In-Class Work: Python — Arithmetic, Decisions, Repetitions, and Lists & Arrays

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)
```

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. Also google "Fibonacci Vi Hart", then you get some cool you tube videos from Vi Hart.

16. Repetitions Sample Program

To practice how repetition commands work, copy into your working directory the sample program

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

Look at this program and before running this program, lookup online (or remind yourself) the used commands. Still before running this program, predict what exactly the program will print

on the screen when you run the program. After your prediction, run the program and compare with your prediction.

17. Factorial

Write a program which reads in N and prints out $N!$. (Use a repetition command.)

18. Lists & Arrays Intro: Containers

For examples how to use arrays copy into your working directory the sample program

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

To learn more about lists and arrays, scan this sample file and then read in Newman as described in the following steps.

Lists

Read §2.4.1 and as practice try for page 48 the following commands:

```
r=[1,1,2,3,8]
print(r)
```

and also

```
x=1.0
y=1.5
z=-2.2
r=[x,y,z]
print(r)
y=2.0
print(r)
r=[x,y,z]
print(r)
```

On page 49 and all following pages of this section, replace the `math` package with the `numpy` package the same way as in last class. For example `sqrt(3.0)` is instead `np.sqrt(3.0)`
In additon try the commands

```
a=[3,2.5,4,"hi",-3]
print(a)
print(a[0],a[1],a[3])
a[1]=7.3
print(a)
```

and for page 51 try

```
import numpy as np
r=[1.0,1.5,2.2]
logr=list(map(np.log,r))
print(logr)
```

and for pages 51 and 52 test

```
r=[1.0,1.5,2.2,9.1]
print(r)
r.append(2.8)
print(r)
r.pop()
print(r)
r.pop(1)
print(r)
print(r[2])
```

Arrays

Read §2.4.2 pages 53 and 54. We will also use numpy, but import differently so the commands on page 54 are replaced by

```
import numpy as np
a=np.zeros(4,float)
print(a)
```

and try also

```
b=np.zeros([2,3],int)
print(b)
```

for pages 55 and 56 try

```
d=np.array([[1,2,3],[4,5,6]])
print(d)
d[1,2]=30
print(d)
```

Continue reading, so read §2.4.3. Notice that `loadtxt` allows us to read data from a file, so we do not have to type in by hand each value. Get a feel for how it works, by using in your linux terminal `gedit` (or any other editor you like) to make the file named `values.txt` with the content as the four last lines on page 57. Then you can use a python script which includes the line `import numpy as np` to test the python commands

```
e=sp.loadtxt("values.txt",float)
print(e)
```

In case this made you curious about how to also write into a file of a specified name, you may have a look at `~kvollmay/share.dir/pythonsamples.dir/sample_inout.py`