TMTplus Introduction to Scientific Programming

Mahdi Farnaghi, Mahdi Khodadadzadeh & Robert Ohuru

March 2021

Note from the teaching staff

The materials that we use for this course are somewhat new and have been heavily reworked. We will likely have made mistakes in this work or have glossed over issues that deserved better or different treatment. Where you find issues worth noting please do report these to us as it will allow to repair and improve.

By the way, welcome to the Exercise book for Introduction to Scientific Programming! We believe this book is self-explanatory, so go ahead and practice.

Chapter 6

Strings, dictionaries & their expressions

Strings are a fundamental information carrier for any information system, and are probably the most versatile and direct source for communication with humans. One may argue that images are equally direct and powerful . . . Scripting 2 must sometimes work on text sources, whether online or stored locally, and present results; in all of this strings are seriously needed, and so the code developer must know how to play them.
Dictionaries provide a direct look-up mechanism that allows to associate a bunch of information with a single characteristic key value, which is often a string. In the old days, a dictionary was called "associative memory," because of this direct access mechanism. Dictionaries are commonplace in languages such as Python, and come in many disguises. Not now but later we will see that JSON objects are also dictionaries, for instance.

The exercises

The main goal of each exercise is to use the Python programming language to review concepts you have learned during the lectures.

In this exercise, keep in mind what is the power of strings and dictionaries: what can you encode with them, and how they can be used for various bookkeeping tasks while a script is running, and so forth. In short: acknowledge the power of these constructs for coding later.

6.1 The use of strings

Ex 6.1

Decide on a variable name, and assign a string that is a fruit name of your choice. Print its length on the console. Use the multiplier operator to print this fruit ten times in the console and add two delimiter symbols (e.g. < and >) at the beginning and at the end of this long string.

Ex 6.2

Now find four fruits of your choice, get their names as strings, and concatenate these to form a unique and long string. Print its length. Use the multiplier operator to print it three times in the console.

Ex 6.3

Define a long string that contains the name of five different fruits, separated by a space character. Since strings are also sequences, use the slicing operator to print all the characters every 2 steps, and also 4 steps. Now do the same, but using negative steps. Are the outputs the same? Do you understand why?

Ex 6.4

Determine what are the truth values of the following expressions. Can do so by your interpretation or by using the console to verify

expression	truth value
"watermelon" == "waterMelon"	
"BANANA" == "BANana"	
"a" in "cranberry"	
"g" not in "dragon fruit"	
"berry" in "blueberry"	
"cherry".isupper()	
"CHERRY".islower()	
"pear***apple".isalnum()	
"pear444apple".isalpha()	

Ex 6.5

Write a string containing four fruit names, separated by a space character. Turn this list into a list of characters and print its length. Join the list of characters to become a unique string and replace the space character by an arrow symbol (e.g., '->').

Ex 6.6

Make a string of five fruit names of your choice separated by an arrow symbol (e.g., '->'). Use the `.split()` method to create the list and use the arrow symbol for the splitting. Remove one of the fruit names from the list using the `del` command and merge the list of fruit names back to a long string using the `.join()` method. This time, use an asterisk to separate the fruit names in the string. Can you replace these asterisk symbols by blank spaces in one script line?

Ex 6.7

Make a long string containing seven different fruit names. Iterate over the list so every time you find a vowel in it, the program prints that vowel. Recall that

the `in` operator can be useful to avoid unnecessary `if` statements.