Jonas van den Brink, jvbrink

j.v.d.brink@fys.uio.no

8.1—Assignment and in-place modifications of NumPy arrays

The code does not work as intended because the assignments `y = x` and `z = x` produces *shallow copies* of the NumPy-array `x`. This effectively means that the variables `y` and `z` point to the same array in memory as `x`, so changing one of them, will change all of them simultaneously.

To fix this, we should instead use the `npdarray`-method `copy()`, which returns a deep copy of the array. So we have the code

```
from scitools.numpytools import *
x = sequence(0, 1, 0.5)
y = x.copy(); y *=2; y += 1
z = x.copy(); z *= 4;
print x, y, z
```

This is equivalent to the syntax

```
from scitools.numpytools import *
x = sequence(0, 1, 0.5)
y = 2*x + 1
z = 4*x - 4
print x, y, z
```

Both of these programs produce the output

```
[ 0.   0.5  1. ] [ 1.  2.  3.] [-4. -2.  0.]
```

8.2—Process comma-separated numbers in a file

The exercise asked for a compact script, so readability has not been emphasized

```
from numpy import array, sum

def process_spreadsheet(infile):
    # Read the inputfile
    with open(infile, 'r') as f: data = f.readlines()
    # Extract the names from the first column
    names = [l.split(',')[0] for l in data]
    # Extract the numbers and sum them
    nums = sum(array([l.split(',')[1:] for l in data], dtype='float'), axis=1)
    # Print the results
    print "\n".join([names[i]+" : "+str(nums[i]) for i in range(nums.size)])

if __name__ == '__main__':
    # Test the function on the example given in the exercise text
    process_spreadsheet('example.dat')

'''
user$ python process_spreadsheet.py
"activity 1" : 2719.0
"activity 2" : 128.0
"activity 3" : 365.5
'''
```

8.3—Matrix-vector multiply with NumPy arrays

```
import numpy as np

A = np.array([[1, 2, 3], [4, 5, 7], [6, 8, 10]], float)
b = np.array([-3, -2, -1], float)

print "A*b gives wrong result: \n", A*b
print "np.dot(A,b) gives right result: \n", np.dot(A,b)
print "A.dot(b) gives right result: \n", A.dot(b)

'''
user$ python matvec.py
A*b gives wrong result:
[[ -3.   -4.   -3.]
 [-12.  -10.   -7.]
 [-18.  -16.  -10.]]
np.dot(A,b) gives right result:
[-10.  -29.  -44.]
A.dot(b) gives right result:
[-10.  -29.  -44.]
'''
```

8.4—Replace lists by NumPy arrays

Simply changing lists to NumPy-arrays makes little difference. I instead use the function `np.genfromtxt` to generate a NumPy-array that behaves much like a dictionary.

```
import numpy as np
import sys

usage = 'Usage: %s infile' % sys.argv[0]
try:
    infilename = sys.argv[1]
except:
    print usage; sys.exit(1)

with open(infilename, 'r') as f:
    f.readline(); dt = float(f.readline()) # Read header and dt
    # Read in data from file, using first line as names
    data = np.genfromtxt(f, names=True, delimiter=" ", dtype=None)

# Write out 2-column files with t and data[name] for each name
for name in data.dtype.names:
    with open(name+'.dat', 'w') as ofile:
        for k in range(data[name].size):
            ofile.write('%12g %12.5e\n' % (k*dt, data[name][k]))


'''
user$ ls
convert2_wNumPy.py  hmt.out
user$ python convert2_wNumPy hmt.out
user$ ls
c0.dat   CaJSR.dat  cCa3.dat            j.dat        pc2.dat  xto1.dat
c1.dat   CaNSR.dat  cCa4.dat            Ki.dat       po1.dat  yL.dat
c2.dat   Cass.dat   convert2_wNumPy.py  LTRPNCa.dat  po2.dat  yto1.dat
c3.dat   cCa0.dat   h.dat               m.dat        V.dat    z_b.dat
c4.dat   cCa1.dat   hmt.out             open.dat     xKr.dat
Cai.dat  cCa2.dat   HTRPNCa.dat         pc1.dat      xKs.dat
user$ less m.dat
           0  9.97681e-01
         0.5  9.97285e-01
           1  9.96574e-01
         1.5  9.95633e-01
           2  9.94415e-01
         2.5  9.92980e-01
           3  9.91259e-01
         3.5  9.89217e-01
           4  9.86821e-01
        ...
'''
```

8.5—Rock, Paper, Scissors

```
from random import randint
import sys

def ask(question):
    print question,
    answer = sys.stdin.readline()
    # Abort on ctrl+d
    if answer == '' : sys.exit()
    print ""
    return answer

def win():
    print "Human: %7s\tComputer: %7s\tHuman wins!" % (n[human], n[comp])
    wins[0] += 1
    print "Score: Human %d \t Computer %d\n" % (wins[0], wins[1])

def draw():
    print "Human: %7s\tComputer: %7s\tA draw" % (n[human], n[comp])
    print "Score: Human %d \t Computer %d\n" % (wins[0], wins[1])

def loss():
    print "Human: %7s\tComputer: %7s\tComputer wins!" % (n[human], n[comp])
    wins[1] += 1
    print "Score: Human %d \t Computer %d\n" % (wins[0], wins[1])

p = {'r\n':0, 's\n':1, 'p\n':2}
n = {0:'rock', 1:'scissors', 2:'paper'}
results = {-2:loss, -1:win, 0:draw, 1:loss, 2:win}
wins = [0, 0]

print "Welcome to rock, paper, scissors!\n"
while True:
    try:
        required_to_win = int(ask("How many points are required to win? "))
        break
    except ValueError:
        print "Sorry, I did not understand that number. Please try again."
while required_to_win not in wins:
    while True:
        try:
            human = p[ask("Choose (r)ock, (p)aper, or (s)cissors? ")]
            break
        except KeyError:
            print "Sorry, the only legal moves are r, p or s."
    comp = randint(0, 2)
    results[human-comp]()
print "Final Score: Human %d \t Computer %d" % (wins[0], wins[1])

'''
user$ python roshambo.py
Welcome to rock, paper, scissors!

How many points are required to win? 2

Choose (r)ock, (p)aper, or (s)cissors? r

Human:    rock  Computer:   paper   Computer wins!
Score: Human 0   Computer 1

Choose (r)ock, (p)aper, or (s)cissors? p

Human:   paper  Computer:   paper   A draw
Score: Human 0   Computer 1

Choose (r)ock, (p)aper, or (s)cissors? r

Human:    rock  Computer:   paper   Computer wins!
Score: Human 0   Computer 2

Final Score: Human 0     Computer 2
'''
```