2018/2019

Practicals°1: Secret codes & network

We'll use the Python language for these practical (version 2.7.x for programming reasons) on Raspberry Pi or Linux devices.

There are two possibilities to log on Raspberry Pi

- directly on the physical device with your login and password (you must have at least a keyboard and a screen)
- or remotely via ssh interface (you must have found its IP address)

Python language is already installed on Raspberry Pi as well as necessary libraries.

Exercice 1:

In cryptography, the Caesar's cipher is a very simple technique of encryption where each letter is replaced by another one. For example, with a substitution of 3, A will be replaced with the letter D, B will become an E, and so on. Julius Caesar used this technique to communicate with his generals. **ROT13** (rotation of 13 places) is often used as an example. In Python, the key ROT13 can be represented by the following dictionary:

```
key = {‘a’:’n’, ‘b’:’o’, ‘c’:’p’, ‘d’:’q’, ‘e’:’r’, ‘f’:’s’, ‘g’:’t’, ‘h’:’u’,
       ‘i’:’v’, ‘j’:’w’, ‘k’:’x’, ‘l’:’y’, ‘m’:’z’, ‘n’:’a’, ‘o’:’b’, ‘p’:’c’,
       ‘q’:’d’, ‘r’:’e’, ‘s’:’f’, ‘t’:’g’, ‘u’:’h’, ‘v’:’i’, ‘w’:’j’, ‘x’:’k’,
       ‘y’:’l’, ‘z’:’m’, ‘A’:’N’, ‘B’:’O’, ‘C’:’P’, ‘D’:’Q’, ‘E’:’R’, ‘F’:’S’,
       ‘G’:’T’, ‘H’:’U’, ‘I’:’V’, ‘J’:’W’, ‘K’:’X’, ‘L’:’Y’, ‘M’:’Z’, ‘N’:’A’,
       ‘O’:’B’, ‘P’:’C’, ‘Q’:’D’, ‘R’:’E’, ‘S’:’F’, ‘T’:’G’, ‘U’:’H’, ‘V’:’I’,
       ‘W’:’J’, ‘X’:’K’, ‘Y’:’L’, ‘Z’:’M’}
```

Your task in this exercise is to implement a **ROT-13 encoder / decoder**. Once done, you should be able to read the following secret message: `Pnrfne pvcure? V zhpu cersre Pnrfne fnynq!`

Exercice 2:

Nothing stops a cryptanalyst from guessing one key, decrypting the ciphertext with that key, looking at the output, and if it was not the correct key then moving on to the next key. The technique of trying every possible decryption key is called a **brute-force attack.** It isn't a very sophisticated hack, but through sheer effort (which the computer will do for us) the Caesar cipher can be broken.

Your task in this exercise is to implement a brute-force decoder that tries and displays every solution (that means for every rotation key)

Try with this message: GUVF VF ZL FRPERG ZRFFNTR

Exercice 3:

The alphabet defined by ICAO (International Civil Aviation Organization) assigs words to each letter (Alfa for A, Bravo for B, etc.) allowing a non-ambiguous communication between emitters and receptors.

- Define a dictionary in Python allowing this automatic transcription
- Write the function `speak_ICAO()` which translate a text into words pronounced with a TTS (Text-to-Speech) according the ICAO alphabet
- Modify this function to accept a supplementary parameter: a floating-point number to indicate a delay between two ICAO words
  Note: you will have to import two libraries `os` and `time`.
- Under Linux systems, you can use the **espeak** command to pronounce sentences (`espeak -ven "text to say" 2>/dev/null`)
- Under mac systems, the **say** command should be fine …
- Finally, under windows systems, you can simply download the **say** executable (*SayStatic*.exe) here: `https://github.com/truillet/ups/blob/master/m1csa/Tools/SayStatic.exe`

Note 2:

With Python language, here are the command lines to be able to “pronounce” something:

```
import os

txt = "papa tango charlie"

os.system("espeak -ven \"" + txt + "\" 2>/dev/null")
```

Exercice 4: Ping/Pong using sockets

1. Using the document **http://docs.python.org/2/library/socket.html#example**, write `client.py` and `server.py` files in which server answers “*pong*” when client sends “*ping*”.
2. Using the `time.clock` function, modify the client to measure the response time of its request. Successively, test several clients and verify that the response time varies from one client to another.
3. Open two clients. What happens if client2 sends its request while client1 performs its request? To see this phenomenon, add an artificial slowdown of the server using the `time.sleep(5)` instruction available in the `time` module.
4. To improve processing of clients, it is decided to thread the server, in other words the server will delegate the processing of the client to different process while it is satisfied to wait for the connection on its listener socket.

Using the documentation available here **http://python.developpez.com/faq/?page=Thread**, write a class named Service (`threading.Thread`) that receives the client socket in its constructor and processes the client after Its connection with the server.