LAB EXERCISE TWO

Branching and iteration

Before you attempt these exercises, please make sure that you watched the video entitled *“Branching and iteration”* on Learning Central.

You can work on these exercises in your own time or during the scheduled on-line lab sessions. If you find the first few exercises too easy — feel free to skip to the harder ones. If you get stuck — do not be shy to ask for help or advice from the teaching associates; they are here to help you. It is also ok to discuss the solutions with your peers (these labs are not assessed!), however make sure you understand everything by yourself.

Good luck!

1 Write a Python program that prompts the user for a number and then prints “`Odd`” if the number is odd, or “`Even`” if the number is even.
Hint: Use the `input()` function to read a string from the user. Use the appropriate conversion functions, like `int()` or `float()`, to convert the user’s input into numbers.

2 Write a Python program that prompts the user for their height (in centimetres) and mass (in kilograms) and computes and prints the approximate waist size (circumference) of the user in inches. Assume that humans are cylindrical.
Hint: The volume of a right cylinder is $V = \pi r^2 h$. The circumference of a circle is $C = 2\pi r$. Density $\rho = \frac{m}{V}$. Humans just barely float in water.
A cylinder is a decent approximation for human shape. Can you devise a better one?

3 Write a program that reads three numbers and prints the largest one of the three. Please use the `if`/`else` constructs for this exercise instead of the built-in `max()` function.

4 Write a program that repeatedly reads numbers from the user until the user enters an empty string. The program should then print the average of these numbers. Make sure your program gracefully handles the case when the user enters no numbers.
Bonus task: Modify your program to also print the smallest and the largest of the numbers entered.

5 The traditional song “99 Bottles of Beer” has very repetitive lyrics:

```
99 bottles of beer on the wall, 99 bottles of beer.
Take one down, pass it around, 98 bottles of beer on the wall.
... and so on ...
```

This verse is repeated, each time with one fewer bottle, until the number of bottles reaches zero. Write a program that prints all the verses of the song.

Hint: The construct

```
for bottles in range(99, 0, -1):
    ...
    <your code here>
    ...
```

will iterate for loop variable `bottles` from 99 down to 1.
See also `https://docs.python.org/3.8/library/stdtypes.html#range`

6 Write a program that reads a string from the user and calculates the number of letters and digits in the string. For example, given the input

```
The 2nd chapter has 12 pages.
```

the output should be:

```
Letters: 20
Digits: 3
```

Hint: use the `isdigit()` and `isalpha()` methods to determine if a character is a digit or a letter respectively. Remember, the construct

```
for ch in string:
    ...
    <your code here>
    ...
```

will iterate through the characters in the string one by one.

See also:
`https://docs.python.org/3/tutorial/classes.html#iterators`
`https://docs.python.org/3/library/stdtypes.html#str.isalpha`
`https://docs.python.org/3/library/stdtypes.html#str.isdigit`

7 Write a program, appropriately using `for` loops, to print the following pattern:

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

Make sure you make the size of this pattern a *parameter* and hence can print such a pattern of any size.

8 Write a program that plays the “higher or lower” game with the user. The program should choose a random integer number between 1 and 10. The program should then repeatedly prompt the user to enter a guess of the number, until the user guesses correctly. After

each attempt, the program should respond whether the guess was higher than, or less than the chosen number. An example game may look like this:

```
Your guess? 3
-- My number is bigger.
Your guess? 8
-- My number is smaller.
Your guess? 6
-- My number is smaller.
Your guess? 5
-- Correct!
```

Hint: you can generate random integer numbers as follows:

```
# Import the "random" module
import random
# A random integer between 1 (inclusive) and 11 (non-inclusive)
number = random.randrange(1, 11)
```

See also `https://docs.python.org/3.8/library/random.html`

Play this game with a friend and try to devise a strategy for winning this game with the smallest number of questions. If your opponent picks integer numbers in the range $1 \ldots 1000$, how many questions are definitely sufficient to guess the number?

9 Some words, like “evitative” or “rotator”, look the same written backwards; they are called *palindromic* words. Write a program that reads a word and checks whether it is a palindromic word. *Bonus task:* modify your program to also recognise palindromic *sentences*, ignoring spaces.

10 Let’s do a numerical experiment! Often in computer science, the simplest computations produce remarkably rich behaviour. Consider the following sequence of numbers. Start with a positive integer number n. If n is even, then take $n/2$ to be the next term. However, if n is odd, then take $3n + 1$ to be the next term. In other words, each next number in the sequence is

$$n_{\text{next}} = \begin{cases} n/2 & \text{if } n \text{ is even,} \\ 3n+1 & \text{if } n \text{ is odd.} \end{cases}$$

The sequence stops when n becomes 1. For example, starting with $n = 12$ we get:

$$12 \to 6 \to 3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1.$$

Or, starting with $n = 19$, we get:

$$19 \to 58 \to 29 \to 88 \to 44 \to 22 \to 11 \to 34 \to 17 \to 52 \to$$
$$26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1.$$

Write a program that for any given positive integer n will print this sequence. Investigate, experimentally or otherwise, *how many steps* does it take to reach 1 depending on the starting number n.
Do you think this sequence will *always* reach 1 no matter what number n you start with? If not, can you find such n for which it will not?