Exercise 00

Introduction to Linux and Python

Deadline: No deadline for this exercise. It will not be graded.

0.1 Linux

In this exercise you can practice the basic Linux commands.

1. Open a terminal.
2. Identify the path of your home directory (*hint:* pwd). List all directories here (*hint:* ls).
3. Create a new directory (*hint:* mkdir) for your exercises, e.g. "MD_Exercises". Within this directory, create a subdirectory for the current exercise, e.g. "Ex00".
4. Create a text file and type in some text, e.g. your personal data, a list of linux commands you know etc (*hint:* vi).
5. Prompt the first line of the file. Prompt the last line. (*hint:* head, tail). Redirect the output to a new file (*hint:* >>).
6. Count the number of words in your file. How often have you used the letter "m"? (*hint: wc, grep*).
7. Create a backup directory. Copy your file into it.
8. Rename the backup file to <filename>_backup.txt (*hint:* mv).
9. Check who (user, group or others) has permission to read, write or execute the file. Change writing privileges, so that you can no longer modify the file (*hint:* chmod).
10. Delete the backup file. Delete the backup directory (*hint:* rm).
11. Create a .tar archive of the exercise folder (*hint:* tar).

0.2 Python

In this exercise you can get familiar with python. You can work with python interactively or write a script.

Calculations:

1. Assign two integer values to `int1` and `int2`, where `int2` is larger than `int1`, but smaller than two times `int1`.
2. Test, if the values fulfil the requirement, so that you get the output `True`.
3. Divide `int1` by `int2`.
4. Convert the values to floats `f1` and `f2`. Divide `f1` by `f2`. Try also integer division (//).

Arrays (don't confuse with lists):

1. Import the module numpy.
2. Create a numpy array `A` with 4 elements containing all zeros and create a variable holding the length of `A`.
3. Assign an integer value to each array element.
4. Using a for loop, calculate the sum over all array elements.
5. Calculate the same using a while loop.
6. Check your results using the built-in function sum() or numpy.sum().

Matrices:

1. Create a 4x4 Matrix with random values between 0 and 1 (with numpy.random.uniform).
2. Change all elements of the 2nd row to 1 and subsequently all elements of the 3rd column to 0 (with numpy.zeros and numpy.ones).
3. Count the number of values larger than 0.5 using a double for loop.
4. Check the result with indexing (numpy.where).

Functions/Plotting:

1. import matplotlib.pyplot.
2. Create two arrays `A` and `B` of 20 random integers between 0 and 10.
3. Write a function that reads an array as input and returns an array of same length containing in each element the sum of elements up to the current element (cumulative sum).
4. Use `A` and `B` as input for your function.
5. Plot the two new arrays with different colours. Add a legend to show which colour belongs to which array.

Histogram

1. Create an array with 100 random floating values between 0 and 30.
2. Discretise the data using a bin width of 2. The new array should contain values ranging from 0 to 15.
3. Create a histogram containing the respective number of times the value i occurred in the discretised array.
4. Plot the histogram using matplotlibs bar(). Compare the results with a histogram generated by matplotlibs histogram() function.

Distance matrix

1. Generate an array `P` of size $N \times 3$, with $N = 1000$ random numbers. Each row of the array represents the position of a point in 3D space.
2. Create a function that generates a symmetric matrix `d` of size $N \times N$, where each element `d[i,j]` is the euclidean distance between each pair of points (i,j):

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{1}$$

3. Calculate this distance matrix for `P` and measure the execution time (import time).
4. Measure the execution time that the scipy function cdist() needs for the same calculation.
5. Display the matrix as a matrix plot (e.g. with matplotlibs imshow()) and add a colorbar legend.

Reading and writing files

1. Open a new file and write the following lines using a for loop:
0
H
1
He
2
Hel
3
Hell
4
Hello
5
HelloW
6
HelloWo
7
HelloWor
8
HelloWorl
9
HelloWorld
10
HelloWorld!

2. Open the file again and read each line containing a number to save these in a list of integers. How would that work using numpys function loadtxt() or genfromtxt() instead?
3. Use the following function to create a 5×11 matrix of 5 shuffled versions of the list.

```
def shuffled(arr):
    lst = list(arr)
    random.shuffle(lst)
    return lst
```

4. Write this matrix to a new file.

User input

1. Use sys.argv() (import sys) to save a command line argument in a variable called `greet`.
2. Ask for the user's name. Then use the greeting saved in `greet` to greet the user with the name.
3. Ask the user for the current year and then print the next leap year. A year is a leap year, if it can be evenly divided by 4. It is not a leap year, if it can be evenly divided by 100, unless it is also evenly divided by 400.
4. Write a guessing game in which the user has to guess a secret, randomly generated number between 0 and 100. After a guess the program tells the user, if the number was to low or to high until the right one has been found.