Week 4 Exercise Session: functions

E1. Write a function that prints the sum of the first n natural numbers: $1 + 2 + \ldots + n$. Your function has one argument, called n.

E2. Write a function with the name `toFahrenheit` that accepts a temperature in Celsius and returns the temperature in Fahrenheit. Use this function in your program that accepts a stream of temperatures from the user and stop when the user enters q

$$F = 32 + 1.8 * Celsius \tag{1}$$

E3a. Toastmasters club is an international organisation whose aim is to improve public speaking. Every meeting, one person is keeping track how long a speech takes. Every speech has a minimum and maximum length. It is the timekeeper's job to show a green sign at the minimal length, a red sign at the maximal length and an orange sign in the midpoint of minimal and maximal length.

Write a function that prints the minimal length, midpoint and the maximal length given the minimum and maximum length of the speech. Use the `round` function to limit the number of decimals to 2 digits.

E3b. Add a function to make sure that the time is properly formatted. For instance, 3.5 minutes should be written as 3 min 30 sec. Use this function in the previous exercise.

E4a. A prime number is one that is not divisible by any number other than 1 and itself. Write a function `isPrime` that accepts a positive integer and returns true if the given integer is a prime number and false otherwise.

E4b. Use the above function to write a function to write all prime numbers up to an integer n. Write a program to ask for an integer and then use the function you just wrote

E5a. Write a function that computes the factorial of a positive integer. $n! = 1 * 2 * 3 * \cdots * n$

E5b. Use the above function to write a function `estimateExp` that calculates e^x. An estimate of e^x can be computed as the power series

$$e^x = \sum_{i=0}^{n} \frac{x^i}{i!} \tag{2}$$

The function `estimateExp` returns an estimate of e^x. Write two versions of the function `estimateExp`. The first version accepts a parameter n, this is the number of terms you need to include in the sum. The second version accepts a parameter delta. Keep on adding terms to the sum until the term is smaller than delta.

TIP: In the second version of `estimateExp`, you can write the computed term to the screen, this can help you find errors in your functions.
E6. Last exercise session, you wrote a program to check if the number of a credit card is valid. Transform your code to a function. Think about the number of arguments the function needs. If necessary, think about the result this function returns.

E7a. Write a function that draws a pyramid on your screen. The function needs two arguments: the first is the height of the pyramid. The second argument is optional: if not supplied, the symbol '#' should be used to draw the pyramid. A sample pyramid with height 7 and no symbol looks like this:

```
      #
     ###
    #####
   #######
  #########
 ###########
#############
```

However if you call the function with height 5 and symbol '*', the pyramid looks like this:

```
    *
   ***
  *****
 *******
*********
```

E7b. Rewrite your function so it returns a string that contains your pyramid instead of printing all symbols directly to the screen. Use this function in a program that asks for a height, a symbol and then prints out the string returned from your function. To add a newline character to your string, take a look at 'Escape Sequences' at special Topic 2.5 in your book.

E8a. In the US, postal codes are rewritten in bar codes. Each postal code has 5 digits and a check digit. The check digit is computed as follows: Add all 5 digits together, the check digit is the number you have to add to this sum so it is a multiple of 10. For example, postal code 95104 has a sum of 19, so the check digit is 1 to make the sum equal to 20. Write a function that returns the check digit. Don't forget to divide the work in several functions!

E8b. Digits are recoded with bars | and colons :. The next table shows the coding of all digits. Write a function that encodes a digit in a string of colons and bars. If the argument is less than zero or greater than nine, return an error message.

E8c. Finally write a function that given a postal code, generates a barcode for the five digits and the checkcode. Test your code with a program that asks for postalcode. For example, if the postal code is 95014, then the according barcode is `|:|:::|:|:||:::::::||:|::|:::||`
TIP: to make debugging easier, add a special character after each encoded digit - a space for instance - for better readability.

E9. The drag force on a car is given by

$$F_D = \frac{1}{2}\rho v^2 A C_D \tag{3}$$

Table 1: Encoding of digits with bars

Digit	Bar 1	Bar 2	Bar 3	Bar 4	Bar 5
1	:	:	:	|	|
2	:	:	|	|	:
3	:	:	|	|	:
4	:	|	:	:	|
5	:	|	:	|	:
6	:	|	|	:	:
7	|	:	:	:	|
8	|	:	:	|	:
9	|	:	|	:	:
0	|	|	:	:	:

with ρ the density of air (1.23 kg/m^3), v is the velocity in units of m/s, A is the projected area of the car ($2.5m^2$) and C_D is the drag coefficient (0.2).

The amount of power in watts required to overcome such drag force is $P = F_D v$, and the equivalent horsepower required is $Hp = P/746$.

Write a function for each computatation. Next, write a program that accepts a car's velocity and computers the power in watts and in horsepower needed to overcome the resulting drag force. In case you are used to express velocity in mph, then you should know that $1mph = 0.447m/s$.

E10. Start with the number n. If n is even, you divide it by two; otherwise you multiply with three and add one. On this result, you perform the same instruction. You keep on repeating this until you get the number one. How many steps do you need?

An example to help you out. Start with the number 5. If you perform the instructions correctly, you get the next sequence of numbers: 16,8,4,2,1. So you need 5 steps to get to one.

Write a function that given a number n, returns the number of steps required to get to one.

The conjecture of Collatz states in a finite number of steps, you will reach to one.

E11a. Write a function that computes the area of a circle with radius r.

E11b. Suppose you have n circles with radius r, $2r$, $3r$, ..., nr. What is the sum of all areas? Write a function area with two parameters n (the number of circles) and r (the radius of the first circle). Use the function you defined in the first part.

E12. Write a function `string_multiple` with parameters n and s. It should return a string that is equal to n times s. For instance, `string_multiple(3,"abc")` should return 'abcabcabc'. Write a version with the $*$ operator and a version with a while loop. Proof the correctness of the second version.

E13. Write the following functions to compute metrics of spheres:

1. `sphereVolume(r)` computes the volume of a sphere with radius r
2. `sphereSurface(r)` computes the surface of a sphere with radius r
3. `cylinderVolume(r,h)` computes the volume of a cylinder with radius r and height h

4. `cylinderSurface(r,h)` computes the surface of a cylinder with radius r and height h

Define a global variable that is incremented everytime an instruction is executed. Call each functions defined above once with random values and print the value of the global variable. **WARNING:** As mentioned in a programming tip in the book, the use of global variables is discouraged as they result in programs that are more difficult to understand.

E14. Functions themselves can be an argument to an other function as well. Take a look at the following code.

```
def isOK1(x):
        #isOK if greater than 5
        return x>5
def isOK2(x):
        #isOK if has remainder 2 when dividing by 5
        return x%5==2
def checkNr(f,n):
        if f(n):
                print "n is ok"
        else:
                print "n is not ok"
checkNr(isOK1,3)
checkNr(isOK2,3)
```

In the last line, the function `isOK2` is passed as parameter f. So while executing `checkNr`, `f(n)` comes down to the execution of `isOK2(n)`.

Write a function `applyComputation` with a function f as a first parameter and a number n as a second parameter. This function f should be applied to the number n and the result should be the result of `applyComputation`.