In-Class Work: Python — Lists, Arrays, User-Defined Functions, Good Programming

19. Traffic Model (application of §2.4 & §2.5)

Copy the file

`~kvollmay/share.dir/inclass2019.dir/road.dat`

into your working directory. The file contains 100 integers describing a road of 100 lattice sites at a certain time. `-1` means an empty site and a number ≥ 0 means a car with the corresponding speed in mi/h. (**Hint:** use from `~kvollmay/share.dir/pythonsamples.dir/sample\_arrays.py` the command `sp.loadtxt`)

19a. Read in this file, so that you have an array with your values. Check with a print command that you succeeded.

19b. How many cars N are on this road?

19c. What is the average velocity

$$\langle v \rangle = \frac{1}{N} \sum_{i=1}^{N} v_i$$

where v_i is the velocity of car i and there are N cars.

19d. What is the largest velocity ?

20. Fun with Matrix

20a. Write a program that sets an array to be as follows:

```
0 0 0 0 0
0 1 1 1 0
0 0 1 0 0
0 0 0 0 0
0 0 0 0 0
```

Check your program with a print command.

20b. Now change the printing so that there are no square brackets being printed.

20c. Define the length of the array, i.e. 5, as variable you set at the beginning of your program. This will allow us later to change the size of our lattice easily.

20d. (**Tool for Fun**) You can look at this array graphically by adding to your python script the following command at the beginning

```
import pylab as pl
```

and then after you have set your array the following two lines (assuming that your array is called `lat`)

```
pl.imshow(lat,interpolation='nearest')
pl.show()
```

For more information Newman describes this tool in §3.3.

21. User-Defined Functions

21a. Start reading in Newman's book §2.6 and as practice for the understanding of page 76 and the first half of page 77, test the commands

```
def factorial(n):
  f = 1.0
  for k in range(1,n+1):
    f *= k
  return f

print(factorial(10))
```

21b. Next add to your program that n is read from input and then $n!$ is printed.

21c. Next let's combine this user-defined function with the repetition tools from last class. Write a program which prints out n and $n!$ for $n = 1, 4, 7, 10, 13, \ldots, 22$.

21d. Continue reading on page 77, and include in your program the user-defined function `distance` . If you want to use `scipy` make sure to include in your program
`import scipy as sp`
and now add to your program (or write a small new python script) with the lines

```
def distance  (r,theta,z):
  x = r*np.cos(theta)
  y = r*np.sin(theta)
  d= np.sqrt(x**2+y**2+z**2)
  print("x ",x," y= ",y," d=",d)
  return d
```

and test your program for the case of $r = 2.0, \theta = 0.5\pi, z = 1.0$ and check whether you get the expected value.

21e. Next write a function which reads in as integer
a total number of seconds and returns the time interval in the form of hours, minutes, and remaining seconds. Test your program with a few examples.

21f. Continue reading §2.6 and then include in a program the definition of `factors(n)` as defined on page 81. Test this function by reading in an integer and printing the factorlist.

22. Good Programming Style

Read carefully §2.7. I could not agree more with each of Newman's rules.

Flow Chart:(if time) start working on the flow chart for your main project