SOFTWARE 1 PRACTICAL

Loops

Week 3 – Practical 3

We have seen that Python **list** are iterables that can be modified. We can access an element from the list using its position. We can also add an element at the end of the list by using the method **append**. We have already used another iterable without knowing it, Python **string**. A string can be seen as a collection of characters. In the same way as list, we can access one of its character by using its position as shown below on lines [2] and [4]. However, as opposed to list, strings cannot be modified. To add characters at the end of a string, we need to concatenate a string at the end of the original string using the + operator, and reassign the newly built string to the variable as shown below on line [6].

```
[1] >>> word = ‘Flying’
[2] >>> print(word[0])
[3] F
[4] >>> print(word[2])
[5] y
[6] >>> word = word + ‘ Circus’
[7] >>> print(word)
[8] Flying Circus
```

Exercise 1:

1. Write a script that takes a sentence from the user without any punctuation and prints the sentence without any white spaces. Note a white space is represented by `' '`, and an empty string is represented by `''`.

```
>>> enter a sentence: this is a SHORT sentence
thisisaSHORTsentence
```

2. Same as above except that each word in the output should start with a upper case letter and all other letter should be lower case (also known as CamelCase).

```
>>> enter a sentence: this is a SHORT sentence
ThisIsAShortSentence
```

3. Write a script that takes a sentence from a user written in CamelCase (without any blank spaces), creates the list of words from that sentence, and then prints that list.

```
>>> enter a sentence in CamelCase: ThisIsAShortSentence
['This','Is','A','Short','Sentence']
```

Exercise 2:

1. Write a script asking the user to enter a series of positive integers separated by a white space, and then prints the number of even numbers that was entered. For example:

```
>>> enter a series of numbers: 1 2 4 3 6 8 5 2 11 100 101
There are 6 even numbers
```

   You will need to use the built-in function `input()` and the string method `split()`. You should also remember that we can convert a string representing an integer into an int like so:

```
>>> twenty = int('20')
>>> type(twenty)
<class 'int'>
```

2. Same question except we would like to have the list of even number as well

```
>>> enter a series of numbers: 1 2 4 3 6 8 5 2 11 100 101
There are 6 even numbers: 2 4 6 8 2 100
```

3. Same question except we would like to have the list of even number without duplicate, that is remove the second 2 in the example below.

```
>>> enter a series of numbers: 1 2 4 3 6 8 5 2 11 100 101
There are 5 distinct even numbers: 2 4 6 8 100
```

Exercise 3:

The aim of this exercise is to draw a sudoku board (not to solve it). To start with we will be using a 4×4 board (as opposed to the classic 9×9).

1. Write a script that asks a user to enter 4 digits (comprised between 0 and 4) separated by a white space, the store each digit in a list and print the list.

```
>>> enter 4 digits (0..4) separated by a space: 0 2 1 4
[0, 2, 1, 4]
```

2. Write a script that ask a user to enter 4 times a sequence of 4 digits, and store it in a 2D list, that is a list containing 4 lists of 4 digits each. The script should print the 2D list. The output should look like:

```
[[0,2,1,4],[3,4,2,1],[1,2,3,4],[0,0,2,3]]
```

3. Modify your script so the output looks likes:

```
+-+-+-+-+
|0|2|1|4|
+-+-+-+-+
|3|4|2|1|
+-+-+-+-+
|1|2|3|4|
+-+-+-+-+
|0|0|2|3|
+-+-+-+-+
```

4. Modify your script so the 0 are replaced by a blank space.

```
+-+-+-+-+
| |2|1|4|
+-+-+-+-+
|3|4|2|1|
+-+-+-+-+
|1|2|3|4|
+-+-+-+-+
| | |2|3|
+-+-+-+-+
```

Challenge: *Very Hard*

Mastermind is a code-breaking game for two players. For this exercise we assume the code is a five digits number such as 01091. The codemaker set up a code and hide it from the other player (the codebreaker). The codebreaker tries to guess the code. After each guess, the codemaker provides feedback to the codebreaker. For each digit that is in the right place, a red peg is given to the code breaker. For each digit that is misplaced, a black peg is given. If there are duplicate digits in the guess, they cannot all be awarded a key peg unless they correspond to the same number of duplicate digits in the hidden code. For example, if the hidden code is 00211 and the player guesses 11011, the codemaker will award two red pegs for the two correct 1s (third and fourth 1s in the guess), nothing for the first and second 1s as there is only two 1 in the code, and a single black key peg for the 0s (as there is only one 0 in the guess). No indication is given of the fact that the code also includes a second 0. In the feedback we provide the red peg first and then the black peg so it does not provide any indication of where the correct peg are placed. Using the previous example, the feedback is “RRB”, and not “--BRR".

For the purpose of the program, the code and the guess are stored as strings (for example “00211”). Implement a script that takes a code and a guess from a user, and print the feedback.

```
>>> enter a 5 digits code:
00211
Enter a guess (5 digits code):
11011
Result is RRB
```

Advanced:

Modify your script so a code is generated randomly (search for random in Python), and the user is prompted to enter a guess until he/she finds the code. After each guess, the feedback should be printed.