Department of Mathematics and Computer Science
University of Southern Denmark, Odense

February 5, 2024
Marco Chiarandini

DM545/DM871 – Linear and integer programming

Sheet 0, Spring 2024 [pdf format]

Solution:
Included.

Exercise 1

Write a list of the first 100 numbers in which any number divisible by three is replaced by the word "fizz" and any divisible by five by the word "buzz". Numbers divisible by both become "fizz buzz".

Solution:

```
for num in xrange(1,100):
  if num % 5 == 0 or num % 3 == 0:
    print("fizzbuzz")
  elif num % 5 == 0:
    print("buzz")
  elif num % 3 == 0:
    print("fizz")
  else:
    print(num)
```

Exercise 2 Data Types

Revise the difference between the main data types in Python: list, tuples, dictionaries and sets. Write an example for each of them in which you define and initialize a variable for each type and then print the content looping through the elements of the variable.

Solution:

```
# lists
my_list = [10,20,30,40,50,60]
for i in my_list:
  print(i)

# tuples
my_tuple = (1,2,3,4,5,6,7,8,9)
for i in my_tuple:
  print(i)

# dictionaries
my_dict = {'name': 'Esau', 'age': 2, 'occupation': 'My dog'}
for key,val in my_dict.iteritems():
  print("My {} is {}".format(key,value))

# set
my_set = {20,30,40,50,60,20,30,40,50}
for i in my_set:
  print(i)
```

Exercise 3 Python: One liner quizzes

Write a one line Python code for the following tasks:

a) Construct the set $S = \{x \in \mathbb{R} \mid x \geq 0 \wedge x \bmod 3 \equiv 1\}$

Solution:

```
S = {x for x in range(50) if x % 3 == 1}
```

b) Using list comprehension make a list for $\{(i, j) \mid i \in \{1, 2, 3, 4\}, j \in \{5, 7, 9\}\}$

Solution:

```
[(i,j) for i in (range(4)+1) for j in [5,7,9]]
```

c) Calculate the inverse of a function or the index function for an invertible function (ie, bijective = injective + surjective) given in form of a dictionary.

Solution:

```
{d[k]:k for k in d}
{v:k for k in d.keys() for v in d.values}
{v:k for k,v in d.items()}
```

d) What is the result of the following lines?

```
map(lambda x: x%3, range(5))
filter(lambda x: x%2==0, range(5))
```

(In Python 3.x, you have to enclose those lines in the list constructor `list()`.)

Exercise 4 Matrix Calculus in basic Python

The basic data structures in Pyton are lists, tuples, sets and dictionaries. Vectors and matrices can be implemented in Python as lists. How?

a) Generate a couple of numerical examples for vectors and matrices. Experiment with the operators + and ∗. Do they yield the same result as expected from linear algebra?

b) Write a function for the sum of two vectors using list comprehension.

c) Write a function for the multiplication of a vector by a scalar.

d) Write a function for the sum of two matrices using list comprehension.

e) Write a function for the multiplication of a matrix by a scalar.

f) Write a function for the multiplication of two matrices not necessarily square. (Raise a ValueError exception if the size of the matrices is not compliant.)

Solution:

It is important to note that the operators + and ∗ are overloaded for list. They concatenate or replicate the two lists, respectively. This is definetly not what we learned to be the definition of those operations.

```
def sumVec(a,b): return [a[i]+b[i] for i in range(len(a))]
def multScalVec(alpha,a): return [alpha * a[i] for i in range(len(a))]


def sumMat(M,N)
  result = [[0 for x in range(len(N[0]))] for y in range(len(M))]
  for i in range(len(M)):
      for j in range(len(N[0])):
          result[i][j]=M[i][j]+N[i][j]
  print(result)


def mult(M,N):
    if (len(M[0])!=len(N))
        raise ValueError("The inner size of the martices does not match")

    result = [[0 for x in range(len(N[0]))] for y in range(len(M))]

    for i in range(len(M)):
        for j in range(len(N[0])):
            for k in range(len(N)):
                result[i][j] = result[i][j] + M[i][k] * N[k][j]

    return result

M=[[1,2,3],[1,2,3]]
N=[[1,2],[1,2],[1,2]]

try mult(M,N) except ValueError: print("Oops, a ValueError occurred")

def printMatrix(M):
    for row in M:
        print(["%3.0f" % a for a in row])
    print("\n")
```

Exercise 5 Matrix Calculus in `numpy` and `scipy`

The modules `numpy` and `scipy` make available another data structure in Python, the 'array' type. This exercise guides you to the discovery of how operators are overloaded for the 'array' type module. You can read more about `numpy` and `scipy` from the Tutorial linked above and from the section "Literature: Links" of the course web page.
Generate in Python two matrices A and B of size 3×2 and 2×4, respectively, made of integers numbers randomly drawn from the interval $[1, \ldots, 10]$. Calculate the following results, first by hand and then checking the correctness of your answer in Python:

a) $A + B, A - B$

b) $A \cdot B$

c) A/B

[In IPython and Jupyter it is possible from command line to ask for completion via tab. This can be used to explore which functions are available for a given module. Try for example to type

```
import numpy as np
np.
```

followed by a tab. You should see a list of available functions. Among them there are two submodules that will be useful for us: `random` and `linalg`. The first implements a function to generate random numbers and matrices. The second implements functions from linear algebra. It is possible to get a manual for each function by following the function with a question mark. For example: `np.random.randint?`.]

Solution:

```
A=np.random.randint(1,10,(3,2))
B=np.random.randint(1,10,(2,4))
A+B
A-B
A*B
```

All these produce an error because the +,- operators, as in linear algebra perform an element-wise addition and subtraction, which requires the two matrices to have exactly the same size.
The * operator performs also an element-wise multiplication, which require the two matrices to have exactly the same size. However, contrary to addition and multiplication, this operation is not defined element-wise in linear algebra.
The correct way to obtain the matrix product is:

```
np.dot(A,B)
```

The division by a matrix is not defined in linear algebra. In Python it does an element-wise operations if the matrices have the same size.

Exercise 6

Solve by Gaussian elimination the following system of linear equations $A\mathbf{x} = \mathbf{b}$ where

$$A = \begin{bmatrix} 3 & 1 & 1 & 1 \\ 2 & 4 & 1 & 1 \\ 2 & 1 & 2 & 2 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$$

First carry out the calculations by hand and then try using Python.

Solution:

```
A=np.array([[3,1,1,1],
  [2, 4, 1, 1],
  [2,1,2,2]])
b=np.array([ 2, 2, 1])

# np.linalg.solve(A,b)
print np.linalg.matrix_rank(A)
AA=np.column_stack([A,b])
AA=column_stack([A,b])
AA[0,:] = f(1,3) * AA[0,:] # remember indices start from 0
AA[1,:] = -2 * AA[0,:] + AA[1,:]
AA[2,:] = -2 * AA[0,:] + AA[2,:]
printm(AA)
AA[1,:] = f(3,10)* AA[1,:]
AA[2,:] += -f(1,3) * AA[1,:]
printm(AA)
AA[2,:] = f(10,13) * AA[2,:]
printm(AA)
AA[1,:] += -f(1,10) * AA[2,:]
AA[0,:] += -f(1,3) * AA[2,:]
printm(AA)
AA[0,:] += -f(1,3) * AA[1,:]
printm(AA)
```

After elementary row operations the augmented matrix looks like:

```
[1, 0, 0, 0, 9/13]
[0, 1, 0, 0, 3/13]
[0, 0, 1, 1, -4/13]
```

From here we can write directly the solution:

$$\begin{aligned} x_1 &= 9/13 \\ x_2 &= 3/13 \\ x_3 &= -4/13 - t \\ x_4 &= t \end{aligned}$$

which in vector notation is equivalent to

$$\mathbf{x} = \begin{bmatrix} 9/13 \\ 3/13 \\ -4/13 \\ 0 \end{bmatrix} + t \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix}$$

Hence the solution subspace is expressed by an affine combination.