External libraries

BB1000 Programming in Python KTH

Essential Python libraries

- NumPy: 'Numerical python' - package for scientific computing in Python. Ment primarily to sort, reshape, and index array types…
- SciPy: functions like numerical integration, linear algebra routinges,… Numpy and SciPy are often used together; the difference is not sharp; newer functionalities might reside in SciPy.
- pandas: data structures and functions to work with structured data. The main object in pandas is the `DataFrame`, which is a two-dimentional tabular.
- matplotlib: producing plots; the basic functions handled in this course are all in the matplotlib.pyplot module.

```
>>> import pandas as pd
>>> import numpy as np
>>> import scipy as sp
>>> import matplotlib.pyplot as plt
```

Pandas - Baby names 1880-2015

On http://www.ssa.gov/oact/babynames/limits.html the total number of births for each gender/name combination is given as a raw archive.

```
Mary,F,7065
Anna,F,2604
Emma,F,2003
Elizabeth,F,1939
Minnie,F,1746
```

Since this is a comma-separated form, use is made of `pandas.read_csv` to load the data

```
import pandas as pd
names1880 = pd.read_csv('names/yob1880.txt', names=['name', 'sex', 'births'])
```

The data are printed as

```
>>> print(names1880)
         name sex  births
0        Mary   F    7065
1        Anna   F    2604
2        Emma   F    2003
3        Elizabeth   F    1939
4        Minnie   F    1746
...
1996     Worthy   M       5
1997     Wright   M       5
1998       York   M       5
1999  Zachariah   M       5

[2000 rows x 3 columns]
```

To get an overview over all births, we can use the sum of the births by sex:

```
>>> names1880.groupby('sex')['births'].sum()
sex
F     90992
M    110490
Name: births, dtype: int64
```

Pandas - Excel

On the internet, Kaningentix finds an excel sheet containing all herbs, grasses and vegetables which can be found in the forest. The list contains not only the names and the subsequent characterizations, but also where these are found and the time of the medicinal effect.

It is advisable to use pandas, making use of the ExcelFile class.

```
>>> import pandas as pd
>>> xls_file = pd.ExcelFile('data.xls')
```

Data stored in a sheet can then be read into DataFrame using parse:

```
>>> table = xls_file.parse('sheet1')
```

NumPy

One of the key fatures of NumPy is its N-dimensional array object: `ndarray`. They enable to perform mathematical operations on blocks of data.

```
>>> import numpy as np
>>> data1 = [6, 7.5, 8, 0, 1]
>>> arr1 = np.array(data1)

>>> print(arr1)
[ 6.   7.5  8.   0.   1. ]

>>> data2 = [[1, 2, 3, 4], [ 5, 6, 7, 8]]
>>> arr2 = np.array(data2)

>>> print(arr2)
[[1 2 3 4]
 [5 6 7 8]]

>>> print(arr2.ndim)
2
>>> print(arr2.shape)
(2, 4)
```

NumPy - Default arrays

```
>>> print(np.zeros(10))
[ 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.]

>>> print(np.zeros((3,6)))
[[ 0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.]]

>>> print(np.arange(15))
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
```

Remark that for higher dimensional arrays, we have used tuples.

```
>>> print(np.empty((2,3,2)))
[[[  2.35558336e-310   2.02731498e-316]
  [  2.35558575e-310   2.35558575e-310]
  [  2.35558575e-310   2.35558575e-310]]

 [[  2.35558575e-310   2.35558575e-310]
  [  2.35558575e-310   2.35558575e-310]
  [  2.35558575e-310   2.42092166e-322]]]

>>> print(np.eye(3))
[[ 1.  0.  0.]
 [ 0.  1.  0.]
 [ 0.  0.  1.]]
```

`empty` creates an array without initializing its values to any particular value. It does the ideal recipe to return garbage…

NumPy - Operations between arrays and scalars

```
>>> arr = np.array([[1., 2., 3.], [4., 5., 6.]])
>>> print(arr*arr)
[[  1.   4.   9.]
 [ 16.  25.  36.]]

>>> print(arr-arr)
[[ 0.  0.  0.]
 [ 0.  0.  0.]]

>>> print(1/arr)
[[ 1.          0.5         0.33333333]
 [ 0.25        0.2         0.16666667]]

>>> print(arr**0.5)
[[ 1.          1.41421356  1.73205081]
 [ 2.          2.23606798  2.44948974]]
```

NumPy - Basic indexing and slicing

```
>>> arr= np.arange(10)
>>> print(arr[5])
5
>>> print(arr[5:8])
[5 6 7]
>>> arr[5:8] = 12
>>> print(arr)
[ 0  1  2  3  4 12 12 12  8  9]

>>> arr_slice = arr[5:8]
>>> arr_slice[:] = 64
>>> print(arr)
[ 0  1  2  3  4 64 64 64  8  9]
```

```
>>> arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
>>> print(arr2d[2])
[7 8 9]
>>> print(arr2d[0][2])
3
```