

## Redes de Computadores I (CK0249) 2020.1 - PPE

Prof. Dr. Emanuel Bezerra Rodrigues

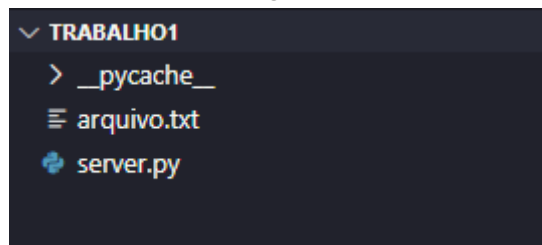
Nome: Daniele Carnaúba Gonçalves Matrícula:473257

### ATIVIDADE PRÁTICA PROGRAMAÇÃO COM SOCKETS

#### TAREFA 1 - SERVIDOR HTTP

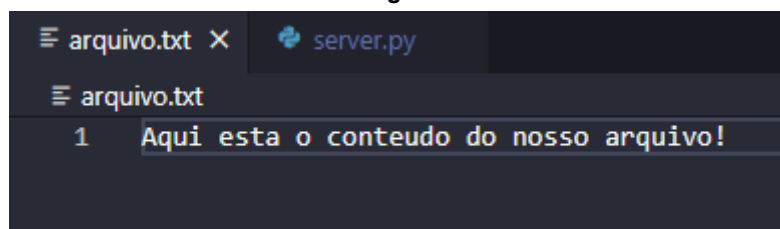
Primeiro vamos criar os arquivos necessários. Na Imagem 1 podemos ver o arquivo.txt que será usado pelo servidor web e também o server.py onde será montado nosso socket.

Imagem 1



Na Imagem 2 podemos conferir o conteúdo que será emitido pelo servidor no navegador que atuará como cliente.

Imagem 2



Na Imagem 3 podemos conferir toda a estrutura do socket montada. Agora vejamos cada parte do código. Na **linha 1** vamos importar o socket. Nas **linhas 3 e 4** vamos inicializar variáveis, primeiro vamos criar socket que realiza conexões TCP com a variável serverSocket, AF\_INET é usado pois vamos utilizar o protocolo IPv4 e o SOCK\_STREAM é usado pois vamos utilizar o protocolo TCP. A segunda variável é a serverPort, que vai indicar por qual porta vamos nos comunicar, nesse caso a porta escolhida foi 6789. Nas **linhas 6 e 7** vamos conectar o socket ao endereço e a porta do servidor. Indicamos na tupla primeiro o endereço e depois a porta. Logo em seguida definimos a quantidade de conexões que o servidor irá fazer por vez, no caso abaixo, apenas uma conexão por vez.

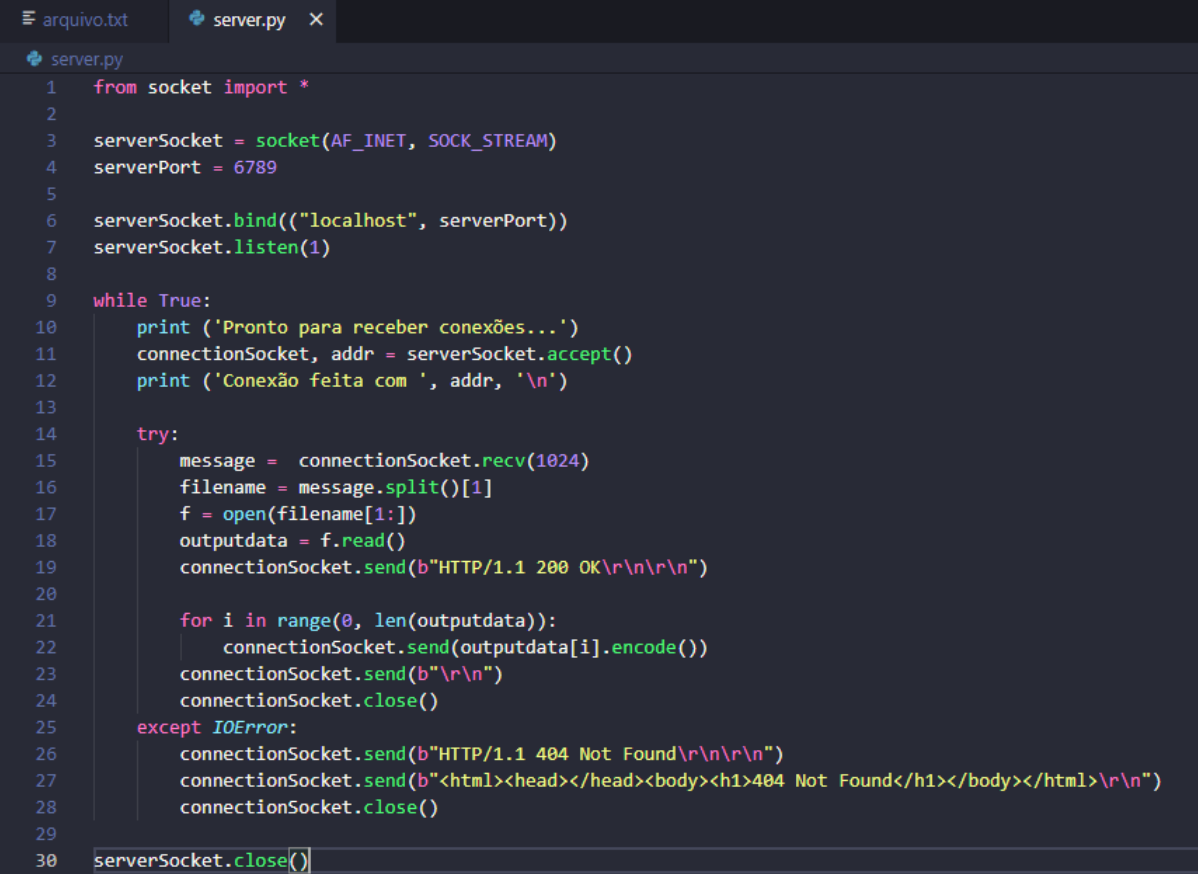
Agora o servidor está pronto e disponível para realizar conexões. Na estrutura da **linha 9 a 28** temos um laço while True. A cada nova interação no laço, uma nova conexão é feita. Na **linha 11** vamos configurar uma nova conexão com o cliente, recebendo o endereço da mensagem e o endereço do cliente. Na estrutura

try e except das **linhas 14 a 28** vamos verificar se será possível realizar a conexão ou não, se sim, try é executado, e não for possível, except é executado.

Primeiro vamos analisar o try. Na **linha 15** recebemos e guardamos na variável message a mensagem de pedido do cliente. Com isso, devemos extrair o caminho do objeto que é solicitado pelo cliente na mensagem, isso é feito nas **linhas 16 e 17** onde guardamos a segunda parte da mensagem em filename. A segunda parte representa o caminho que se deseja extrair. Na variável filename temos o caminho do objeto, porém esse caminho vem acompanhado na primeira posição do caractere \. Por isso na **linha 17** realizamos a leitura do arquivo a partir da posição 1, ou seja, a partir do segundo caractere e na **linha 18** armazenamos o conteúdo do arquivo em um buffer temporário. Na **linha 29** emitimos uma mensagem HTTP confirmando que foi encontrado o arquivo que será enviado. Entre as **linhas 21 e 24** temos um for que envia a mensagem do arquivo para o socket e ao final fechamos a conexão com o cliente.

O except na **linha 25** vai cuidar das exceções, ou seja, caso o arquivo não seja encontrado, é emitida uma mensagem de resposta HTTP e é finalizada a conexão com o cliente.

Imagem 3



```
server.py
1  from socket import *
2
3  serverSocket = socket(AF_INET, SOCK_STREAM)
4  serverPort = 6789
5
6  serverSocket.bind(("localhost", serverPort))
7  serverSocket.listen(1)
8
9  while True:
10     print ('Pronto para receber conexões...')
11     connectionSocket, addr = serverSocket.accept()
12     print ('Conexão feita com ', addr, '\n')
13
14     try:
15         message = connectionSocket.recv(1024)
16         filename = message.split()[1]
17         f = open(filename[1:])
18         outputdata = f.read()
19         connectionSocket.send(b"HTTP/1.1 200 OK\r\n\r\n")
20
21         for i in range(0, len(outputdata)):
22             connectionSocket.send(outputdata[i].encode())
23         connectionSocket.send(b"\r\n")
24         connectionSocket.close()
25     except IOError:
26         connectionSocket.send(b"HTTP/1.1 404 Not Found\r\n\r\n")
27         connectionSocket.send(b"<html><head></head><body><h1>404 Not Found</h1></body></html>\r\n")
28         connectionSocket.close()
29
30 serverSocket.close()
```

A Imagem 4 exibe o terminal com a execução do código server.py. Observe que o servidor está pronto para inicializar novas conexões:

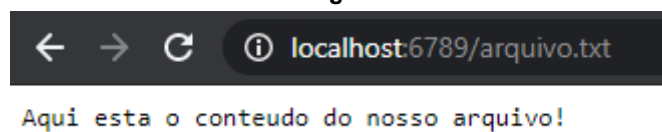
Imagem 4

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

IndexError: list index out of range
PS C:\Users\Jorio\Desktop\REDES\Trabalho1> python3 server.py
Pronto para receber conexões...
█
```

Observando a Imagem 5 veja que após a inicialização do servidor, indo ao navegador e acessando a URL: localhost:6789/arquivo.txt teremos a representação visual da resposta do servidor que emite na tela o conteúdo do arquivo.txt:

Imagem 5



E finalmente na Imagem 6 é possível visualizar o que aconteceu após o acesso da URL acima. Note que o servidor exibe uma mensagem de conexão realizada com sucesso com o cliente, além de imprimir o endereço do cliente e seu id de conexão que varia a cada nova conexão, pois estamos realizando uma conexão não persistente.

Imagem 6

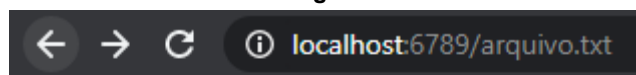
```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

PS C:\Users\Jorio\Desktop\REDES\Trabalho1> python3 server.py
Pronto para receber conexões...
Conexão feita com ('127.0.0.1', 53805)

Pronto para receber conexões...
Conexão feita com ('127.0.0.1', 53806)
█
```

A imagem abaixo mostra a resposta enviada ao cliente caso o except seja executado, ou seja, ele não encontrou o arquivo.txt que foi excluído para realizar este teste.

Imagem 7



**404 Not Found**