Homework 9 – Part 1

- Keeping time, scheduling tasks, threading and multiprocessing
- Sending email and text messages
- Socket programming

Given April 2nd – Due April 23rd

Exercise 1. Umbrella and jacket reminder

Homework 8 exercise 3 you developed a program to retrieve the current weather data from http://openweathermap.org. Write a program that runs just before you wake up in the morning and checks whether it's raining that day, it's going to be very cold, it's going to snow, etc. If so, have the program email you a reminder to pack an umbrella or an extra jacket.

Make sure that while your program is waiting for the next day it is NOT in the main thread, so run it asynchronously.

(Not need to do this, just informative) If you would like to have this program (or any other program) to run every day but not in your computer you could do the following:

1. Upload your code to github.
2. Create an online virtual machine on AWS (or any other platform). You can learn how to do that in the following link: https://aws.amazon.com/getting-started/tutorials/launch-a-virtual-machine/
3. Connect to the virtual machine instance using SSH from your terminal
4. Clone your GitHub project.
5. Run the program you want.
6. (Optional) You may want to use *tmux* to be able to run different python programs at the same time. It is useful for running more than one command-line program at the same time. https://hackernoon.com/a-gentle-introduction-to-tmux-8d784c404340

Exercise 2. Controlling your computer through email

Write a program that checks an email account every 15 minutes for any instructions you email it and execute those instructions automatically. For this exercise you will only need to execute prints but it will be a good sample for your future projects. That means that one instruction could print "Hello world!".

Your project should have the following functionalities:

- Retrieve emails every 15 minutes
- Your email subject line should contain a keyword and the instructions. Your program should understand the instructions. An example of subject line could be:

```
EMAIL INSTRUCTIONS {"instruction": 2, "arguments": {"name": "Damia"}}
```

- Execute the instructions. At least develop 3 different instructions. Remember that those instructions could be as simple as printing "Hello world!" to the terminal.
- Make sure it doesn't repeat instructions of repeated emails. Let's say you send an email with instruction number 2. Later on, when you check the email again and you read that same email with instruction 2 you should not execute it as it was executed before.
- Make sure that you run this asynchronously.

For extra points you could develop the following functionalities:

- If you send an email with the keyword but wrong JSON format, the program has to send an email to you with instructions on how to use the program.
- Send an email when instructions are received correctly and started.

Exercise 3. Socket programming.

Write a TCP Server and a TCP Client. The server will have a string variable that will be changed with values sent by the client. The client will ask the user for the following options:

1. Retrieve the text in the server
2. Update the text in the server
3. Close

On option 1 the client will ask the server for the text and the server will send the text to the client. On option 2 the client will ask the user for some input text and that text will be sent to the server, which will update the text variable. On option 3 the client will tell the server to close the socket and both client and server will close the socket connection.

You can save your files as tcp-client.py and tcp-server.py. The output of both server and client could be as the following:

tcp-client.py

```
damiafuentes$ python3 tcp-client.py
Starting client

What do you want to do?
1. Retrieve the text in the server
2. Update the text in the server
3. Close
1
Received 'default text'

What do you want to do?
1. Retrieve the text in the server
2. Update the text in the server
3. Close
2

What text do you want to update with?
Hello world!
Received 'Text changed!'

What do you want to do?
1. Retrieve the text in the server
2. Update the text in the server
3. Close
1
Received b'Hello world!'

What do you want to do?
1. Retrieve the text in the server
2. Update the text in the server
3. Close
3

Socket closed from client
```

tcp-server.py

```
damiafuentes$ python3 tcp-server.py
Starting tcp server
Listening on 127.0.0.1:65432
Accepted connection from 127.0.0.1:54094

Received b'{"option": 1}'
Text returned: default text

Received b'{"option": 2, "text": "Hello world!"}'
Text updated to: Hello world!

Received b'{"option": 1}'
```

```
Text returned: Hello world!

Received b'{"option": 3}'
Socket closed from server
```

Tips:

- The communication between server and client can be done with JSON data encoded in bytes. For that you may use in your server:

  ```
  mydict = json.loads(data.decode('utf-8'))
  ```

  to transform from bytes to string to JSON. And in your client:

  ```
  dataToSend = json.dumps(mydict).encode('utf-8')
  ```

  to transform from a dictionary to JSON to bytes. Now this data can be sent using the `sendall(dataToSend)` method.

- In order to avoid the following error:

  ```
  OSError: [Errno 48] Address already in use
  ```

  You may call the following method in the socket before calling `bind()` in your server:

  ```
  s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
  ```

If the exercise does not say which name to save the file. Submit your code files as *hm9_name_surname_ex_num.py*, where *num* is the exercise number 1, 2, etc. Comment everything so we know you wrote the code! On top of your files write this multiline comment with your information:

"""

Homework 1, Exercise 1 (or 2...)
Name
Date
Description of your program.

"""