In-Class Work: Python — Arithmetic & Decisions

12. Arithmetic: Read §2.2.4. pages 23 – 27 in Newman's book. While you read the text, test one by one the following commands. (Use a python script for your testing, i.e. use the `print` command as in last class, and for each line first predict the result):

```
2+3*9
2+(3*9)
(2+3)*9
12/5
12.0/5.0
12//5
12.3//5.0
12%5
12.3%5.0
-3+5*2**3
x=3; y=2*x
a=5;b=2;a,b=b,a
a=5;b=2;a,b=3*b,a
```

13. Periodic Boundary Conditions

There is a great application of the above commands which we will use later in the course for the traffic model. The model which we will use is on a one dimensional lattice with so called periodic boundary conditions. That is, if the one-dimensional street has sites $0, 1, 2, \ldots, (L-1)$, where L is the length of the street, then the last site $(L-1)$ continues back on site 0, i.e. like a circular road. For an update from time step t to $t+1$ a car on site $x_{\rm old}$ with integer velocity v moves v lattice sites further on this circular road. Write a program (python script) for a street of $L = 20$, that reads in $x_{\rm old}$ and v and prints out the updated site $x_{\rm new}$. Test your program for several $x_{\rm old}$ and v.

14. Packages

(The following is a slight variation on §2.2.5.)
Try using

```
exp(2.0)
```

If you have not imported a package which contains the function `exp`, then you will get an error message. In Newman's book he uses the package `math`, we will use numpy. So ensure to include in your program for example the line

```
import numpy as np
```

In that case you can determine `exp(2.0)` with

```
print(np.exp(2.0))
```

Test your program with the `import` command and without. Note, you can comment out a line with # (see Newman §2.2.7). Test a few other functions like `sin` and others, by quickly scanning Newman's §2.2.5.

15. Decisions

Read §2.3.1 and as you read try the commands on page 39

```
x=int(input("Enter a whole number no greater than ten: "))
if x > 10:
    print("You entered a number greater than ten.")
    print("Let me fix that for you.")
    x=10
print("Your number is",x)
```

and the commands at the bottom of page 41

```
x=int(input("Enter a whole number no greater than ten: "))
if x>10:
    print("your number is great than ten.")
elif x>9:
    print("Your number is OK, but you're cutting it close.")
else:
    print("Your number is fine. Move along.")
```

Continue reading §2.3.2 and try the commands

```
x=int(input("Enter a whole number no greater than ten: "))
while x>10:
    print("This is greater than ten. Please try again.")
    x=int(input("Enter a whole number no greater than ten: "))
print("Your number is",x)
```

Continue reading §2.3.3 (The third line needs the correction "is not met" to instead "is met".) but stop reading on page 45 before you get to the commands `f1=1` and following lines. Put the book aside and try yourself to write a program which determines the Fibonacci numbers. When you succeeded, continue reading on page 45.

Try the commands on top of the page 46

```
f1,f2=1,1
while f1<=1000:
    print(f1)
    f1,f2=f2,f1+f2
```

and try to understand how exactly this program works. If you google "Fibonacci Nature" you get some beautiful examples and explanations for the occurance of fibonacci sequence in nature.