2019/20

Programação Paralela e Distribuída

Prática 3: Python Threading

(https://docs.python.org/3/library/threading.html)

(https://www.tutorialspoint.com/python/python_multithreading.htm)

1 - Pretende-se uma aplicação para gerir o dinheiro em **caixa de um clube recreativo**. Para isso construa as seguintes classes:

a) Uma classe *ContaBancaria* que deverá permitir:
- Consultar o saldo disponível em cada instante;
- Simular um levantamento, cada vez que um utilizador pretenda levantar uma quantia menor ou igual à existente;
- Simular um depósito.

b) Uma classe *Financiador* que periodicamente vai depositando quantias num objeto do tipo ContaBancaria.

c) Uma classe Utilizador que periodicamente vai levantando quantias do objeto do tipo ContaBancaria.

d) Para testar as classes anteriores construa uma classe teste em que um objeto do tipo *ContaBancaria* seja partilhado concorrentemente por um objecto do tipo *Financiador* e por pelo menos 3 objectos do tipo *Utilizador*.

e) Altere a classe *ContaBancaria* para suspender o Utilizador sempre que o saldo da conta for inferior ao valor do levantamento.

2 – Pretende-se uma aplicação que verifique se um conjunto de websites estão ou não ativos. A função que se segue permite aceder a um determinado endereço (address) e caso o código de erro devolvido seja superior a 400, é gerada uma exceção.

```
import requests

def check_website (address, timeout = 5):
    try:
        response = requests.head(address , timeout = timeout)
        if response.status_code >= 400:
            raise Exception ()
    except Exception:
        print ( "Error: " + str (address) + " returns code: " + str(response.status_code) )
```

a) Defina uma função, worker(), que num ciclo infinito obtenha um URL de um objeto do tipo Queue e invoque a função check_website para esse endereço.

b) Construa um programa que:

i) Coloque num objeto do tipo queue.Queue uma lista de URLs. Abaixo apresentam-se alguns exemplos, deve acrescentar outros.

ii) Cria e inicie a execução de um conjunto de threads cujo *target* será a função worker().

c) Teste o programa para 1, 2, 4 e 8 threads e analise os tempos de execução.

```
## Lista de URLs:
WEBSITE_LIST = [
'http://envanto.com',
'http://amazon.co.uk',
'http://amazon.com',
'http://facebook.com',
'http://google.com',
'http://google.fr',
'http://google.es',
'http://bing.com',
'http://google.co.uk',
'http://internet.org',
'http://gmail.com',
'http://stackoverflow.com',
'http://github.com',
'http://shopyfy.com',
'http://instagram.com',
'http://youtube.com',
]
```

3 – As listagens que se seguem permitem implementar um servidor multi-treaded em Python (retirado de http://net-informations.com/python/net/thread.htm) :

```
#Client:
import socket
SERVER = "127.0.0.1"
PORT = 2000
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((SERVER, PORT))
client.sendall(bytes("This is from Client",'UTF-8'))
while True:
    in_data = client.recv(1024)
    print("From Server :" ,in_data.decode())
    out_data = input()
    client.sendall(bytes(out_data,'UTF-8'))
    if out_data=='bye':
        break
client.close()
```

```
#Thread que no servidor comunica com o cliente:
import socket, threading
class ClientThread(threading.Thread):
    def __init__(self,clientAddress,clientsocket):
```

```
        threading.Thread.__init__(self)
        self.csocket = clientsocket
        print ("New connection added: ", clientAddress)

    def run(self):
        print ("Connection from : ", clientAddress)
        msg = ''
        while True:
            data = self.csocket.recv(2048)
            msg = data.decode()
            if msg=='bye':
                break
            print ("from client", msg)
            self.csocket.send(bytes(msg,'UTF-8'))
        print ("Client at ", clientAddress , " disconnected...")
```

```
#Server:
LOCALHOST = "127.0.0.1"
PORT = 2000
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
server.bind((LOCALHOST, PORT))
print("Server started")
print("Waiting for client request..")
while True:
    server.listen(1)
    clientsock, clientAddress = server.accept()
    newthread = ClientThread(clientAddress, clientsock)
    newthread.start()
```

- Implemente e teste o código anterior

4 - Usando o modelo anterior para uma aplicação cliente servidor, pretende-se um jogo em que:

- Cada cliente aposta num número inteiro de 0 a 99;
- O Servidor gera um número inteiro aleatório entre 0 e 99 e se for igual à aposta do cliente este ganha um determinado prémio. Note que para cada cliente deve ser gerado um novo número.
- O valor do prémio é determinado da seguinte forma: é inicializado a 0; para cada cliente que joga é adicionado ao prémio o valor de 1€. Quando um cliente acerta no número gerado ganha 50% do valor acumulado até ao momento no prémio. Após um cliente acertar no prémio, o prémio volta a zero.
- O Servidor devolve uma mensagem com o texto " Parabéns, ganhou XXX €" caso acerte e em que XXX deve conter o valor ganho. Caso o cliente não acerte no valor gerado a mensagem deverá ser "Continue a tentar, o prémio já é de YYY €" em que YYY é o valor que poderia ganhar nesse momento.

Nota: Tem uma solução em java em:
https://www.di.ubi.pt/~pprata/spd/SD_19_20_T03a-exerc%C3%ADcio.pdf