Laboratory 3: For and while loops

1. We can use for-loops in order to calculate finite sums and products. For instance, in order to calculate the sum

$$\sum_{\nu=1}^{n} (\nu^4) = 1^4 + 2^4 + 3^4 + \ldots + (n-1)^4 + n^4$$

we can use a programming pattern centered around an accumulator that contains partial sums. Initially, the accumulator is zero. We then generate the numbers $\nu = 1,2,3,\ldots$ in turn and add the addend ν^4 to the accumulator. The resulting accumulator at the end of the for-loop is the sum. The code fragment below gives the code. The lines are numbered on the right. In line 1,

```
n = int(input("Enter n: "))                          #1
accumulator=0                                        #2
for i in range(1,n+1):                               #3
    accumulator += i**4                              #4
    #print("i", i, "accumulator:", accumulator)  #5
print("The result is", accumulator)                  #6
```

we just asks the user to enter the range of the summation. Notice again the need to convert the string returned by the input-function into an integer. The second line initializes the accumulator. Line 3 gives the for loop. Notice the bounds. The first value for *i* has to be 1, the last one *n*, meaning that the stop value is *n*+1*.* Many people are disturbed by Python using the stop-value (the first value **not** taken), but once you understand a bit more of the language, it actually does make sense. Line 4 then updates the accumulator. The += is just a short-cut notation for

`accumulator = accumulator + i**4.`

Line 5 is commented out and is used for debugging. It prints out the progress in the for-loop. Finally, Line 6 presents the result. Notice that it is dedented as it is not part of the for-loop.

Tasks:

A. Calculate $\sum_{i=0}^{1000} i^2$. (The answer is 333 833 500).

B. Calculate $\sum_{i=0}^{1000} \frac{1}{1+i^2}$. (The answer is 2.075 674 547 634 748.)

2. For loops can similarly be used for the calculation of products. However, the accumulator then needs to be initialized to 1 (the neutral element of multiplication). If it were initialized to 0, then updating it by multiplying it with an factor still gives you 0.

Tasks:

A. Calculate $\prod_{i=1}^{1000} \frac{2i+1}{2i}$. (The answer is approximately 35.696).

B. Calculate $\prod_{i=1}^{100} \frac{i^2+1}{i+2}$. (The answer is approximately 6.5944×10^{154}).

Approximations for π

3. There are many formulas that allow us to calculate the mathematical constant π. Test out the following product and sums for π by evaluating the first 100, 1000, and 10000 members of the sums or products.

A. Eulers Formula: $\frac{\pi^2}{6} = \sum_{i=1}^{\infty} \frac{1}{i^2} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \ldots$

B. Plouffe's BBP digit extraction algorithm:

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{4}{8k+4} - \frac{4}{8k+5} - \frac{4}{8k+6} \right)$$

ASCII Drawing

4. Write a program that draws the picture below. Notice that the first line has 6 empty spaces (" ") followed by an asterisk character, followed by the same line in reverse; the second line five empty spaces, followed by an asterisk, followed by an empty space, and then the first part in reverse: one empty space, an asterisk and five empty spaces (which we do not need to explicitly write. The second line can be generated by printing the expression

```
5*" "+"*"+1*" "+1*" "+"*"
```

and the next line by printing

```
4*" "+"*"+2*" "+2*" "+"*"
```

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

A number guessing game

5. Implement a game, where the user guesses a number between 1 and 100. If the guess is less than the number, the program tells the user that, if the guess is higher, the program tells the user that, and if the user guesses the number, then the program congratulates the user.

The program needs to generate a random integer between 1 and 100 (limits included). This is achieved using the module `random` and its method `randint`. You can adopt the following code fragment that prints out 10 random numbers between 1 and 100.

```
import random
for _ in range(10):
    print(random.randint(1,100))
```