

Trabalho de Aprofundamento 1, Grupo 7

Universidade de Aveiro

Alexandre Martins, Tomás Rodrigues



universidade de aveiro
theoria poiesis praxis

Trabalho de Aprofundamento 1, Grupo 7

Departamento de Eletrónica, Telecomunicações e
Informática (DETI)

Universidade de Aveiro

Alexandre Martins, Tomás Rodrigues
(103552) alexandremartins@ua.pt, (104090) tcercarodrigues@ua.pt

31 de maio de 2022

Agradecimentos

Professor Auxiliar António Manuel Adrego da Rocha, pelas excelentes aulas sobre Python e Sockets, e esclarecimento de dúvidas, assim como pelas aulas do semestre passado, sem as quais não conseguiríamos ter completado este relatório.

Índice

1	Introdução	1
2	Metodologia	2
2.1	Início do serviço	2
2.2	Operações	3
2.2.1	NUMBER	3
2.2.2	STOP	4
2.2.3	QUIT	5
3	Testes	6
3.1	Pré-conexão entre client e server	6
3.2	Funcionamento após a conexão entre client e server	6
3.2.1	Com devolução de valores	6
3.2.2	Sem devolução de valores	7
4	Conclusão	9

Capítulo 1

Introdução

Este trabalho, desenvolvido no âmbito da unidade curricular de Laboratórios de Informática, constou na criação um servidor que suporte a análise estatística de listas de números inteiros. Mais concretamente que determine, o mínimo e o máximo números de uma lista de números inteiros fornecidos pelos clientes.

O projeto do gitHub associado a este trabalho pode ser consultado em <https://github.com/AleexMaartins/Currently-working-on>

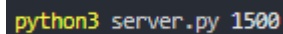
Capítulo 2

Metodologia

Apresentação e descrição da metodologia utilizada para a realização do projeto e obtenção de resultados

2.1 Início do serviço

Deve ser chamado o **server.py** no formato: **python3 server.py porto**

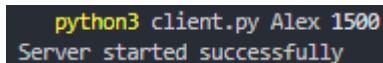


```
python3 server.py 1500
```

Figura 1: Exemplo da invocação do server.py

Após a sua iniciação o servidor imprime no terminal uma mensagem de ***Running...*** e aguarda que um **Client** se conecte.

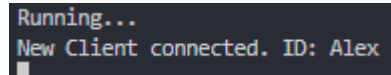
Quando o servidor estiver a correr deve agora ser chamado o **client.py** no formato: **python3 client.py <client_id><porto>**. O **client_id** pode variar desde que não haja dois iguais conectados ao servidor ao mesmo tempo. O porto tem de ser o mesmo iniciado no **server.py**. Como não é chamado um **ip**, é assumido o **host** como **localhost**.



```
python3 client.py Alex 1500
Server started successfully
```

Figura 2: Exemplo da invocação do client.py e mensagem de conexão bem sucedida

Após feita a conexão é imprimindo posteriormente, no terminal do server.py, *New Client connected. ID: <Client_id>*

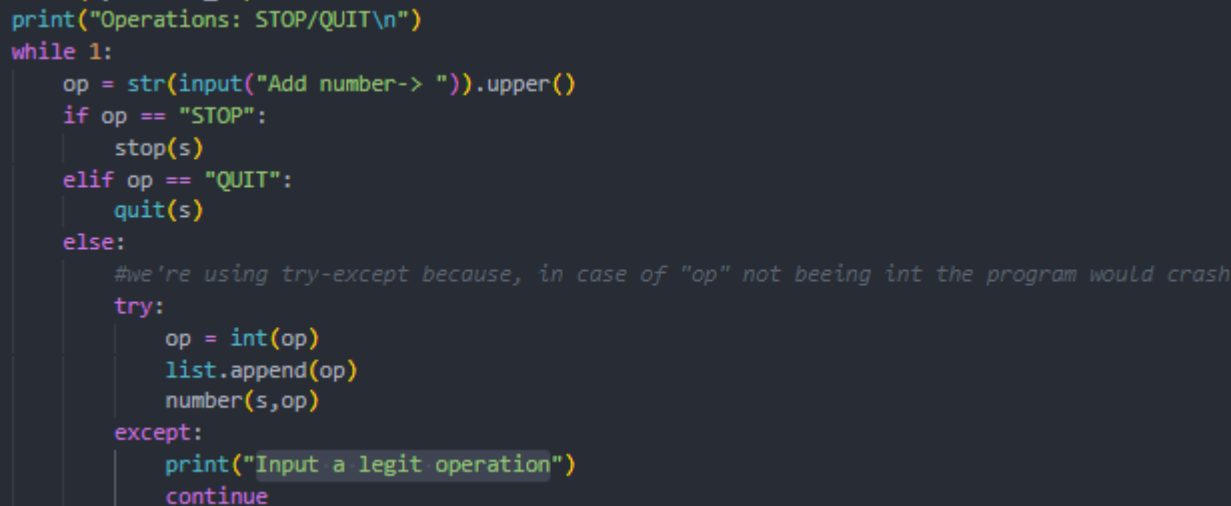


```
Running...
New Client connected. ID: Alex
```

Figura 3: Terminal do server.py após um client se conectar

2.2 Operações

Existem 3 operações possíveis após a conexão ser realizada. **STOP**, **QUIT** e no caso de nenhuma destas ser chamada, e caso o valor introduzido seja um inteiro é assumida a terceira **NUMBER**. No caso do input do **client** não ser nenhum destes é devolvido uma mensagem de erro, e permite ao **client** colocar um novo input sem sair do programa.



```
print("Operations: STOP/QUIT\n")
while 1:
    op = str(input("Add number-> ")).upper()
    if op == "STOP":
        stop(s)
    elif op == "QUIT":
        quit(s)
    else:
        #we're using try-except because, in case of "op" not beeing int the program would crash
        try:
            op = int(op)
            list.append(op)
            number(s,op)
        except:
            print("Input a legit operation")
            continue
```

Figura 4: Código do **client.py** que lê a operação indicada pelo **client**

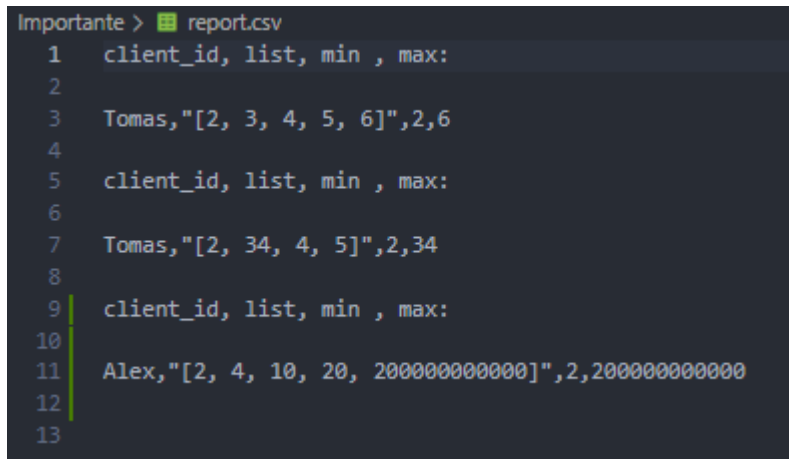
2.2.1 NUMBER

Enquanto o valor introduzido for diferente de **STOP** ou **QUIT** lê os valores inteiros adicionados pelo **client** e atribui-os a uma lista associada ao **client_id**

2.2.2 STOP

Apenas pode ser chamada após serem adicionados valores à lista através da operação **NUMBER**. Caso contrário é devolvida uma mensagem de erro e permite ao **client** colocar um novo input sem sair do programa.

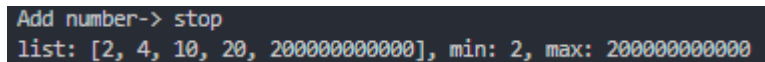
Quando a string **STOP** for chamada pelo **client**, o **server** calcula o valor **mínimo** e **máximo** da lista obtida em **NUMBER** e imprime num ficheiro csv, através da função *create_file*, o **client_id**, seguido da **lista** toda, do **mínimo** e **máximo**.



```
Importante > report.csv
1  client_id, list, min , max:
2
3  Tomas,"[2, 3, 4, 5, 6]",2,6
4
5  client_id, list, min , max:
6
7  Tomas,"[2, 34, 4, 5]",2,34
8
9  client_id, list, min , max:
10
11 Alex,"[2, 4, 10, 20, 200000000000]",2,200000000000
12
13
```

Figura 5: Exemplos da estrutura imprida no ficheiro csv

No terminal do **client.py** aparece a mesma mensagem impressa no ficheiro csv.



```
Add number-> stop
list: [2, 4, 10, 20, 200000000000], min: 2, max: 200000000000
```

Figura 6: Exemplo da estrutura imprida no terminal do **client.py**

Os dados do **client_id** são finalmente eliminados dos dicionários **users** e **clients_aux** com auxílio da função *clean_client*. E posteriormente desconectar o **client.py** do **server.py**.


```
def clean_client(client_sock):  
    client_id = clients_aux.get(client_sock)  
    users.pop(client_id)  
    clients_aux.pop(client_sock)
```

Figura 7: Função *clean_client*

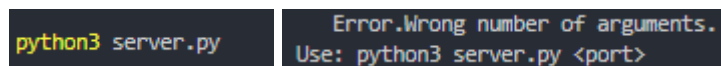
2.2.3 QUIT

A operação **QUIT** serve como uma forma do **client.py** se desconectar do **server.py**. O programa termina, e assim como na operação **STOP** os dados do **client_id** são finalmente eliminados dos dicionários **users** e **clients_aux** com auxílio da função *clean_client*. A diferença entre **QUIT** e **STOP** é que **QUIT** não imprime valores no terminal, nem no ficheiro csv. Servindo apenas para terminar o processo caso não seja pretendido alterar os valores do ficheiro csv.

Capítulo 3

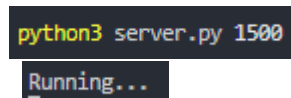
Testes

3.1 Pré-conexão entre client e server



```
python3 server.py      Error:Wrong number of arguments.  
                        Use: python3 server.py <port>
```

Figura 8: Terminal do **server**: Tentativa de criar server.py sem port e mensagem de erro.



```
python3 server.py 1500  
Running...
```

Figura 9: Terminal do **server**: Tentativa de criar server.py com port e mensagem de sucesso.

3.2 Funcionamento após a conexão entre client e server

3.2.1 Com devolução de valores

-> NUMBER -> STOP

```

Server started successfully
Operations: STOP/QUIT

Add number-> 2
Number added successfully
Add number-> 4
Number added successfully
Add number-> 10
Number added successfully
Add number-> 20
Number added successfully
Add number-> 20000000000
Number added successfully
Add number-> sstop
Input a legit operation
Add number-> stop
list: [2, 4, 10, 20, 20000000000], min: 2, max: 20000000000

```

Figura 8: Terminal do **client**

```

client_id, list, min , max:

Alex,"[2, 4, 10, 20, 20000000000]",2,20000000000

```

Figura 9: **report.csv**

3.2.2 Sem devolução de valores

-> NUMBER -> QUIT

```

Server started successfully
Operations: STOP/QUIT

Add number-> 2
Number added successfully
Add number-> 4
Number added successfully
Add number-> 10
Number added successfully
Add number-> 5
Number added successfully
Add number-> quit
Client quitted successfully

```

Figura 10: Terminal do **client**: desconecta com sucesso

-> STOP antes de NUMBER

```
Server started successfully
Operations: STOP/QUIT

Add number-> stop
Insuficient Data Make sure to add numbers first
```

Figura 11: Terminal do **client**: invoca um erro

-> QUIT antes de NUMBER

```
Server started successfully
Operations: STOP/QUIT

Add number-> quit
Client quitted successfully
```

Figura 12: Terminal do **client**: desconecta com sucesso

-> Operação diferente das pedidas

```
Server started successfully
Operations: STOP/QUIT

Add number-> arroz
Input a legit operation
```

Figura 13: Terminal do **client**: invoca um erro

Capítulo 4

Conclusão

Este trabalho de aprofundamento permitiu-nos consolidar a implementação de sockets, manipulação de estruturas de ficheiros ou mensagens json e csv, e também nos forneceu com mais conhecimento no geral da linguagem de programação Python.

Contribuições dos autores

O trabalho foi feito presencialmente na totalidade pelo que cada um contribuiu um pouco para todas as partes, com o Alexandre a fazer um pouco mais. Desta forma concordamos que o Alexandre merece 60% e o Tomas 40%

Acrónimos

DETI Departamento de Eletrónica, Telecomunicações e Informática