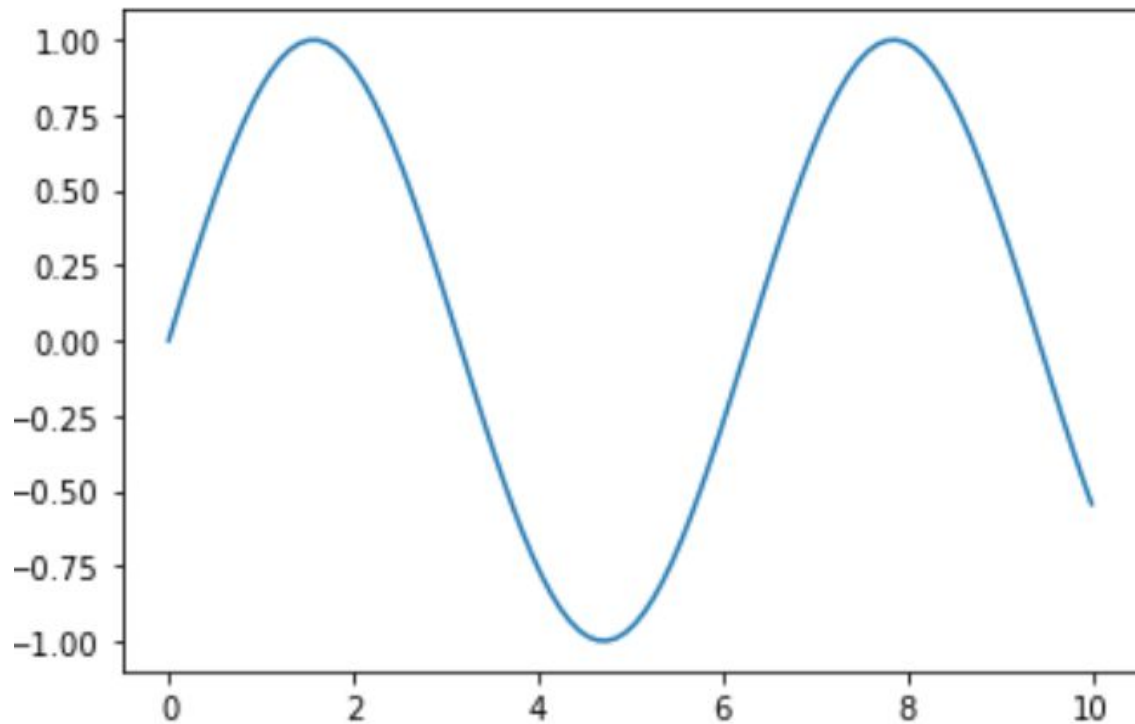
```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```

```
np.linspace(start, slut, antal indelinger)
```

```
[75]: x = np.linspace(1,10, 1000)  
print(x)
```

```
[ 1.          1.00900901  1.01801802  1.02702703  1.03603604  1.04504505  
 1.05405405  1.06306306  1.07207207  1.08108108  1.09009009  1.0990991  
 1.10810811  1.11711712  1.12612613  1.13513514  1.14414414  1.15315315  
 1.16216216  1.17117117  1.18018018  1.18918919  1.1981982   1.20720721  
 1.21621622  1.22522523  1.23423423  1.24324324  1.25225225  1.26126126  
 1.27027027  1.27927928  1.28828829  1.2972973   1.30630631  1.31531532  
 1.32432432  1.33333333  1.34234234  1.35135135  1.36036036  1.36936937  
 1.37837838  1.38738739  1.3963964   1.40540541  1.41441441  1.42342342  
 1.43243243  1.44144144  1.45045045  1.45945946  1.46846847  1.47747748  
 1.48648649  1.4954955   1.5045045   1.51351351  1.52252252  1.53153153  
 1.54054054  1.54954955  1.55855856  1.56756757  1.57657658  1.58558559  
 1.59459459  1.6036036   1.61261261  1.62162162  1.63063063  1.63963964  
 1.64864865  1.65765766  1.66666667  1.67567568  1.68468468  1.69369369  
 1.7027027   1.71171171  1.72072072  1.72972973  1.73873874  1.74774775
```

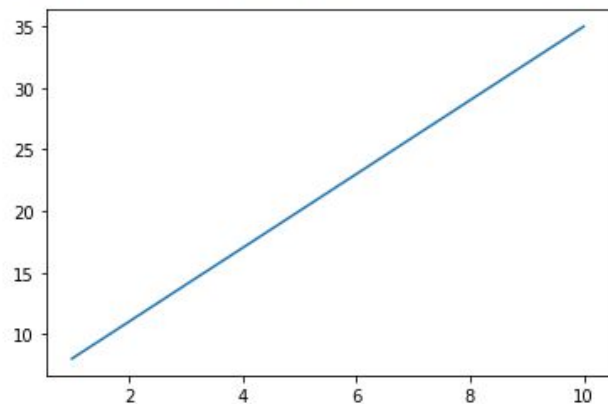
Lad os bruge hvad vi ved til at lave en graf!



plt.plot()

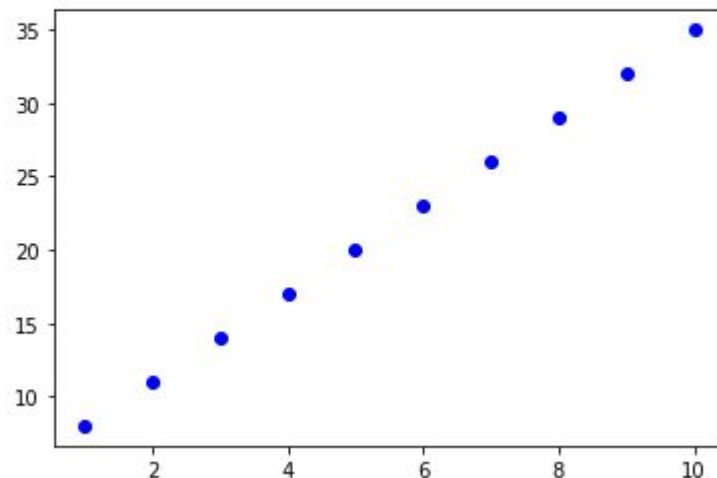
```
[77]: x = np.linspace(1,10,10)
      def f(a):
          b = 3*a + 5
          return b
      y = f(x)
      plt.plot(x,y)
```

```
[77]: [<matplotlib.lines.Line2D at 0x7f7a89da09d0>]
```



```
[79]: plt.plot(x,y, 'ob')
```

```
[79]: [<matplotlib.lines.Line2D at 0x7f7a8788f6d0>]
```



Så er det din tur!

- Skriv et program som laver et plot!
- Vigtige funktioner:

```
import matplotlib.pyplot as plt  
import numpy as np
```

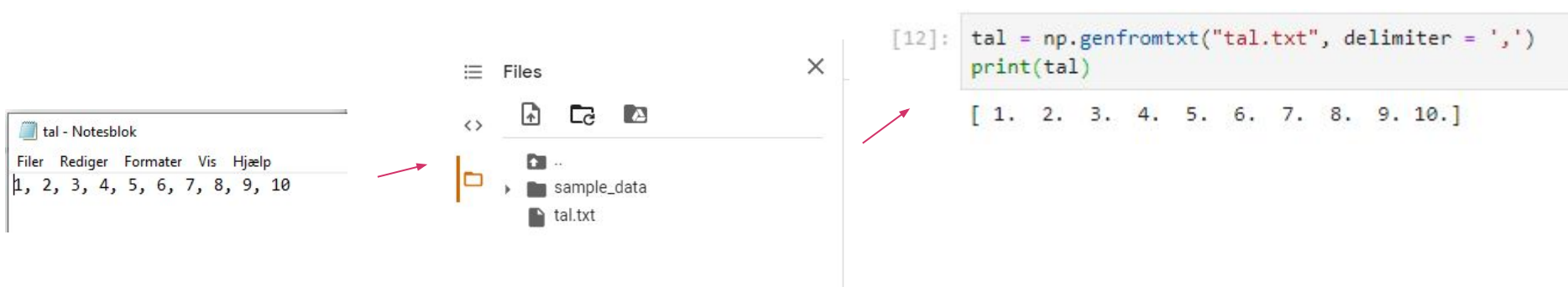
```
np.linspace(start, slut, antal indelinger)
```

```
plt.figure(0)  
plt.plot(x, y)  
plt.xlim(0,2)  
plt.ylim(-1,1)  
plt.show()
```



Importerer af data

- `np.genfromtxt()` kan tage en fil og importere den i python



The screenshot illustrates the process of importing data from a file into a Python environment. On the left, a text editor window titled 'tal - Notesblok' contains the numbers '1, 2, 3, 4, 5, 6, 7, 8, 9, 10'. A red arrow points from this text to a file explorer window titled 'Files'. The file explorer shows a directory structure with a folder named 'sample_data' and a file named 'tal.txt'. Another red arrow points from 'tal.txt' to a Python console window on the right. The console shows the execution of the following code:

```
[12]: tal = np.genfromtxt("tal.txt", delimiter = ',')
      print(tal)
```

The output of the code is displayed below the prompt:

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
```

Lad os prøve med noget ægte data

- <https://jonan15.github.io/Python-intro/>

Naturfags sommer camp

Datasets

- [stjerner_x.csv](#)
- [stjerner_y.csv](#)
- [stjerner_par.csv](#)



Opgaver til stjerner!

1. Lav en graf som viser alle stjernernes position
2. Beregn afstanden fra hver stjerne til jorden
3. Beregn den gennemsnitlige afstand af stjernehopet til jorden
4. Find stjernen længst væk samt tættest på og markér dem på plottet

