HomeWork 2 - CS5007

Learning objectives

- Functions
- Lists
- Loops

The basic assignment is worth 100 points and the extra credit option is worth up to 5 additional points. Details are provided in this document. This is an individual assignment. You may discuss any aspect of this assignment with anyone, but you must type everything into an IDLE window (or other editor) yourself.

Write your code and comments on a file `FIRSTNAME-LASTNAME-HW2.py` that you will create from scratch (there is no template). Upload it on the course website (no delay accepted).

1 Patterns (20 points)

1) Write a function named `rectangle` that given two integers n and m prints out two identical rectangles of width n and height m, separated by an empty column, using the symbol '$*$'. You may use a loop or recursion. For instance, the call `rectangle(4,6)` should exactly lead to the following result on IDLE:

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

2) Write a function named `triangle` that, given a strictly positive integer n given as argument, prints out a triangle of height n, using the symbol '$*$'. You may use a loop or recursion.

By indexing the height from 0 (top) to $n-1$ (bottom), this triangle should be such that: (1) Each row at depth i, $i > 0$, has two more stars than the number of stars of the previous row at depth $i-1$. (2) At depth 0, the function prints one star.

For instance, the call `triangle(6)` should exactly lead to the following result on IDLE:

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

2 Function on lists (15 points)

Write a function `clean1(aList)` that takes a list of integers `aList` as argument, and returns a new list where multiple occurrences of values have been removed. **The function should NOT modify its argument**. This means that if we print the list (given as argument) after executing the function, it did not change. For instance, the following statements:

```
al = [1,2,3,4,4,4,5,1,2,1,5]
newlist = clean1(al)
print(al)
print(newlist)
```

print the following result (a different order of values is acceptable):

```
[1,2,3,4,4,4,5,1,2,1,5]
[1,2,3,4,5]
```

▷ *The principle of your algorithm may be the following:*

- *Create a new empty list* `tmp`*.*
- *Use a "for loop" to fill* `tmp` *with each new value found in the argument list.*
- *Return* `tmp`*.*

3 Function on lists (15 points)

Write a function `clean2(aList)` that takes a list of integers `aList` as argument, and modifies this list, such as multiple occurrences of values have been removed. **The procedure does not return a list: it modifies its argument (reference `aList`).**

For instance, the following statements:

```
al = [1,2,3,4,4,4,5,1,2,1,5]
print(al)
clean2(al)
print(al)
```

print the following result (a different order of values is acceptable):

```
[1,2,3,4,4,4,5,1,2,1,5]
[1,2,3,4,5]
```

4 Function on lists (25 points)

Write a function named `fct(aList)` that takes a list of integers `aList` as argument and returns a string containing elements of the list that are a power of 2, all on the same line (i.e., without a new line after each printed value). Write a call to this function with argument $[1, 2, 3, 4, 5, 16, 255, 256, -1, -2, 84]$, embedded in a print statement to see the result on IDLE.

5 Taylor Sinus (25 points)

Required: To have points, you must use loop(s), NO recursion.
From calculus, we know that every continuously differentiable function can be represented as the sum of an infinite series. A finite subset of the series can be used to approximate the function. In this part, you will use Taylor series to approximate a sine function of an angle x in radians. Write a function `taylor(x,n)` that approximates the $\sin(x)$ by returning the sum of the first n elements of its Taylor series:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^{n+1} * \frac{x^{2n-1}}{(2n-1)!}$$

For instance, the following statements:

```
x = math.pi/2
print(math.sin(x))
print(taylor(x,7))
```

lead to the following result on the IDLE:

```
1.0
1.0000000006627803
```

6 Extra credit (5 points)

Implement a function `consecutives(aList)` that takes a list of integers as argument and return the maximum number of consecutive integers that are all equal, within this list. For instance, the following statements:

```
l1 = [1,1,1,3,3,4,4,4,4,4,3,1,7,3,3,4,4]
l2 = []
l3 = [1,2,1,1,2,1,1]
print(consecutives(l1))
print(consecutives(l2))
print(consecutives(l3))
```

lead to the following result on the IDLE:

```
5
0
2
```