UDP Server v2.0

Created on Mon Mar 28 15:40:47 2022

@author: Conrado

Version: 2.0

What it is: UDP Server

What it does:

```
1) Creates a Socket with certain parameters
2) Executes 3 parallel-independent threads:
    #Thread-1 - keeps listening at UDP assigned Port
    #Thread-2 - generate random STR and INT
    #Thread-3 - does something after x amount of time
```

1 - Imports and Parameters

In [ ]:

```python
import socket
import threading
import ClientOrderFunctions as COF

#Connection Parameters:
localIP = "127.0.0.1"
port = 54321
bufferSize = 1024


msgFromServer = "Hello UDP Client" #Standard REPLY Message
bytesToSend = str.encode(msgFromServer) #Encode String to Bytes
```

2 - Create Socket and Assign Port and Address

In [ ]:

```python
# Create a Socket
UDPServerSocket = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)

# Bind to Address and IP
UDPServerSocket.bind((localIP, port))

print("UDP Server READY!\n")
```

3 - Threads Functions

3.1 - Thread 1

In [ ]:

```python
#Thread-1

# Listen for incoming messages
def listenIO_handler(SSocket):
    '''
    Handler that allows constant Port listening, and other functions if needed.
    Handler helps killing the thread.

    [INPUT]: UDP Server Socket (SSocket) Object

    '''
    global flag, data
    while True:
        flag, data = listenIO(SSocket)


    if flag:
        print("Valid Order Arrived!")

    return
```

In [ ]:

```
def listenIO(SSocket):
    '''
    [INPUT]: UDP Server Socket (SSocket) Object
    [OUTPUT]:
        -> True and clientMsg if "<order>" exists
        -> False and "" if "<order>" doesn't exist
    '''
    try:
        bytesAddressPair = SSocket.recvfrom(bufferSize)

        message = bytesAddressPair[0]
        address = bytesAddressPair[1] #tuple (address, port)

        #-------------------------------------------------------------------------
        #clientMsg = "\t> Message Received from Client:{}".format(message)
        clientMsg = "\t> Message Received from Client: " + message.decode("utf-8")
        clientIP  = "\t> Client IP Address:{}".format(address)

        print(clientMsg)
        print(clientIP)
        print("\n")
        #-------------------------------------------------------------------------

        # Sending a REPLY to client
        UDPServerSocket.sendto(bytesToSend, address)


        if (COF.numberOfOrders(clientMsg) > 0):
            #Extract number of Orders inside the Client Message
            return True, clientMsg
        else:
            return False, ""

    except:
        return False, ""
```

3.2 - Thread 2

In [ ]:

```
#Thread-2
import string
import random

def generateRnd_handler(length):
    '''
    Handler that allows constant rand. String and rand. Integer production
    Handler helps killing the thread.
    [INPUT]: Length [INT] of the desired String
    '''
    while True:
        generateRndSTR(length)

    return
```

In [ ]:

```
def generateRndSTR(length):
    #This is just a Dummy Function to assign to a Thread
    '''
    If the Random Integer is inside a certain interval, print the random local variables
    [INPUT]: Random String Length [INT]
    [OUTPUT]: -
    '''
    characters = string.ascii_letters + string.digits
    randomSTR = ''.join(random.choice(characters) for i in range(length))
    randomINT = random.randint(0,100000)

    if (randomINT == 0 or randomINT == 1):
        print("%d > %s" % (randomINT, randomSTR))

    return
```

3.3 - Thread 3

In [ ]:

```
#Thread-3
import time

def reminder_handler(delay):
    '''
    Handler that does something (print string) with a certain delay
    Handler helps killing the thread.
    [INPUT]: delay [INT] in seconds
    '''
    while True:
        reminder(delay)

    return
```

In [ ]:

```
def reminder(dt):
    '''
    Put execution (in our case the Thread) to sleep for a certain amount of seconds
    [INPUT]: delay [INT]
    '''
    time.sleep(dt)
    print("\n[!!!!] Don't forget to save your work!\n")
    return
```

4 - Assign and Start Threads

In [ ]:

```
#Assign Threads:
thread1 = threading.Thread(target = listenIO_handler, args=[UDPServerSocket])
thread2 = threading.Thread(target = generateRnd_handler, args=[4])
thread3 = threading.Thread(target = reminder_handler, args=[4])

#Start Threads:
thread1.start()
thread2.start()
thread3.start()
```