In-Class Work: Random Walks

1. Copy into your working directory the sample python-file for random numbers

```
~kvollmay/share.dir/pythonsamples.dir/sample_rndnumbers.py
```

1a. Run the program and **scan** it to approximately know what the sample-program does. It is okay, if you do not yet understand what exactly the program does. The following steps guide you through the sample-program.

1b. Copy the python program to another file (e.g. `classrndwalk1b.py`), so that you can successively use the tools of the program. To play with the program locate the line with `#exit()`. Use this line without the comment symbol, so `exit()` to exit the program early. Move this line successively lower in the program to test each section of the program. Notice that you need at the top the command `import numpy as np`
Run the program. Change the program such that only 50 lines are written. To look at the resulting data you can either use python-plotting tools or `xmgrace` for which I will show to you several tools for making nice figures for your project results and talks in a class later in the course. Let's say your python-program is named `classrndwalk1b.py` and prints out two columns, then type in the commandline of a terminal:

```
./classrndwalk1b.py | xmgrace -pipe
```

Another way of plotting the data would have been to first write them into a file (instead of on the screen in the terminal window), let's say you want to write the data into the file named `dat1` you can do this with the command `./classrndwalk1b.py > dat1` . Have a look at the just created file `dat1`. Now we look at the data by plotting them using xmgrace with the command

```
xmgrace dat1
```

or you could have done two commands in the same commandline with

```
./classrndwalk1b.py > dat1 ; xmgrace dat1
```

1c. Run your python-program again, but this time write the data instead into the file `dat2`. Compare `dat1` and `dat2`. What is going on? How can you ensure to get instead each time when you run the program the same random numbers? (Reproducibility is crucial e.g. for program testing). Yes, this is the `np.random.seed(15)`. Put this line before the for-loop, i.e. before you use the random number generator. Use a different integer than `15`. Confirm that `dat1` and `dat2` are the same. For line by line comparison of two files you may use in the command-line the linux-command

```
diff dat1 dat2
```

outcome of this command are the differing lines, so when you use `diff` on two same files nothing is written on the screen.

1d. Now put the command `exit()` later in the program and run `sample_rndnumbers.py` again to understand what the other commands are doing.

1e. You find at the end of `sample_rndnumbers.py` between ''' and ''' commands you can use to make a figure with python. The quotes are to comment out a whole block. So move the quote lines together, i.e. uncomment, and see how the plotting works.

2. Random Walk in One Dimension

Next we will do important analysis on the random walk. To simplify the task (and yet being able to get the main concept) let us start with the random walk in one dimension.

2a. Write a program for a random walk in one dimension. In all following we assume $p = q = 0.5$, i.e. there is equal probability to jump to the right or to the left with jump size 1. Initialize with $x(t = 0) = 0$ (so `x=0`) and print t and x for each time step. Use `NSTEPS = 500` time steps. Look at the resulting $x(t)$ with `xmgrace` (or with python plotting).

2b. The next step is a preparation for 2c. Instead of printing every time step, print only once after $N =$`NSTEPS` time steps (i.e. after the time-loop) the resulting $x(N)$ and $(x(N))^2$. Increase `NSTEPS` to `NSTEPS=5000`.

2c. Now add a loop over simulation runs to your program from 2b. Each simulation run starts with $x = 0$ and gives you an $x($`NSTEPS`$)$ and an $x^2($`NSTEPS`$)$. First set `NSIMRUNS=10` and print for each simulation run $x($`NSTEPS`$)$ and $x^2($`NSTEPS`$)$. Next change your program such that it determines the average over `NSIMRUNS=10000` simulation runs of x and an x^2 and prints out the resulting averages $\langle x \rangle$ and $\langle x^2 \rangle$.

2d. Next no longer use the number of steps `NSTEPS` as constant but instead add a loop over $N=$`nsteps` to your program of 2c. Loop `nsteps` from 100 to 2000 in steps of 100. For each `nstep` print out $N=$`nstep`, $\langle x \rangle$ and $\langle x^2 \rangle$. Look at the resulting $\langle x(N)$ and also $\langle x^2(N) \rangle$. Please get me when you got this done, so that we can interpret your results with the class. Try if you can derive a theoretical prediction.

2e. In the following steps we will use `gnuplot` to fit the resulting $\langle x^2(N) \rangle$ simulation data. First save your data with `./classrndwalk2d.py > dat2d`, Then type the command

```
gnuplot
```

This will start a session in the graphics-tool `gnuplot`. To do a power law fit and to look at the comparison of the fitted line and your simulation data type

```
a=1;b=1;f(x)=a*x**b;fit f(x) "dat2d" using 1:3 via a,b;
plot "dat2d" using 1:3,f(x)
```