Chapter 5

Functions and Modules

In programming, a function is a unit of code that works on various inputs, many of which are variables, and produces concrete results involving changes to variable values or actual operations based on these inputs. Almost all programming languages have functions or similar structures. Python is not exception.

5.1 Functions with def

In python functions use the def constructor

```
def name (parameter1, parameter2, etc):
    body code
    return
```

For example is we want to build a function that calculates a power of a number (x^y) we could do

```
def power(x,y):
    r=0
    total=1
    while(r<y):
        total = total * x
        r=r+1
    return total
```

In python is quite easy to have functions with a variable number of arguments The code below would print 1.

```
def minimum(*numbers):
    if len(numbers) == 0:
        return None
    else:
        minimum=numbers[0]
```

```
        for n in numbers[1:]:
            if n < minimum:
                minimum = n
        return minimum


print (minimum(3,1,2,8))
```

5.2 Modules

In python a module is a file containing code. It defines a group of python functions (or objects) and the name of the module is derived from the name of the file. This should have the .py extension. For example, one can include several mathematical functions in mymath.py

to be able to call this functions you just import the module in the beginning of your file:

```
import mymath
```

Futher Reading:

- Generators
- Decorators

Exercises

1. Write a function that receives a list and returns the maximum value in that list
2. Write a function that returns the factorial of a given number. Also attempt this by using a recursive function (a function that calls itself), noting that factorial of n is equal to n times factorial of n-1
3. Write a function that receives a list and returns a ordered version of that list.

Chapter 6

File System

Dealing with files is crucial for almost any program. Programs often have to store data before they finish and/or read it when they (re)start.

To open a file you use the open function

```
file_object = open('myfile','r')
line = file_object.readline()
```

The first line opens a file with the name myfile (which should be in the same directory of the program) with the intent of reading it ('r'). To be able to write one would use 'w'. The second like reads the first line into a variable.

After one is done with a file, it should be closed

```
file_object.close()
```

Here's a full example that counts the number of lines in a file.

```
file_object = open("file_name",'r')
count=0
while file_opbject.readline() != "":
    count=count+1
print(count)
file_object.close()
```

6.1 CSV files

python is an excellent tool to process CSV (comma separated files). CSV files look like this:

```
1/2/2014,5,8,france
1/3/2014,5,2,uk
1/4/2014,9,1,portugal
```

```
import csv

with open('example.csv') as csvfile:
    readCSV = csv.reader(csvfile, delimiter=',')
    dates = []
    colors = []
    for row in readCSV:
        color = row[3]
        date = row[0]

        dates.append(date)
        colors.append(color)

    print(dates)
    print(colors)

    # now, remember our lists?

    whatColor = input('What country do you wish to know the date of?:')
    coldex = colors.index(whatColor)
    theDate = dates[coldex]
    print('The date of',whatColor,'is:',theDate)
```

Exercises

1. Implement a program that reads the content of a Wireshark (https://www.wireshark.org) output file into memory (dictionaries). Run Wireshark in your computer and produce a CSV file. Alternatively download an example file from here https://www.ee.ucl.ac.uk/∼mrio/ws.csv

2. Write a python program that takes the dictionary in the previous exercise and prints the list of flows in the file. For every flow you print the IP source and destination, TCP port and destination and total number of bytes. In the file each row is a packet. A flow is a set of packets with the same IP source, IP destination, TCP source port and TCP destination port (rows 3,4,6 and 7)