```
#####################
# PYTHON INTRO      #
#                   #
# 2009-12-08 Moritz #
#####################

# Documtentaion
# * http://docs.python.org/tutorial/index.html
# * http://docs.scipy.org
# * http://www.obspy.org

# Start Python interactively
# Starting ipython in a windows cmd consolue with -pylab option:
# /Start/Run type cmd /t:F0
# ipython -pylab


#####################
# Basic Arithmetics #
#####################

2+5*7

5/3    #1, integer devision

5.0/3  #1.66666 correct

5./3+2

5./(3+2)

5./3**2

(5./3)**2

5.**18

5./0   #ZeroDivisionError

0./0   #ZeroDivisionError

4.0+3j #Complex numbers


############################
# Basic Data Types/Objects #
############################

# Intergers, floats, strings
a = 3
a
b = 3.56
b
c = 'Hello World'
c
print a, b, c, type(a), type(b), type(c)

# List
# Lists can contain all other data types, even functions list itselfs,...
# Index in Python starts with 0
d = []
e = [3, 4, 5]
f = ['hello','world','foo','bar']
e[0] #first index of list, here 3
e. #tabtab shows all availble methods
a = range(5) #returns list of [0,1,2,3,4]

# Tuples
# tuples are not oven used, similar to lists
x = (1,2,3,4)
```

T2: Introduction to Python, NumPy and ObsPy

```
x[0] #1 index

# Dictionary
# Dictionaries allow other datatypes than int as index.
g = {'Hello': 100, 'World': -50}
g['Hello'] #results in 100
g['World'] #results in -50

# Find out the type
type(a)
type(g)

#########################
# Functions and Packages #
#########################

# Import module
x = 5
import numpy
numpy.exp(5)
numpy.sqrt(1+0j)

# Import function from module
from numpy import exp
exp(5)

# Import as other name
import numpy as np
np.exp(5)

# Import all functions from module, not allowed on module level
# In the exercises I do not do this such that you know from which package
# which function is used.
from numpy import *

################
# Getting help #
################

# Help
a = ['Hello', 'World']
help(a)

# Help on module
import numpy
help(numpy)
help(numpy.exp)

# List available methods in object or module
dir(numpy)
dir(a)

################
# Flow control #
################

# Python has no brackets, all is controlled by the indentation. Space or
# tabulator matters ==> use space
for i in range(5): #==[0,1,2,3,4]
    print i

if 'a' == 'a':
    print 'a=a'

try:
    z[0] = 'First index'
except IndexError:
    print "A list and index must be initialized before assigning it"

######################
```

```
# IPython essentials #
######################

#Starty ipython with -pylab option which
#preloads most numpy and plotting modules
# ipython -pylab

# Run Python program
run program.py

# Run Python program, in local namespace
run -i program.py

# Tab completions
import numpy
numpy.   #hit tab twice
numpy.ex #hit tab once

# Help
help math
math?  #same as help
math?? #show also source code

# History
#arrow up ---> show previous command in history
#arrow down -> show next command in history
num #arrow up -> show previous command starting with num
history # -> print the complete recorded history

# Basic Unix commands are mapped
ls  #list directory entries
mv file1 file2 #move file1 to file2
cp file1 file2 #copy file1 to file2

#################
# 1 Dim Arrays #
#################

# Arrays in python are efficiently handled with the
# numpy module.
import numpy as np

# Allocating 1 Dim numpy array
N = 5 #size 5
x0 = np.arange(0, 10, 0.1) # sequence from 0 to 10 with step 0.1
x1 = np.zeros(N)
x2 = np.ones(N)
x3 = np.random.randn(N)
x4 = np.random.rand(N)
x5 = np.array([1,2,3,4]) # by converting from a list, inefficient

# Caution, type of arrays is important
y1 = np.ones(N, dtype='int32')
y3 = y1.copy()
y2 = np.ones(N, dtype='float64')
# For inplace multiplication, the type stays the same!
y1 *= 0.5
y2 *= 0.5
# For normal multiplication, the type changes
y3 = y3 * 0.5

# example of methods bound to the arrays
x0.max()
x0.min()
x0.std()
x0.mean()
x1 = x0.copy()
x0 -= x0.mean() # subtrace mean from e.g. time series

# slicing
```



```
x = np.random.random(1000)
x[:100] #first 100th entries
x[100:] #last 100th entries
x[100:800] #100th to 800th entry

####################
# Multi Dim Arrays #
####################

# http://numpy.scipy.org
# Allocating multi Dim numpy array
N = 5
M = 3
x5 = np.zeros((N,M))   # NxM array
x6 = np.zeros((N,M,N)) # NxMxN array
x7 = np.arange(15).reshape(N,M)

# dot product
dot = np.dot(x5.T, x5)  #.T == transpose
# outer product
outer = np.dot(x5, x5.T)

# finding the shape
shape = x5.shape # shape[0] = N, shape[1] = M

# slicing (index start with 0!!)
x7[1,:] # second row
x7[0:2,:] # first three rows
x7[:,1:3] # second and third coloumn
tind = np.array([0,1,2]).reshape((-1,1))
xind = np.array([1,2]),reshape((1,-1))
x7[tind,xind] # second and third entry in the first three rows

##################
# Linear Algebra #
##################

import numpy.linalg as la
X = np.array([[1,1],[1,-1]]) #inverse
la.inv(X)
X = np.array([[1,2],[2,1]]) #eigenvalues, vectors
val, vec = la.eig(X)

####################
# Plotting Arrays #
####################

# * All 2D plotting routines are efficiently handled with the
#   matplotlib module
# * Gallery with different plots, click on them for seeing the source code
#   http://matplotlib.sourceforge.net/gallery.html
import matplotlib.pyplot as plt
import numpy as np

# Plot random series of y values
x3 = np.random.randn(N)
plt.figure()
plt.plot(x3)
plt.show() # you do not need this in an ipython shell

# Plot x agains y values
x4 = np.random.randn(N)
t = np.arange(len(x4)) #start value and interval are optional
plt.figure()
plt.plot(t, x4, 'r--') #red dotted lines
plt.plot(t, x3, 'gx-') #green lines with x markers in overlay
plt.show() #you do not need this in an ipython shell

# Clear current figure plot and plot histogram
plt.clf() # clear figure
```

```
plt.close('all') # close all figures
plt.hist(x4) #plot historgram

###############
# Seismology #
###############

# Basic seismology routines are already implemented in the
# obspy module www.obspy.org
import obspy

# Read in SAC, MSEED or GSE2
st = obspy.read("data/BW.RJOB..EHZ.D.2008.107") #read in stream
tr = st[0] #first trace in stream, trace consits data block that is no gap
tr.stats #contains all the meta/header information
tr.stats.gse2 #contains gse2 specific meta/header
tr.data #contains data as numpy array, C contiguous memory layout

# Plotting
st.plot() # fast view, no control
import matplotlib.pyplot as plt
import numpy as np
plt.figure()
npts, df = tr.stats.npts, tr.stats.sampling_rate
plt.plot( np.arange(npts)/df, tr.data)
plt.title(tr.stats.starttime)

# Write out SAC, MSEED or GSE2. Note only the header entries network,
# station, location, starttime, endtime, sampling_rate, npts, channel are conver
ted!
st.write("myfile.sac", format='SAC')

##############
# notepad++ #
##############

# Edit/Create/Modify Python programs with the notepad++ editor
#
# Crutial setting for notepad++
# * /Settings/Preferences/Language Menu/ x Replace by Space
# * /View/Show Symbols/Whitespace and Tab


############################
# Conversion to MAT files #
############################

from scipy.io import loadmat, savemat
mat = loadmat("matfile.mat")
savemat("matfile.mat", {'a':a, 'b':b}) # save variable a and b to matfile
```