MultiThreading

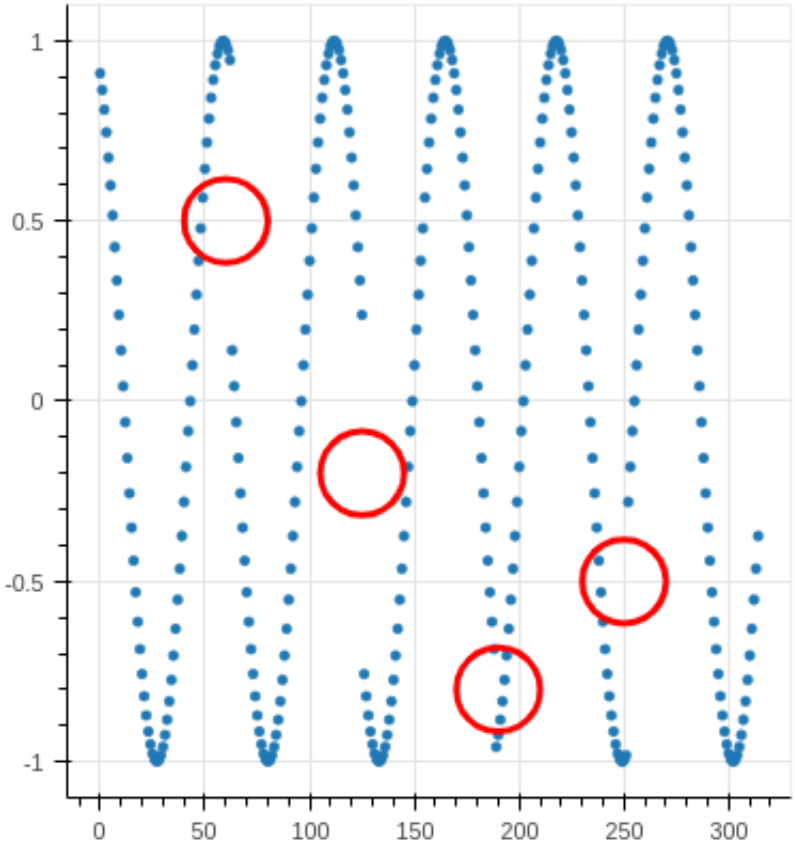
August 21, 2019

1 Introduction

Let's start with an example. Imagine we want to acquire a signal and process the data aquired. So we've a Python script like the following one:

[ ]:
```
while True:
    raw_data=acquire()
    processed_data=process(raw_data)
```

The function *acquire()* collect the input and the function processes the data.
If we try to plot the data collected we may found something like in this picture



You can see how some samples are lost by the *acquire* function. This happens because the acquisition isn't continuos: after the first cycle of the signal is a collected some preprocessing is done, this operations take time and cause the delay in the acquisition

1.1 Multi-threading with the *threading* module

To avoid this kind of issues we can use the multithreading capabilities of Python with the module *threading*. There are also other tools to achieve the same results but we will just use this one. To create our custom thread class we want to use the code below:

[3]:
```
import threading
class MyThread(threading.Thread):
    def __init__(self, threadID):
        threading.Thread.__init__(self)
        #Setup thread
        self.threadID = threadID
    def run(self):
        pass
        #Do something
```

Inside the *init* method we will the define the parameters of the thread while inside the *run* method we will specify the instructions that the trade will execute

2 Exercises

2.1 Exercise 1

In this exercise we want to test if we can improve the the speed of a little piece of code by using multi-threading. We want to execute a GET request to a list of websites in two ways: 1. using a simple for cycle 2. using separate threads for each request We want to measure the execution time of both of the solutions to check which of them is the fastest

Tips and tricks 1. You can use this list of website :["http://yahoo.com", "http://google.com", "http://amazon.com", "http://ibm.com", "http://apple.com","https://www.microsoft.com","https://www.youtube.com/" ,"https://www.polito.it/" ,"http://www.wikipedia.org","https://www.reddit.com/","https://www.adobe.com 2. To measure the execution time you can use this code ~ import time start = time.time() #Code to measure stop=time.time() execution_time=stop-start ~

2.2 Exercise 2

In this exercise we want to create a Caesar cipher. We want a function to crypt a word with a random shift and a function to try to decrypt a word. So for example:

[22]:
```
from Exercise2 import *
import string
#Original sentence
sentence='programming for iot'
#Crypted sentence
crypted=crypt(sentence)
print(crypted)
done=False
while not done:
    #we want to test each possible shift, one for each letter of the alphabet
    for i in range(len(string.ascii_lowercase)):
```

```
        #We create the shifted dict/alphabet
        test_dict=create_dict(i)
        test_decrypt=decrypt(crypted,test_dict)
        if test_decrypt==sentence:
            done=True
            print('Shift={}'.format(26-i))
            break
```

```
programming for iot
Shift=26
```

In the file *wordlist.txt* there a list of word we can use. We want to check what's the fastest way to decrypt a list of word: 1. a "classic" for cycle 2. using separate threads **Tips and tricks** 1. Pay attention that some words in the list may have spaces! 2. Don't try immediatly with all the words in the file, the execution may take some time

[ ]: