Data Structures and Algorithms in Python
Programming Assignments I

dr. Stevan Rudinac and Sebastian Melzer

In the following assignments you will further exercise the Python basics discussed in the first week of the course, such as data types and control flow. These assignments will not be counted towards the grade, but their completion is highly recommended.

1 Temperature converter

Celsius, Fahrenheit and Kelvin are the best known temperature scales. Their relations are given in Table 1. In this assignment you will implement a temperature converter according to the following specifications:

1. User can input temperature in an arbitrary scale, such as: 25C, 77F or 298K.
2. Your code should compute and print temperature in the other two scales.

Table 1: Relations between Celsius, Fahrenheit and Kelvin scales.

	Celsius	**Fahrenheit**	**Kelvin**
Celsius	·	$[°C] = ([°F] - 32) \times 5/9$	$[°C] = [K] - 273.15$
Fahrenheit	$[°F] = [°C] \times 9/5 + 32$	·	$[°F] = [K] \times 9/5 - 459.67$
Kelvin	$[K] = [°C] + 273.15$	$[K] = ([°F] + 459.67) \times 5/9$	·

Hint:

```
text = '25C'
temperature = text[:-1]
scale = text[-1]
```

2 Prime numbers

A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. Example prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23 and 29. In this assignment you should implement a prime number generator according to the following specifications:

1. User is prompted to enter a desired sequence length n.
2. Your Python code should print the first n prime numbers. The output should look like this:

```
1. 2
2. 3
3. 5
4. 7
...
...
...
```

Note: At this point you do not have to worry about algorithm complexity.

✉ s.rudinac@uva.nl, sebastian.melzer@student.uva.nl

3 Hangman game

In this exercise you will implement a hangman game[1]. The hangman game is a simple letter game usually played with pen and paper. In this game players must deduce a secret word by guessing individual letters. The game consists of two parties. The challenger thinks of a secret word and draws short horizontal lines on a piece of paper for each letter of the word. The other party, the guesser, tries to guess the word by suggesting a letter each turn. If the letter is in the word the challenger writes the letter in the correct position on the paper. If the letter does not occur in the word the challenger draws an element of a hangman stick figure. The game ends when either the word is completely guessed or the stick figure is completely drawn.

In your implementation the computer should act as challenger and the user as guesser. Try to think of the control flow that is needed in the game. Below is a walk-through that you can follow to help you through the creation of the game if you get stuck.

1. You can start by creating a main loop for the game. In this loop you should repeatedly ask the user to enter a letter.

2. Next you should create a variable to hold the secret word. Initialize it with `'secret'` or some other placeholder. In the main loop print whether the entered letter is in the secret word or not.

3. Create a variable to keep track of the guessed letters. You can use an empty string, i.e. `guessed_letters = ''`. Each time the user enters a letter you should add this letter to the variable.

4. To print the secret word only filled in with the guessed letters you can iterate over the characters in the secret word. For each character you can check if it is in the guessed letters variable. If it is, just print the character, otherwise print a placeholder, e.g. `'_'`.

   This approach requires you to print without newlines. To do this you can use `print(letter, end='')` or `print letter,` in respectively python 3 or python 2.

5. You have to check whether the player has guessed all letters of the word. You could do this by creating a boolean variable before the printing loop, e.g. `letter_missing = False`. Inside the branch of the loop where you print an unguessed character (`'_'`) you set the variable to `True`. After the completion of the loop you check the variable. If the variable is still set to `False`, print a victory message and break from the main loop.

6. Create a variable to keep track of the number of failed guesses by the user, e.g. `failed_guesses`. If the this number exceeds some limit, e.g. 6, you should break out of the main loop and act accordingly.

7. To draw the actual hangman you have to define a frame for each of the animation phases. Each frame will be associated with some number of failed guesses. Inside the main loop you need to print the correct frame. You can do this with if-else statements, but alternatively you can use a list. A list is an ordered sequence of elements. You can create your list of frames like below.

```
hangman_frames = [
# 0 failed guesses
'''








''',
# 1 failed guess
'''
```

[1] https://en.wikipedia.org/wiki/Hangman_(game)

```
======
''',
# more frames here...
# x failed guesses
r'''
 -----
 |/  |
 |   o
 |  /X\
 |  //
 |
======
'''
]
```

You can access a specific item from the list by its index using square brackets, thus you can print `hangman_frames[failed_guesses]` to show the correct frame.

You will learn more about lists in the upcoming lectures.

8. Lastly we need the secret word to be randomly chosen by the computer. Again we will use a list for this purpose. There are multiple ways to populate this list. You can create a list manually or get one from an external resource.

   To pick a random word you can use random.choice. The random module defines pseudo-random number generators exactly for this purpose. Import the random module by adding the `import random` at the top of your file. Now you can use `secret_word = random.choice(word_list)`