

**ESTI – ESCOLA SUPERIOR DA TECNOLOGIA DA INFORMAÇÃO**

**GRADUAÇÃO EM ANÁLISE E DESENVOLVIMENTO**

**DE SISTEMAS**

**PROJETO DE BLOCO**

Caio M. Veloso Dias

Rio de Janeiro, 22 de Setembro de 2018.

Sumário

[**Introdução**](#_80chwz153yrt) **3**

[**Desenvolvimento do projeto**](#_ibo92nnj4hpj) **4**

[**Resultados e Análise**](#_3ckdy3i7of5n) **31**

[**Conclusão**](#_2f89ineajbu3) **42**

# Introdução

O projeto foi pensado de forma a facilitar o usuário a ter informações de memória, cpu, disco, arquivos, diretório, processos e redes de forma clara e de fácil domínio, para usuários com pouca familiaridade com o assunto.

O projeto consiste em um sistema básico escrito em python, baseado em comunicação via socket, cliente/servidor, comunicação essa estabelecida através do protocolo TCP.

Da parte do cliente, responsável por receber a entrada de instruções do usuário, e formado por um menu de dez opções, a cada opção escolhida o cliente informa ao servidor qual informação ele quer e após uma busca feita pelo servidor, essa informação é entregue ao cliente e o mesmo apresenta ao usuário.

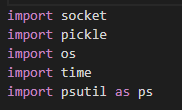
# Desenvolvimento do projeto

O projeto foi desenvolvido na linguagem python em sua versão 3.6, foram utilizadas as bibliotecas socket, pickle, cpuinfo, psutil e OS para auxiliar o desenvolvimento. Toda a comunicação do projeto foi feita baseada em socket por isso foi dividido em dois arquivos, client e server e foi escolhido o protocolo TCP para a comunicação com a intenção de prevenir perdas de pacotes e uma comunicação mais instável.

Cliente:

Os imports das dependências necessárias para o projeto, estarei disponibilizando a documentação das mesmas

[socket](https://wiki.python.org.br/SocketBasico), [pickle](https://docs.python.org/3/library/pickle.html), [os](https://docs.python.org/2/library/os.html), [time](https://docs.python.org/3/library/time.html), [psutil](https://psutil.readthedocs.io/en/latest/)



O projeto possui uma única constante, GIGAFACTOR, que é responsável por ter a operação auxiliar para conversão de KB para GB.

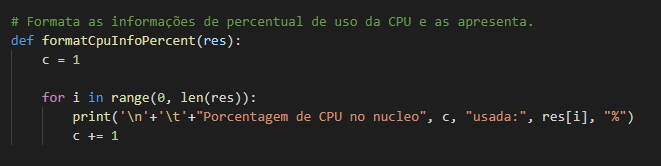


O operador binário [***<<***](https://wiki.python.org.br/BitwiseOperators)tem a função de elevar 1 por 30.

Foram externalizadas algumas funcionalidades da parte do cliente em funções devido a simplicidade do código e melhor entendimento do mesmo.

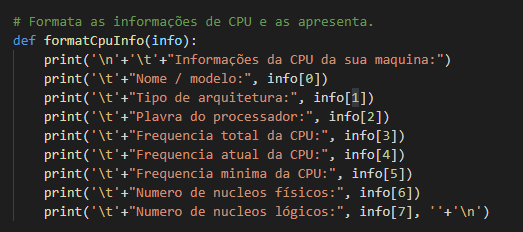
***Das funções:***

formatCpuInfoPercent()



Tem a finalidade de formatar e apresentar o dado recebido do servidor referente a ao percentual de uso da CPU.

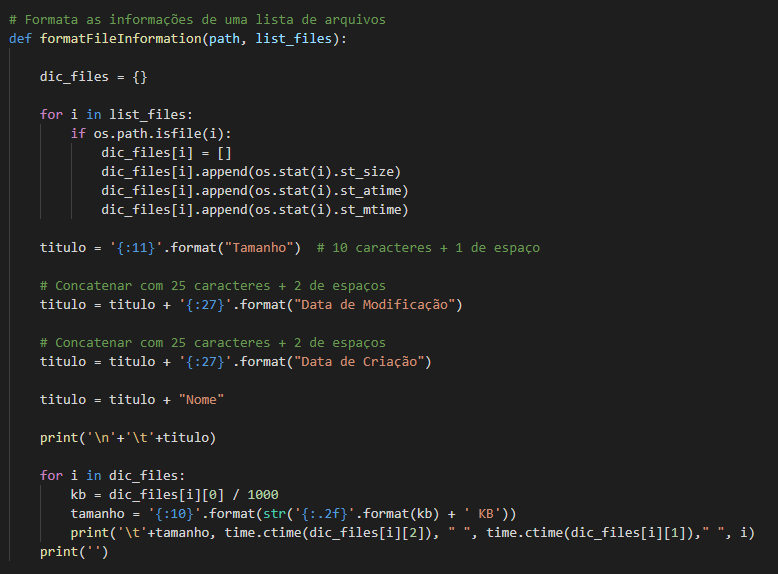
Recebe uma propriedade que representa o percentual de uso, e tem uma variável **c** que funciona como contadora para a quantidade de núcleos possuidos pela CPU.

formatCpuInfo()

Tem a finalidade de formatar e apresentar as informações recebidas do servidor referente

a CPU, nome, arquitetura, número de núcleos, etc.

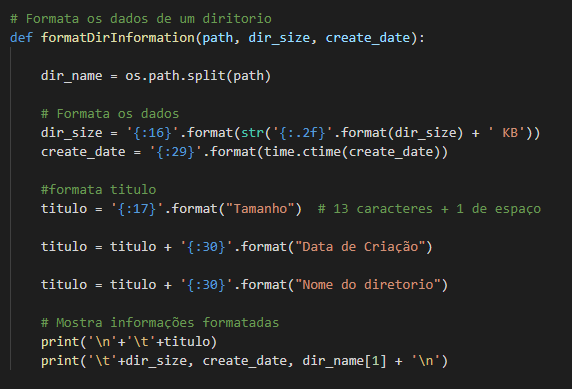
Recebe uma propriedade que representa uma lista com os dado da CPU já dispostos no index da lista ordenados de acordo com a disposição das informações.

getFileInformation()

Tem a finalidade de formatar e apresentar os dados recebidos do servidor referente a arquivos.

Recebe dois parâmetros ***path*** representando o caminho escolhido pelo usuário e ***list\_files*** que representa a lista de arquivos encontrados no servidor com base no caminho que o usuário forneceu.

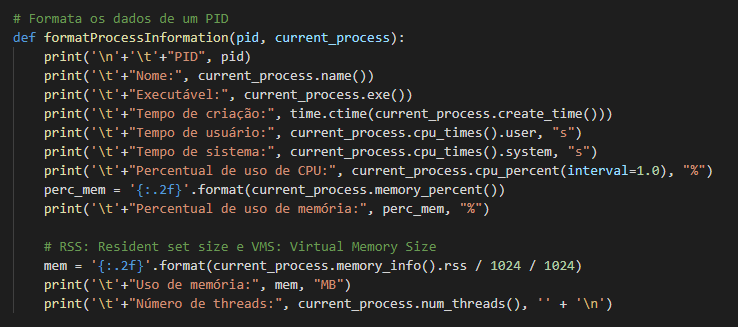
possui uma variável em seu escopo, ***dic\_files*** que representa o dicionario de arquivos que será exibido ao usuário no final.

formatDirInformation()

Tem a finalidade de formatar e apresentar os dados recebidos do servidor referente a diretórios.

Recebe três parâmetros, ***path*** representando o caminho do diretório escolhido pelo usuário, ***dir\_size*** que representa o tamanho do diretório escolhido pelo usuário e ***create\_date*** representando a data de criação do diretório fornecido pelo usuário.

Possui uma variável em seu escopo, ***dir\_name*** que receberá o nome do diretório através do retorno do método ***split()*** passando o path escolhido pelo usuário.

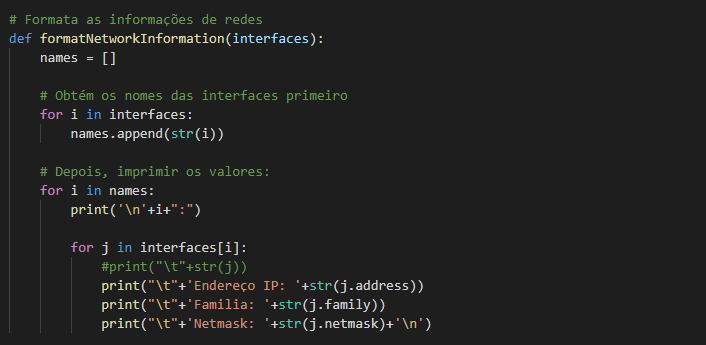
formatProcessInformation()

Tem a finalidade de formatar e apresentar os dados recebidos do servidor referente a um processo, processo esse capturado através do PID encontrado pelo servidor.

Recebe dois parâmetros, ***pid*** representa o PID do processo escolhido pelo usuário e ***current\_process*** que representa o resultado da captura das informações do processo feita pelo servidor.

Possui uma variável em seu escopo, ***perc\_mem*** que formata para apresentar o percentual de uso de memória que tem o processo escolhido.

formatNetworkInformation()



Tem a finalidade de formatar e apresentar os dados recebidos do servidor referente as interfaces de rede em que a máquina se encontra.

Recebe um único parâmetro, ***interfaces*** que representa a lista de interfaces que o servidor buscou e encontrou.

Possui uma variável em seu escopo, ***names*** ela recebe os nomes das interfaces existentes para melhor formatação das informações e exibição para o usuário.

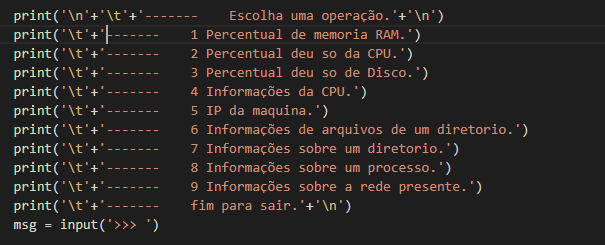
Foi escolhido uma estrutura de try/except para a comunicação com o servidor e uma estrutura de looping com while para a constância do menu para o usuário.

***Da criação do socket:***

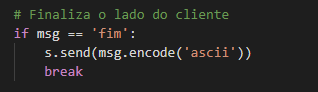
Como dito, foi estabelecida uma conexão via protocolo TCP, utilizando assim a função socket passando os parâmetros de identificação da comunicação, ***family*** = socket.AF\_INET (identifica a família, nesse caso ipv4) e ***type*** = socket.SOCK\_STREAM (identifica o tipo de conexão, nesse caso TCP).



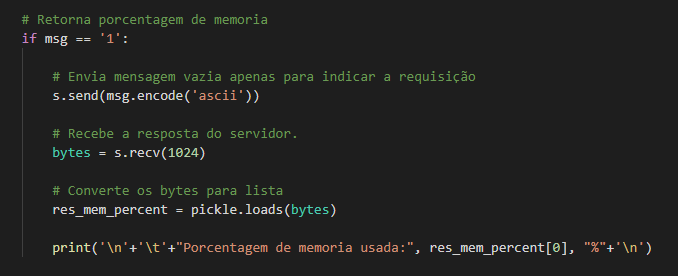
O método ***connect*** do socket tenta estabelecer uma conexão com o servidor, recebe como parâmetro o ***nome do cliente*** (socket.gethostname()) e uma ***porta*** (nesse caso 14562)para estabelecer a conexão, que precisa ser a mesma que a do server.

***Da estrutura de looping:***

Para receber as informações do usuário foi feita uma estrutura de print’s mostrando as opções válidas para o usuário e um input para receber a opção escolhida.

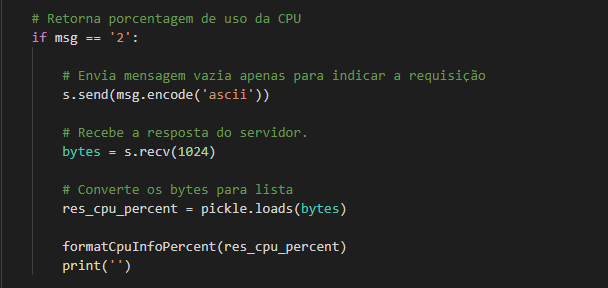


Caso o usuário entre com a string ***“fim”*** , será enviado ao servidor a mensagem de fim que encerra a conexão e encerra o looping do while, encerrando assim a interação do usuário com o sistema.

***Das opções do menu:***

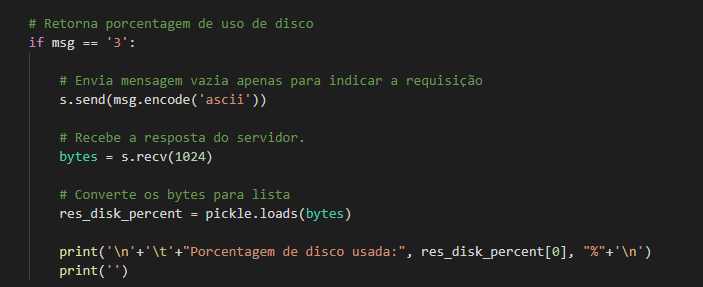
No caso um, será enviado uma mensagem, convertida em bytes, ao servidor indicando qual operação o usuário quer realizar, operação essa que retorna a porcentagem de uso de memória RAM.

A resposta do servidor será convertida de volta através do pickle, uma vez que é usado no servidor o pickle para enviar a mensagem, e será formatada para ser exibida para o usuário.



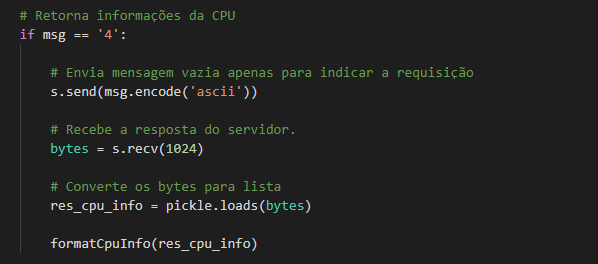
No caso dois, será enviado uma mensagem, convertida em bytes, ao servidor indicando qual operação o usuário quer realizar, operação essa que retorna a porcentagem de uso de CPU atual.

A resposta do servidor será convertida de volta através do pickle, uma vez que é usado no servidor o pickle para enviar a mensagem, e é chamada a função ***formatCpuInfoPercent()*** que formata e exibe a informação para o usuário.



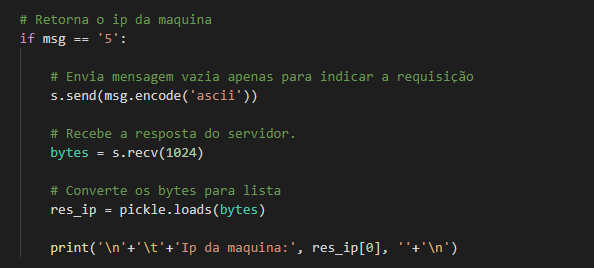
No caso três, será enviado uma mensagem, convertida em bytes, ao servidor indicando qual operação o usuário quer realizar, operação essa que retorna a porcentagem de uso de disco.

A resposta do servidor será convertida de volta através do pickle, uma vez que é usado no servidor o pickle para enviar a mensagem, e será formatada para ser exibida para o usuário.



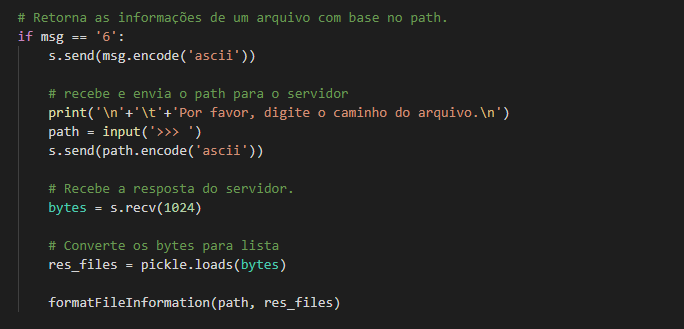
No caso quatro, será enviado uma mensagem, convertida em bytes, ao servidor indicando qual operação o usuário quer realizar, operação essa que retorna informações da CPU.

A resposta do servidor será convertida de volta através do pickle, uma vez que é usado no servidor o pickle para enviar a mensagem, e é chamada a função ***formatCpuInfo()*** que formata e exibe a informação para o usuário.



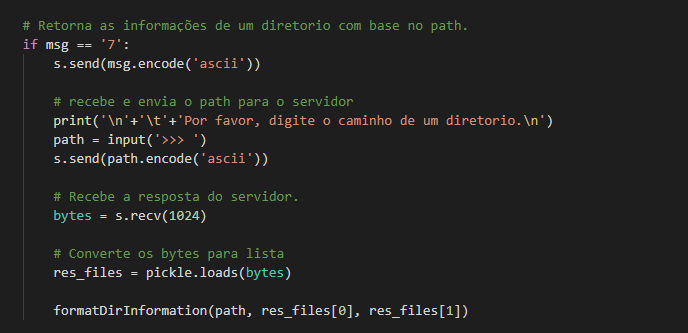
No caso cinco, será enviado uma mensagem, convertida em bytes, ao servidor indicando qual operação o usuário quer realizar, operação essa que retorna o endereço IP da máquina.

A resposta do servidor será convertida de volta através do pickle, uma vez que é usado no servidor o pickle para enviar a mensagem, e será formatada para ser exibida para o usuário.



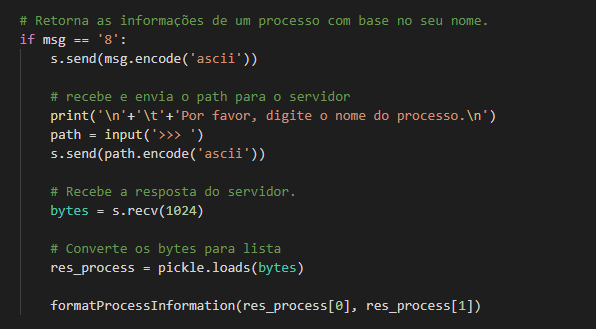
No caso seis, será enviado uma mensagem, convertida em bytes, ao servidor indicando qual operação o usuário quer realizar, após o servidor identificar, o usuário deverá informar um caminho válido, após a entrada do dado será enviada uma nova mensagem ao servidor que irá retornar as informações referentes aos arquivos contidos em um diretório ou de um arquivo em si.

A resposta do servidor será convertida de volta através do pickle, uma vez que é usado no servidor o pickle para enviar a mensagem, e é chamada a função ***formatFileInformation()*** que formata e exibe a informação para o usuário.



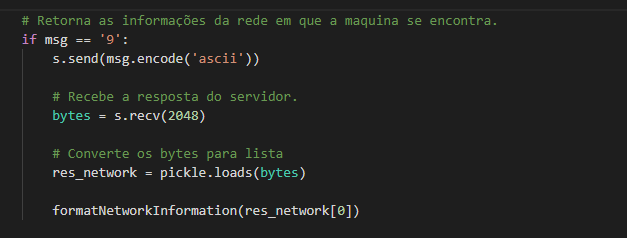
No caso sete, será enviado uma mensagem, convertida em bytes, ao servidor indicando qual operação o usuário quer realizar, após o servidor identificar, o usuário deverá informar o caminho de um diretório válido, após a entrada do dado será enviada uma nova mensagem ao servidor que irá retornar as informações referentes ao diretório informado.

A resposta do servidor será convertida de volta através do pickle, uma vez que é usado no servidor o pickle para enviar a mensagem, e é chamada a função ***formatDirInformation()*** que formata e exibe a informação para o usuário.



No caso oito, será enviado uma mensagem, convertida em bytes, ao servidor indicando qual operação o usuário quer realizar, após o servidor identificar, o usuário deverá informar o nome de de um processo ativo, após a entrada, será enviada uma nova mensagem ao servidor que irá retornar as informações referentes ao processo escolhido.

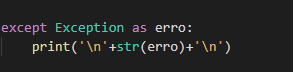
A resposta do servidor será convertida de volta através do pickle, uma vez que é usado no servidor o pickle para enviar a mensagem, e é chamada a função ***formatProcessInformation()*** que formata e exibe a informação para o usuário.



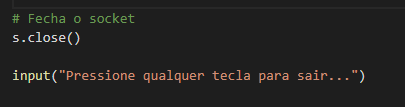
No caso nove, será enviado uma mensagem, convertida em bytes, ao servidor indicando qual operação o usuário quer realizar, operação essa que retorna as informações da rede em que a máquina se encontra..

A resposta do servidor será convertida de volta através do pickle, uma vez que é usado no servidor o pickle para enviar a mensagem, e será chamada a função ***formatNetworkInformation()*** que formata e exibe a informação para o usuário.

Com a última operação finaliza as interações do usuário com o sistema. caso aconteça algum erro durante as operações, ele irá cair na exceção e será exibido para o usuário.



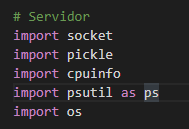
Ao looping ser encerrado o socket e finalizado e é informado ao usuário.

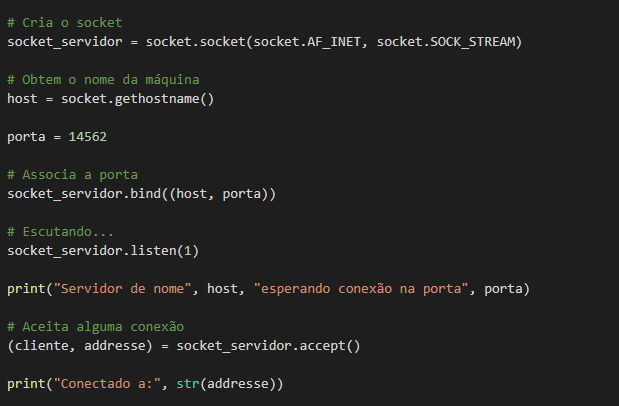


**Servidor**

Os imports das dependências necessárias para o projeto, estarei disponibilizando a documentação das mesmas

[socket](https://wiki.python.org.br/SocketBasico), [pickle](https://docs.python.org/3/library/pickle.html), [os](https://docs.python.org/2/library/os.html), [psutil](https://psutil.readthedocs.io/en/latest/)



***Da criação da conexão e espera de conexão:***

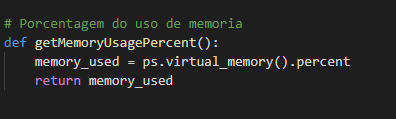
A variável ***socket\_servidor*** cria o socket passando as configurações de família e tipo, como explicado no cliente, family = socket.AF\_INET (ipv4) e type = socket.SOCK\_STREAM (TCP).

o nome da máquina é armazenado na variável ***host*** e a porta que será utilizada pelo servidor na variável ***porta,*** logo em seguida é feito o bind da configuração do socket com o método ***socket\_servidor.bind()*** passando o nome da máquina e a porta a ser utilizada.

com o método ***socket\_servidor.listen()*** inicia-se o processo de escuta do servidor, no qual ele fica escutando uma conexão com um cliente, o último passo é aceitar uma conexão, para isso é utilizado o método ***socket\_servidor.accept()***, é retornado dele as informações do cliente que estabeleceu a conexão, ***client*** que representa as informações do cliente e ***adderesse***  que representa o endereço do cliente, ao estabelecer a conexão é informado no terminal o endereço do cliente em questão.

***Das funções:***

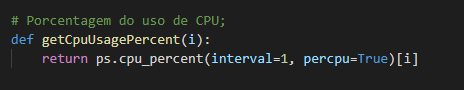
getMemoryUsagePercent()



Tem a finalidade de retornar o percentual de uso de memoria da maquina.

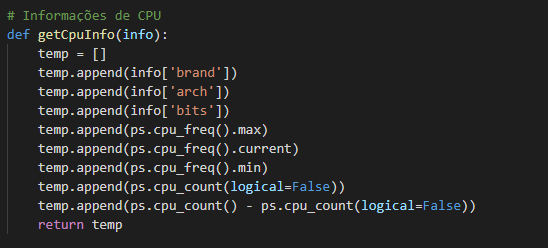
é utilizado o método do módulo psutil, ***virtual\_memory().percent*** para retornar o percentual de memória usada.

getCpuUsagePercent()



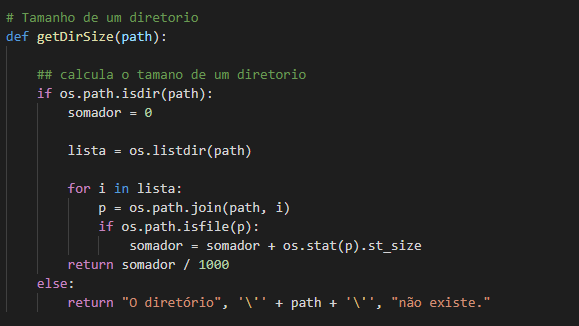
Tem a finalidade de retornar o percentual de uso da CPU da máquina, o metodo recebe um numeral ***i*** que é utilizado como contador para os nucleos da CPU, é utilizado o método do módulo psutil, ***cpu\_percent()***, passando as propiedades ***interval*** (intervalo de 1 segundo para cada busca) e ***percpu*** (retorna o percentual da informação), para retornar o percentual de cpu.

getCpuInfo()



Tem a finalidade de retornar uma lista com as principais informações da CPU da máquina, o método recebe um parâmetro ***info*** que representa o dicionário contendo informações da CPU e para capturar o resto das informaçõesé utilizado o método do módulo psutil, ***cpu\_freq()*** e ***cpu\_count()***, é retornada uma lista temporaria com toda a informação.

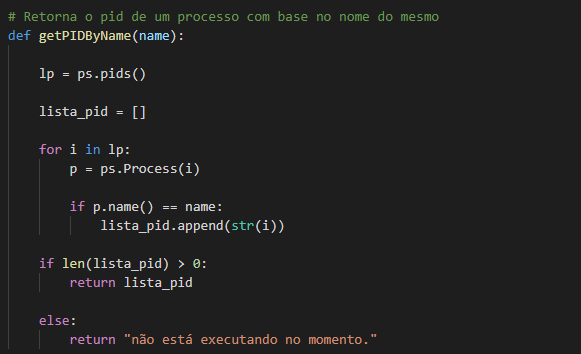
getDirSize()



Tem a finalidade de retornar o tamanho total de um diretório, o método recebe um parâmetro ***path*** que representa o caminho do diretório escolhido pelo usuário.

para uma melhor experiência do usuário, é feita uma verificação se o caminho é realmente um diretório caso contrário é retornado uma mensagem para o cliente, são criadas as variáveis ***somador***  e ***lista*** que representam um somador que receberá o tamanho de tudo que estiver dentro do diretório escolhido e a lista de arquivos e diretórios dentro do path escolhido, por fim é retornado o somador dividido por mill para melhor representatividade da medida KB.

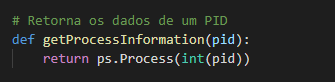
getPIDByName()



Tem a finalidade de retornar o PID de um processo com base no nome do mesmo, o método recebe um parâmetro ***name*** que representa nome do processo que o usuário escolheu.

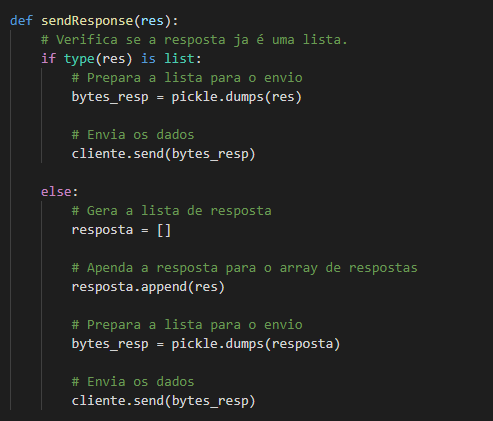
São criadas as variáveis ***lp*** e ***lista\_pid*** que representam todos os pid’s de processos sendo executados no momento e uma lista vazia que será preenchida com os pid’s existentes do processo escolhido pelo usuário. é feita uma verificação da lista no final caso não haja pid nenhum é retornado uma mensagem informando ao cliente que não há execução do processo escolhido.

getProcessInformation()



Tem a finalidade de retornar os dados de um processo com base no PID do mesmo, o método recebe um parâmetro ***pid*** que representa o PID do processo que o usuário escolheu.

sendResponse()

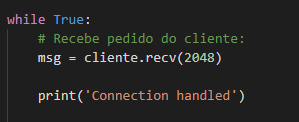


Tem a finalidade de enviar a resposta do servidor para o cliente, foi externalizada em uma função para melhor entendimento do código e menos repetição do mesmo.

Recebe como parâmetro ***res*** que representa a resposta a ser enviada, faz uma verificação do tipo do conteudo da variavel res, caso seja uma lista, sera convertida em bytes atravez do metodo ***pickle.dumbps()***  e enviada para o cliente com o método ***cliente.send()***. Caso não seja um array a resposta será inserida em uma lista, convertida para bytes e enviada para o cliente.

***Da estrutura de looping:***

Foi escolhido uma estrutura de looping feita com ***while*** para capturar as mensagens enviadas do cliente, no looping foi feita uma estrutura de decisões com ***if’s*** para gerenciar as opções e responder corretamente o cliente.



O looping tem como parâmetro o condicional ***True*** omantendo enquanto for verdadeiro,

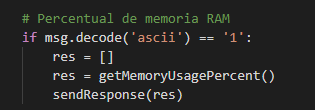
foi criada uma variável ***msg*** que recebe a mensagem do cliente com o método ***cliente.recv()*** passando o máximo de bytes que a mensagem deve ter, no caso 2048, e será informado no console quando a conexão for estabelecida.

***Das estruturas de decisão:***

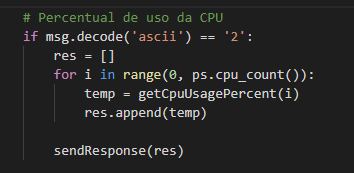
As mensagens recebidas do cliente serão decodificadas e respeitarão uma estrutura de decisão como já foi dito.



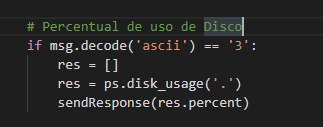
A primeira delas, decodifica a mensagem recebida e caso ela seja a ***string “fim”***  ela para o looping de repetição e encerra o lado do servidor.



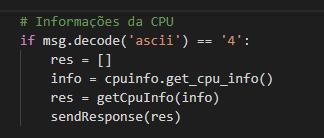
A segunda, decodifica a mensagem recebida e caso ela seja a ***string “1”,*** terá o objetivo de retornar o percentual de memória RAM usada, será criada uma variável**res** que receberá o retorno da função ***getMemoryUsagePercent()*** e será enviada como resposta para o cliente através do método ***sendResponse()***.



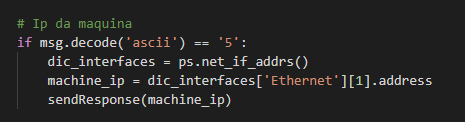
A terceira, decodifica a mensagem recebida e caso ela seja a ***string “2”,*** terá o objetivo de retornar o percentual de cpu usada por núcleo, será criada uma variável***res***, para contar os núcleos e ter a resposta dividia por eles, é usado uma iteração em cima da quantidade dos mesmos na máquina, será criada uma variável temporária que receberá o retorno da função ***getCpuUsagePercent()*** e será inserida na lista res criada anteriormente que ao sair da iteração será enviada como resposta para o cliente através do método ***sendResponse()***.



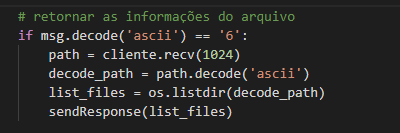
A quarta, decodifica a mensagem recebida e caso ela seja a ***string “3”,*** terá o objetivo de retornar o percentual de disco usado, será criada uma variável**res** que receberá o retorno da função ***ps.disk\_usage()*** passando ***“.”*** como parâmetroe será enviada como resposta para o cliente através do método ***sendResponse()***.



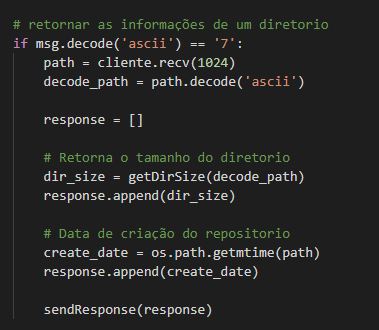
A quinta, decodifica a mensagem recebida e caso ela seja a ***string “4”,*** terá o objetivo de retornar as informações de cpu, será criada uma variável**res** que receberá o retorno da função ***getCpuInfo()*** passando a variável ***info***,que receberá as informações de cpu em um dicionario retornado do método ***cpuinfo.get\_cpu\_info()***,como parâmetroe será enviada como resposta para o cliente através do método ***sendResponse()***.



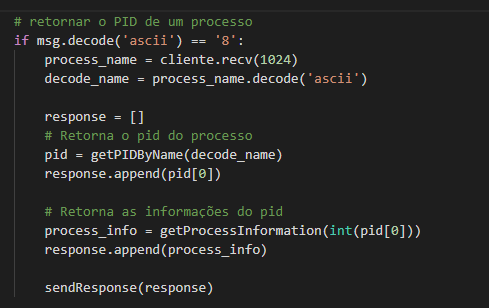
A sexta, decodifica a mensagem recebida e caso ela seja a ***string “5”,*** terá o objetivo de retornar o ip da máquina, será criada uma variável,***dic\_interfaces*** que receberá o retorno da função ***ps.net\_if\_addrs()*** em seguida é criada a variavel ***machine\_ip*** que recebe usa ***dic\_interfaces*** e extrai do dicionario o endereço IP da rede ***Ethernet*** que será enviada como resposta para o cliente através do método ***sendResponse()***.



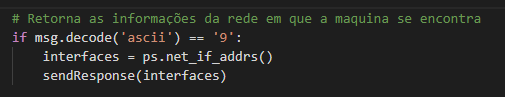
A sétima, decodifica a mensagem recebida e caso ela seja a ***string “6”,*** terá o objetivo de retornar as informações de um arquivo com base em um caminho inserido pelo usuário, após identificar a operação, o servidor ficará em aguardando a entrada do caminho pelo usuário, recebendo, será decodificado e armazenado na variável ***decode\_path***, em seguida será armazenado na variável ***list\_files*** a lista do conteúdo existente no caminho que o usuário inseriu retornada pelo método ***os.listdir()*** passando como parâmetro a variável ***decode\_path***, por fim será enviada como resposta para o cliente através do método ***sendResponse()***.



A oitava, decodifica a mensagem recebida e caso ela seja a ***string “7”,*** terá o objetivo de retornar as informações de um diretório com base em um caminho inserido pelo usuário, após identificar a operação, o servidor ficará em aguardando a entrada do caminho pelo usuário, recebendo, será decodificado e armazenado na variável ***decode\_path***, em seguida será criada três variáveis, ***response***  para armazenar a resposta a ser enviada, ***dir\_size*** que armazena o retorno do método ***getDirSize()*** passando ***decode\_path***  como parâmetro que terá o resultado inserido na lista ***response*** e ***create\_date*** que receberá o retorno do método ***os.path.getTime()*** passando o caminho inserido pelo usuário, por final será inserido na lista ***response*** que será enviada como resposta para o cliente através do método ***sendResponse()***.



A nona, decodifica a mensagem recebida e caso ela seja a ***string “8”,*** terá o objetivo de retornar as informações de um processo com base em um nome de processo inserido pelo usuário, após identificar a operação, o servidor ficará em aguardando a entrada do nome do processo pelo usuário, recebendo, será decodificado e armazenado na variável ***decode\_name***, em seguida será criada três variáveis, ***response***  para armazenar a resposta a ser enviada, ***pid*** que armazena o retorno do método ***getPIDByName()*** passando ***decode\_name***  como parâmetro que terá o resultado inserido na lista ***response*** e ***process\_info*** que receberá o retorno do método ***getProcessInformation()*** passando o PID inserido pelo usuário, por final será inserido na lista ***response*** que será enviada como resposta para o cliente através do método ***sendResponse()***.



A decima, decodifica a mensagem recebida e caso ela seja a ***string “9”,*** terá o objetivo de retornar as informações de redes da máquina em questão, será criada uma variável***interfaces*** que receberá o retorno da função ***os.net\_if\_addrs()*** e será enviada como resposta para o cliente através do método ***sendResponse()***.

Por final, Caso o usuário insira “***fim”***, com o break do looping, o socket do servidor será fechado com o método ***socket\_servidor.close()***

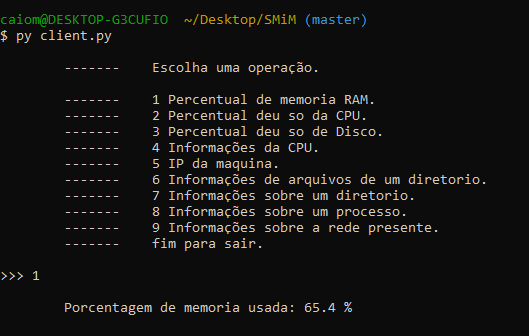


# Resultados e Análise

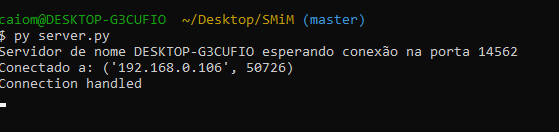
A seguir apresentarei por meio de imagens o funcionamento do sistema, passarei por todas as dez opções contidas no projeto.

***Opção um:***

***Cliente:***



***Servidor:***

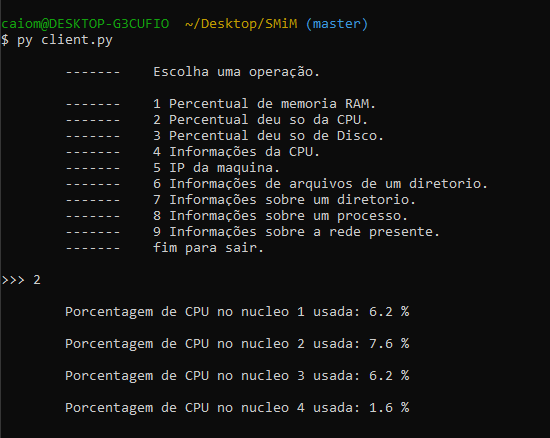


***Analise:***

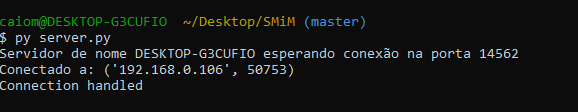
A opção um atinge as expectativas, tem uma ótima performance, e o código é bem simples de entender.

***Opção dois:***

***Cliente:***



***Servidor:***

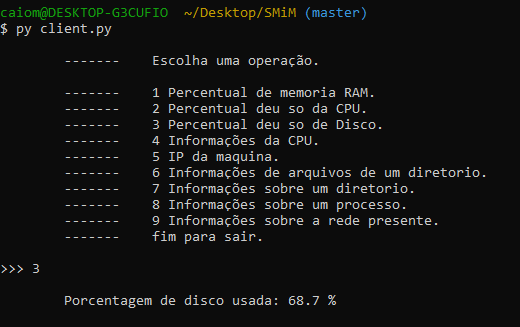


***Analise:***

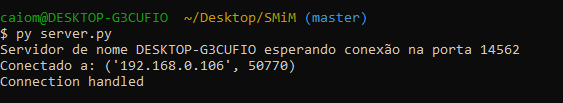
A opção dois atende as espetativas, consegue trazer os dados com precisão, porem acho que poderia ser feita de uma forma onde fosse feita uma busca por chamada e funcionasse encima de uma forma de schedule de alguns segundos ao invez de um time.sleep().

***Opção três:***

***Cliente:***



***Servidor:***

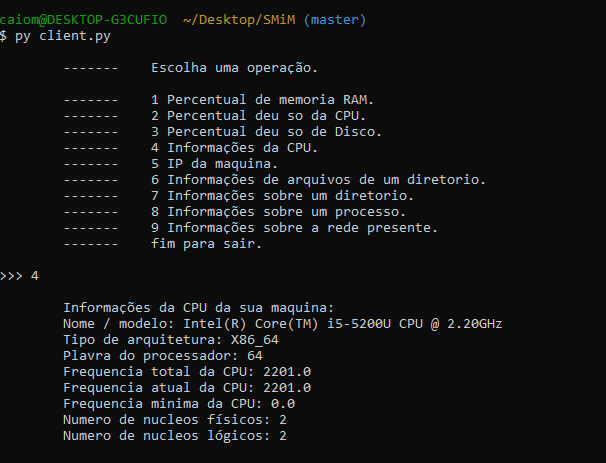


***Analise***

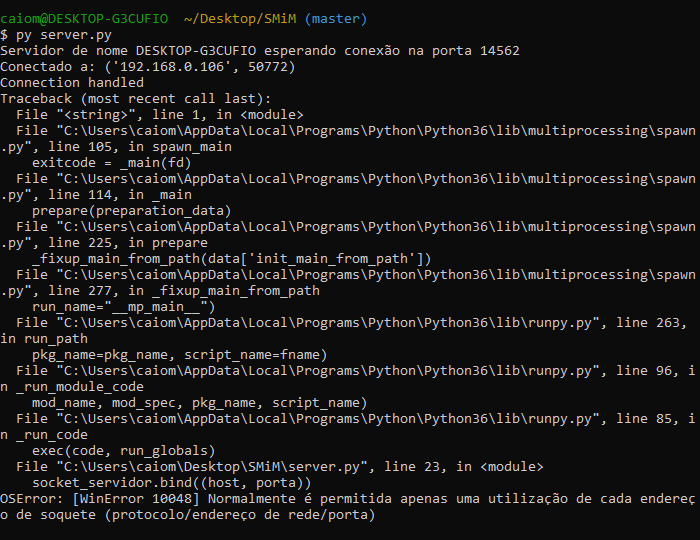
A opção um atinge as expectativas, tem uma ótima performance, e o código é bem simples de entender.

***Opção quatro***

***Cliente:***



***Servidor:***

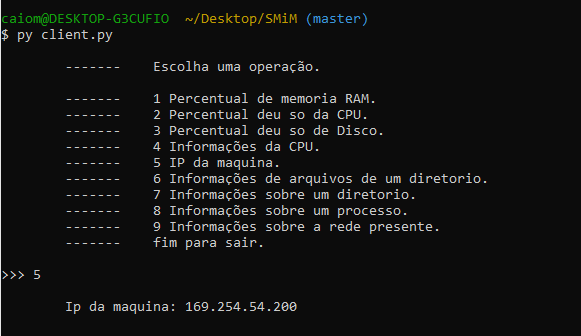


***Analise:***

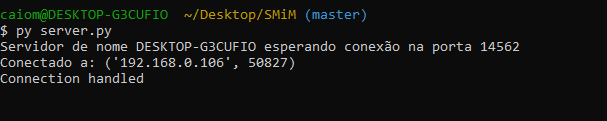
A opção um atinge as expectativas, tem uma ótima performance, e o código é bem simples de entender, porém como visto na imagem do servidor, há um erro que eu nao consegui resolver, ele não interfere no processamento do dado ou para a aplicação de algum jeito, uma [issue](https://github.com/caiomdias/SMiM/issues/5) foi aberta no github com o erro e será resolvida da melhor forma possível.

***Opção cinco:***

***Cliente:***



***Servidor:***



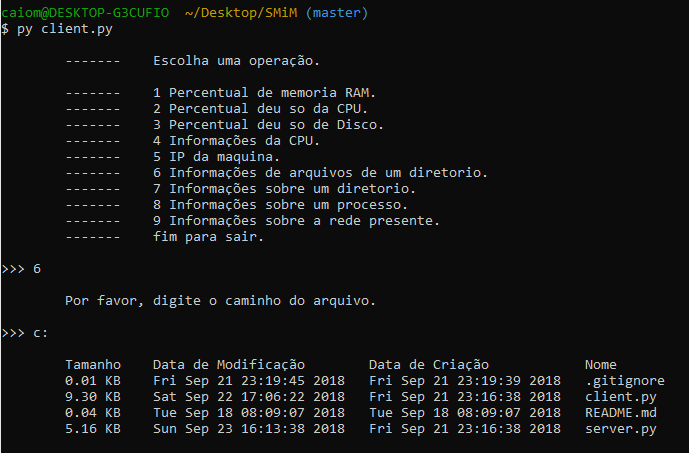
***Analise:***

A opção um atinge as expectativas, tem uma ótima performance, e o código é bem simples de entender.

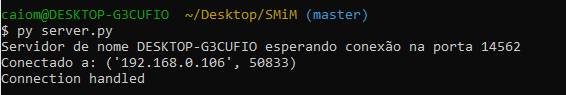
Um ponto de melhoria, seria retornar todos os endereços das redes em que a máquina se encontra.

***Opção seis:***

***Cliente:***



***Servidor:***



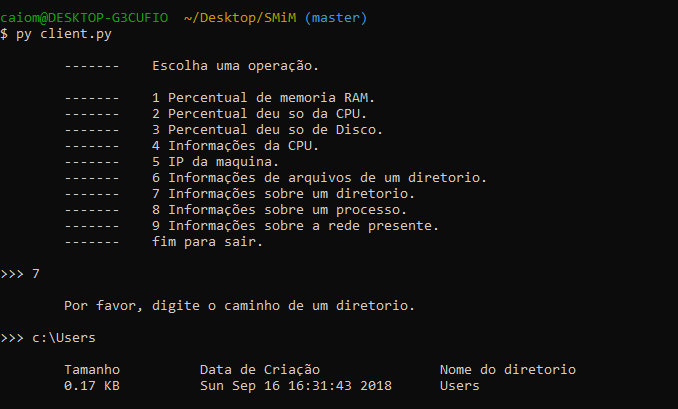
***Analise***

A opção um atinge as expectativas, tem uma ótima performance, e o código é bem simples de entender.

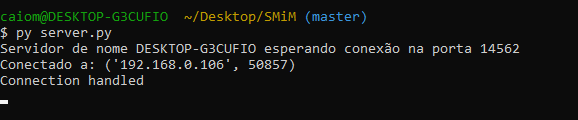
Um ponto de melhoria, seria uma melhor captação da mensagem do usuário junto com um melhor tratamento de erro, talvez um dicionário com os possíveis erros mapeados e identificados por alguma função na hora de lançar a ***exception***.

***Opção sete:***

***Cliente:***



***Servidor:***



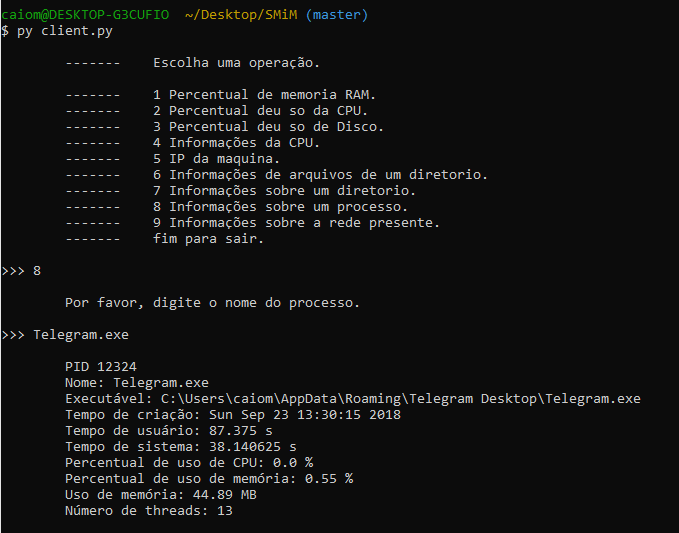
***Analise:***

A opção um atinge as expectativas, tem uma ótima performance, e o código é bem simples de entender.

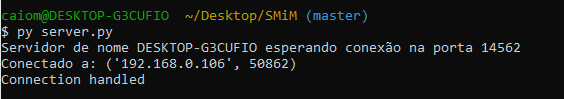
Um ponto de melhoria seria trazer mais informações do diretório em questão porém a disposição das informações para o usuário deve ser pensada melhor para proporcionar uma melhor experiência.

***Opção oito:***

***Cliente:***



***Servidor:***



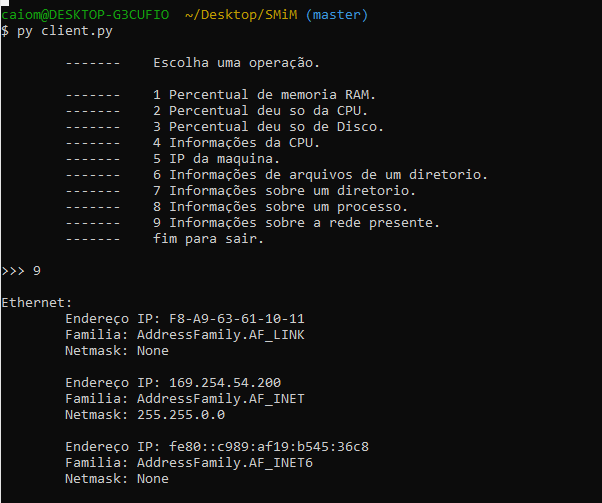
***Analise:***

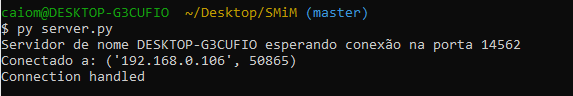
A opção um atinge as expectativas, tem uma ótima performance, e o código é bem simples de entender.

Um ponto de melhoria seria dar a opção de mostrar os processos existentes para o usuario, pos nao necessariamente o usuario tem conhecimento dos nomes do processos e um outro ponto seria um melhor tratamento de erro como falado anteriormente um dicionário prevendo os possíveis erros e ao acontecer identificaria nesse dicionário o erro com base em algum identificador gerado pelo erro.

***Opção nove:***

***Cliente:***



***Servidor:***

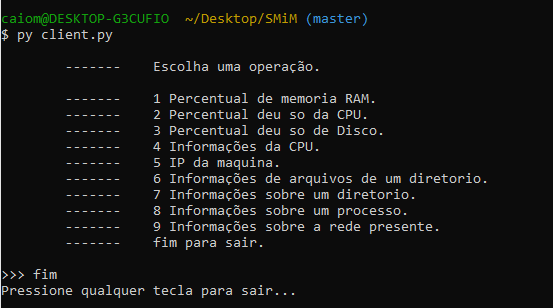
***Analise:***

A opção um atinge as expectativas, tem uma ótima performance, e o código é bem simples de entender.

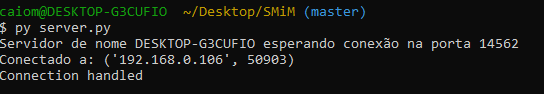
Um ponto de melhoria seria uma melhor disposição das informações para o usuário, em minha opinião não ficou disposto de uma forma agradável.

***Opção dez:***

***Cliente:***



***Servidor:***



***Analise:***

A opção um atinge as expectativas, tem uma ótima performance, e o código é bem simples de entender.

# Conclusão

A experiência de construir um sistema como esse, para mim foi algo no mínimo interessante, podemos ver a linguagem escolhida sendo usada para capturar informações de uma máquina de forma fácil e simples, as bibliotecas que auxiliam sao bem faceis de manusear e simples de entender.

A dificuldade da construção do sistema foi aprender a lidar com as informações de máquina necessárias, porém com todo o andamento dos estudos durante o período, se tornou um desafio agradável.

Com tudo o que foi visto concluo que pude aprender a gerenciar um projeto de pequena escala, pensar um pouco mais de forma analita aos problemas que a mim foram apresentados e absorvi certos conhecimentos do assunto proposto. O projeto tem algumas deficiências como tratamento de erro e disposição das informações para o usuário, porém imagino que sejam fáceis de solucionar, acredito que vale sim apena investir mais tempo e estudo no projeto para melhorá-lo, o projeto atendeu sim as minhas expectativas e acredito que deva dar continuidade.