Introduction to Python Programming

Exercise Sheet 2: Functions

Doc Strings

A *Doc String* is a string in a Python program used to explain what a function or module does. It is always a string using triple-quotes and is the first statement in a function or module. These strings are not required, but are useful. For example, if you type `help(print)` in the python shell it will show you the doc string of the function. This also works for modules. Try typing `import math` and then `help(math)` in Python shell.

We will use doc strings to give a description of the behavior of the function. Further, we will give an example of the expected output for a given input as you would see on the shell. You can also use Python to automatically use these to test your code. The statement `doctest.testmod()` tells Python to scan the file for strings which look like examples and match the output of the code to the output on the following line. Basically, if you fill in the functions and run the module Python will tell your code gives the correct output for the example input. This does not guarantee that your code is correct, however, if your code fails the tests, than it is definitely not correct.

Zeller's Congruence (6 points)

Please note: This is not the only exercise! See the `2_Functions.py` file.

Zeller's congruence is an algorithm devised by Christian Zeller to calculate the day of the week for any Gregorian calendar date. Your task is to implement this algorithm in Python. Day, month and year are to passed to the function as arguments. We provide a program which asks the user for his/her input and displays the result of your function to the user. You don't need to change anything in the file `zeller_driver.py`, just run it. It uses your code from the file `2_Functions.py`

It is usually a good idea to separate the logical parts of a program from the parts which interact with the user (the User-Interface.) This allows one to reuse the code with different user-interfaces, for example on a smart-phone or over the web. It also makes testing easier, which means better code.

The formula is the following:

$$h = \left(q + \left[\frac{(m+1)*26}{10} +\right] + K + \left[\frac{K}{4}\right] + \left[\frac{J}{4}\right] - 2*J\right) mod\ 7$$

- h indicates the day of the week (0=Saturday, 1=Sunday, 2=Monday, ...)
- q is the day of the month
- m is the month
- K is the year within the century (e.g. 68 for 1968)

- J is the century within the year (e.g. 19 for 1968)
- January and February are treated as months number 13 and 14 of the *previous* year, e.g. 01/01/2000 becomes 13/01/1999.
- All the divisions in the formula are **integer divisions**, i.e. we divide the numerator by the denominator and then discard the fractional part, e.g. $\frac{7}{2} = 3$. You can achieve this in Python by using the integer division operator: `7//2`. Hint: This operator is also useful at another part of this program.

Sample Output from zeller_driver.py:

```
** ZELLER's CONGRUENCE **
Let's compute the weekday!
Please enter year: 2011
Please enter month (a number between 1 and 12): 10
Please enter day of the month: 6
10/6/2011  is a Thursday!

** ZELLER's CONGRUENCE **
Let's compute the weekday!
Please enter year: 2000
Please enter month (a number between 1 and 12): 1
Please enter day of the month: 1
13/1/1999  is a Saturday!
```