0.1 Python

Merk: Denne teksten bygger på teksten om Python i AM1.

0.1.1 NumPy

I Python kan man importere det som kalles **bibliotek** for å få tilgang til enda flere typer objekter, funksjoner og liknende. NumPy er et bibliotek som inneholder typen `numpy.ndarray`. Denne typen har mange fellestrekk med en liste, men skiller seg ut ved at den inneholder et bestemt antall elemener. Dette gjør blant annet at prossesser som bruker NumPy-arrays istedenfor lister går raskere, og at NumPy-arrays egner seg bedre til regneoperasjoner.

For å lage NumPy-arrays må vi importere NumPy-biblioteket:

```
1  import numpy as np
2
3  a = np.array([1, 2]) # lager array fra en liste
4  b = np.arange(4) # samme som å skrive np.array(range
       (4))
5  c = np.zeros(3) # array med 3 elementer lik 0
6  d = np.linspace(2,11,4) # array med 4 elementer.
                           d[0] = 2 og d[3] = 11.
                         Naboelement har lik differanse
7
8  print(a)
9  print(b)
10 print(c)
11 print(d)
```

Utdata

```
[1 2]
[0 1 2 3]
[0. 0. 0.]
[ 2. 5. 8. 11.]
```

Merk

Til forskjell fra lister, er elementene skilt bare med mellomrom når de printes.

Klassisek regnearter

Regneoperasjoner mellom NumPy-arrays blir utført elementvis:

```python
import numpy as np

a = np.array([10, 20])
b = np.array([2, 4])

print(a+b) # [10+2 20+4]
print(a*b) # [10*2 20*4]
```

Utdata

```
[12 24]
[20 80]
```

Vektoroperasjoner

NumPy-arrays fungerer ypperlig til å representere vektorer, og har innebygde metoder for å finne skalarprodukt, kryssprodukt og determinanter:

```python
import numpy as np

a = np.array([2, -7])
b = np.array([1, 5])
c = np.array([2, -7, 1])
d = np.array([1, 5, 0])

print(a.dot(b)) # skalarprodukt av a og b
print(np.cross(c, d)) # kryssproduktet av c og d

ab = np.array([a, b])
print(np.linalg.det(ab)) # det(a,b)
```

Utdata

```
-33
[-5 1 17]
17.0
```