

**PROJETO SISTEMAS OPERACIONAIS  
(OPÇÃO 2 – EXPERIMENTO REAL)  
FORMULÁRIO INDIVIDUAL DA ATIVIDADE PRÁTICA**

IDENTIFICAÇÃO	
<b>NOME DO(A) ESTUDANTE 1:</b> Maria Clara Pereira da Costa	
<b>MATRÍCULA:</b> 20211380040	<b>PERÍODO:</b> 2º

IDENTIFICAÇÃO	
<b>NOME DO(A) ESTUDANTE 2:</b> Italo Fernando Sousa Burity	
<b>MATRÍCULA:</b> 20211380015	<b>PERÍODO:</b> 2º

DESCRIÇÃO DO CENÁRIO
<p>Resumo do projeto:</p> <p>Linguagem utilizada:</p> <ul style="list-style-type: none"><li>- Python</li></ul> <p>Bibliotecas utilizadas:</p> <ul style="list-style-type: none"><li>- Os</li><li>- Time</li></ul> <p>O objetivo do projeto, é monitorar os recursos utilizados pelo Desktop em ambientes com o python instalado manualmente (Windows), e outro, com o python nativo do sistema operacional (Ubuntu). Os recursos de hardware disponibilizados serão os mesmos. Durante a execução de um programa em Python (SoBET), que define uma grande quantidade de jogadores e cartelas, onde é encerrado somente quando há um vencedor ou vencedores.</p> <p>Outras observações sobre o "Sorteio":</p> <ul style="list-style-type: none"><li>- Os números sorteados pela cartela não são repetidos</li><li>- No código serão efetuados inicialmente dois testes, na primeira execução, será executado com 10 mil jogadores, e depois, com 100 mil.</li><li>- Será comparado entre Windows e Linux-Ubuntu.</li></ul>

### **Questão de Pesquisa:**

- Qual o impacto no tempo de execução de um código, quando executado em dois sistemas operacionais diferentes, sendo um com python nativo (Linux - Ubuntu), e outro com python instalado manualmente (Windows).

#### **1) DESCRIÇÃO DO AMBIENTE DE AVALIAÇÃO**

##### **Informações da memória:**

###### **Memória Windows**

MemTotal: 5.5GB  
MemFree: 3.6GB  
MemAvailable: 5.5GB

###### **Memória Ubuntu**

MemTotal: 6078664 kB  
MemFree: 4115580 kB  
MemAvailable: 4859196 kB

##### **Informações do processador**

###### **Processador Windows**

Thread per core: 1  
Model name: AMD Ryzen 7 3700U  
CPU MHz: 2300

###### **Processador Ubuntu**

Thread per core: 1  
Model name: AMD Ryzen 7 3700U  
CPU MHz: 2299.998

## 2) CÓDIGO DO PROJETO

Link para visualizar execução:

[https://colab.research.google.com/drive/12NqVyupH\\_MevWsRK8nUSpGt45EYAJGyT?usp=sharing](https://colab.research.google.com/drive/12NqVyupH_MevWsRK8nUSpGt45EYAJGyT?usp=sharing)

```
import os
from random import randint
import time

def contido(n, lista):
    if (n in lista):
        return True
    else:
        return False

def novo(n1, n2):
    numeros = []

    while (len(numeros) < 10):
        num = randint(n1, n2)
        if (contido(num, numeros) == False):
            numeros.append(num)
    return numeros

def numero_sorteado(n1, n2, nsort):
    while True:
        n = randint(n1, n2)
        if not contido(n, nsort):
            nsort.append(n)
        return n

def limpar():
    if os.name == 'nt':
        os.system('cls')
    else:
        os.system('clear')

start = time.perf_counter()
```

```

time.sleep(5)
num = (10000) #num (100000) para o segundo teste
cartelas = []
sorteado = []
ganhadores = []
vencedor = False
count = 0

qtde = 0
for i in range(num):
    lista = novo(1, 50)
    cartelas.append(lista)

while not vencedor:
    sort = numero_sorteado(1, 50, sorteado)
    print("SoBET")
    print(f"Número sorteado: {sort}")
    for i in range(len(cartelas)):
        qtde = 0
        for j in range(len(cartelas[i])):
            if (contido(cartelas[i][j], sorteado) == True):
                qtde += 1
        print(f"Jogador {i + 1}", (cartelas[i]), qtde)
        if qtde == 10:
            vencedor = True
            ganhadores.append(i)
    if not vencedor:
        print("Ainda não temos vencedores.")
        count = count + 1
        print('A quantidade de números sorteados até o momento é:
',count)
        time.sleep(1)
        limpar()

print()
for s in ganhadores:
    print(f"Jogador {s + 1} completou a cartela! ", cartelas[s])

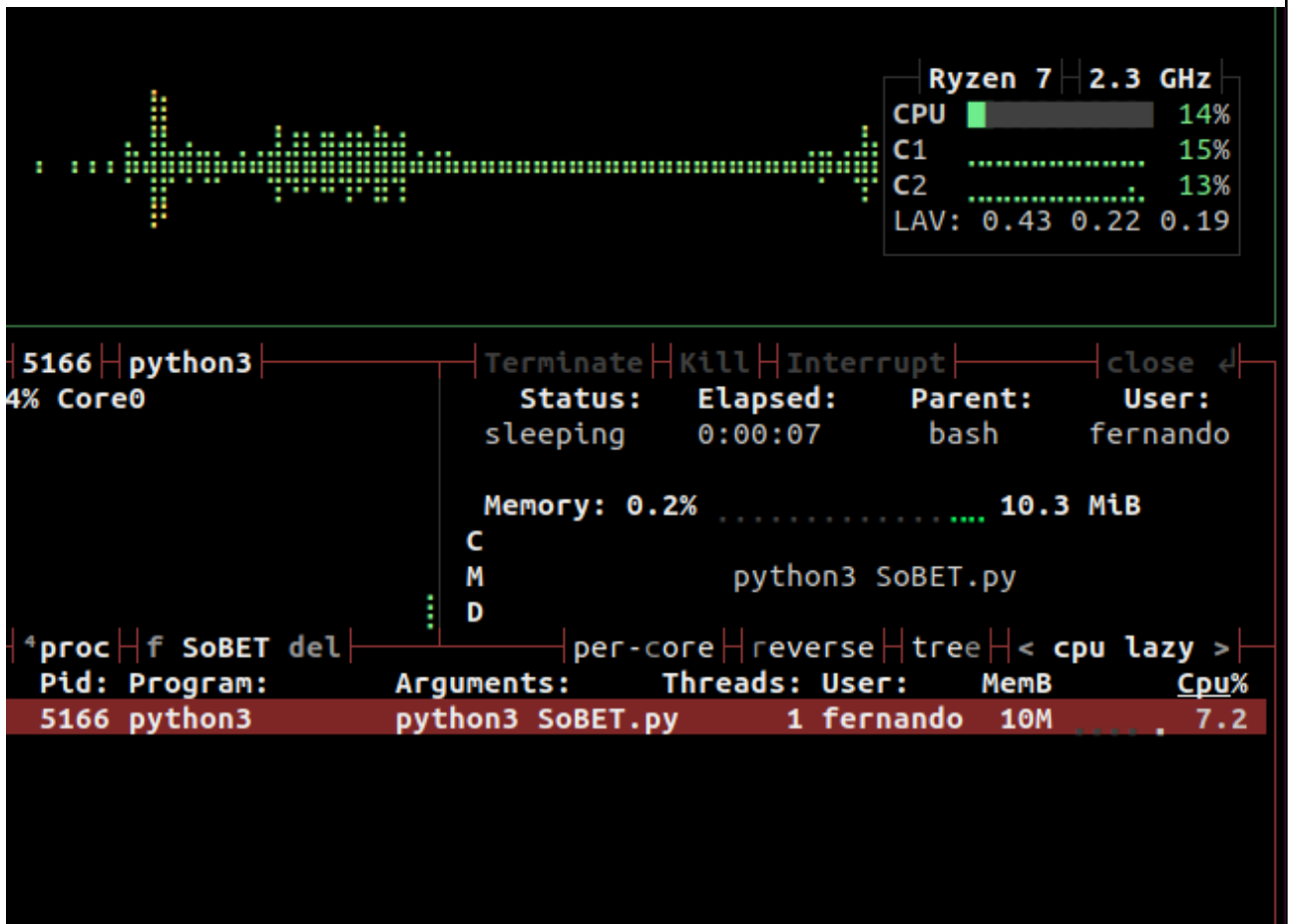
print("Números sorteados: ", sorteado)

end = time.perf_counter()
print(end - start, "Segundos")

```

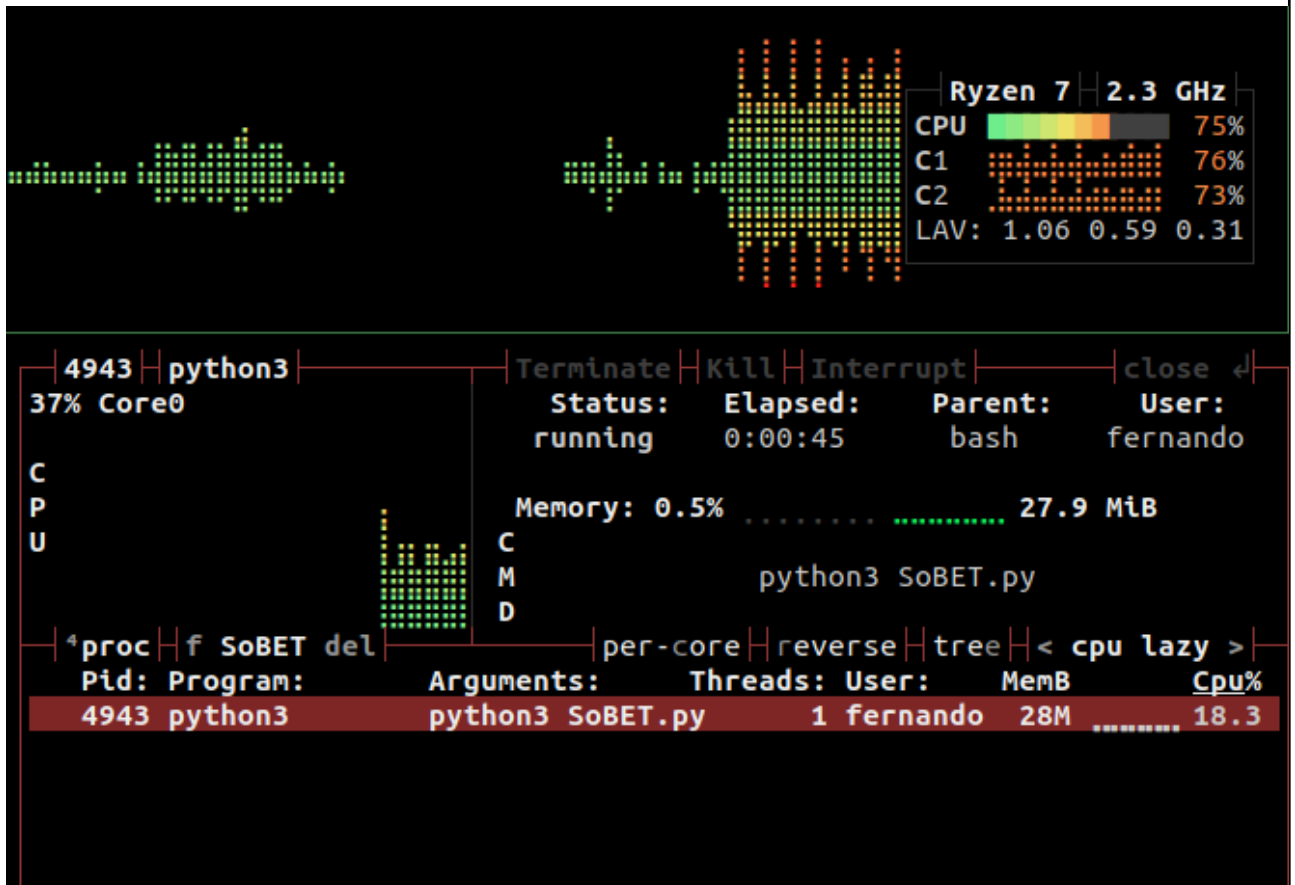
### 3) RESPOSTA

LINUX - UBUNTU - Execução com 10.000 jogadores:



- Ao executar o programa com 10.000 jogadores, utilizando a ferramenta “bpytop” para monitorar e foi verificado o consumo em torno de 7.2% da CPU inicialmente, utilizando apenas 0.2% da RAM.

## LINUX - UBUNTU com pico de jogadores em 100.000:



- Na execução do programa contendo 100.000 jogadores, já é possível observar o consumo maior do recurso da CPU. Chegando a consumir 18.3%, esse valor foi variando para baixo em alguns momentos, mas sempre próximo desse uso. O consumo de RAM subiu em relação anterior apenas para 0.5%

## WINDOWS - Execução com 10.000 jogadores:

Gerenciador de Tarefas

Arquivo Opções Exibir

Processos Desempenho Histórico de aplicativos Inicializar Usuários Detalhes Serviços

Nome	Status	84% CPU	29% Memória	0% Disco	0% Rede	Uso de energia
<b>Aplicativos (4)</b>						
Windows PowerShell (4)		70,9%	48,9 MB	0 MB/s	0 Mbps	Alta
Windows PowerShell		0%	36,2 MB	0 MB/s	0 Mbps	Muito baixo
Python (32 bits)		0%	0,8 MB	0 MB/s	0 Mbps	Muito baixo
Python		11,7%	8,2 MB	0 MB/s	0 Mbps	Baixa
Host da Janela do Console		59,1%	3,1 MB	0 MB/s	0 Mbps	Moderada
Windows Explorer		0%	28,9 MB	0 MB/s	0 Mbps	Muito baixo
Gerenciador de Tarefas		2,1%	18,8 MB	0 MB/s	0 Mbps	Muito baixo
Bloco de notas		0%	2,0 MB	0 MB/s	0 Mbps	Muito baixo

- Quando o programa iniciou, foi possível observar um pico no uso da CPU pelo processo.

Gerenciador de Tarefas

Arquivo Opções Exibir

Processos Desempenho Histórico de aplicativos Inicializar Usuários Detalhes Serviços

Nome	Status	39% CPU	29% Memória	0% Disco	0% Rede	Uso de energia	Tendência de ...
<b>Aplicativos (4)</b>							
Windows PowerShell (4)		36,5%	49,0 MB	0 MB/s	0 Mbps	Moderada	Moderada
Windows PowerShell		0%	36,2 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
Python (32 bits)		0%	0,8 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
Python		20,0%	8,2 MB	0 MB/s	0 Mbps	Baixa	Baixa
Host da Janela do Console		16,5%	3,9 MB	0 MB/s	0 Mbps	Baixa	Baixa
Windows Explorer		0%	28,8 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
Gerenciador de Tarefas		1,7%	18,8 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo
Bloco de notas		0%	2,0 MB	0 MB/s	0 Mbps	Muito baixo	Muito baixo

- 36,5% de USO da CPU pelo PowerShell enquanto o programa com 10 mil jogadores estava sendo executado após o pico inicial. Esse valor sofreu diversas variações, mas sempre em uma média próxima ao valor apresentado na imagem. O consumo de memória RAM foi de 49,0MB, superior ao consumo pelo LINUX.

**WINDOWS - Execução com pico de jogadores em 100.000:**

Windows PowerShell

Jogador 99955 [7, 15, 14, 14]  
Jogador 99956 [42, 19, 50, 14]  
Jogador 99957 [4, 29, 14, 14]  
Jogador 99958 [32, 20, 10, 14]  
Jogador 99959 [46, 48, 30, 14]  
Jogador 99960 [33, 27, 13, 14]  
Jogador 99961 [6, 1, 3, 2]  
Jogador 99962 [3, 45, 33, 14]  
Jogador 99963 [47, 39, 38, 14]  
Jogador 99964 [49, 10, 28, 14]  
Jogador 99965 [46, 1, 21, 14]  
Jogador 99966 [41, 25, 8, 14]  
Jogador 99967 [6, 2, 7, 4]  
Jogador 99968 [8, 34, 2, 14]  
Jogador 99969 [41, 5, 35, 14]  
Jogador 99970 [30, 22, 19, 14]  
Jogador 99971 [14, 42, 13, 14]  
Jogador 99972 [29, 10, 4, 14]  
Jogador 99973 [37, 27, 1, 14]  
Jogador 99974 [3, 44, 37, 14]  
Jogador 99975 [29, 31, 4, 14]  
Jogador 99976 [26, 12, 2, 14]  
Jogador 99977 [20, 4, 3, 14]  
Jogador 99978 [9, 38, 48, 14]  
Jogador 99979 [36, 41, 39, 14]

Gerenciador de Tarefas

ArquivoOpçõesExibir

ProcessosDesempenhoHistórico de aplicativosInicializarUsuáriosDetalhesServiços

Nome

Status

100% CPU

40% Memória

0% Disco

0% Rede

Uso de energia

Aplicativos (4)

Windows PowerShell (4)

Windows PowerShell

Python (32 bits)

Python

Host da Janela do Console

Windows Explorer

Gerenciador de Tarefas

Bloco de notas

Processos em segundo plano (...)

Nome	Status	100% CPU	40% Memória	0% Disco	0% Rede	Uso de energia
Aplicativos (4)						
Windows PowerShell (4)		68,7%	57,6 MB	0 MB/s	0 Mbps	Alta
Windows PowerShell		0%	29,3 MB	0 MB/s	0 Mbps	Muito baixo
Python (32 bits)		0%	0,8 MB	0 MB/s	0 Mbps	Muito baixo
Python		33,4%	23,9 MB	0 MB/s	0 Mbps	Moderada
Host da Janela do Console		35,3%	3,6 MB	0 MB/s	0 Mbps	Moderada
Windows Explorer		0%	31,4 MB	0 MB/s	0 Mbps	Muito baixo
Gerenciador de Tarefas		5,1%	20,8 MB	0 MB/s	0 Mbps	Muito baixo
Bloco de notas		0%	2,0 MB	0 MB/s	0 Mbps	Muito baixo
Processos em segundo plano (...)						

- Ao executar o programa com 100.000 jogadores. O consumo da CPU aumentou consideravelmente. Antes saindo de um valor médio que variou na faixa dos 30%, foi possível verificar uma média girando em torno dos 60% de consumo da CPU.

Gerenciador de Tarefas

ArquivoOpçõesExibir

ProcessosDesempenhoHistórico de aplicativosInicializarUsuáriosDetalhesServiços

Nome	Status	100% CPU	28% Memória	0% Disco	0% Rede	Uso de energia
Aplicativos (4)						
Windows PowerShell (4)		98,0%	67,1 MB	0 MB/s	0 Mbps	Muito alto
Windows PowerShell		0%	35,7 MB	0 MB/s	0 Mbps	Muito baixo
Python (32 bits)		0%	0,8 MB	0 MB/s	0 Mbps	Muito baixo
Python		63,0%	26,8 MB	0 MB/s	0 Mbps	Alta
Host da Janela do Console		35,0%	3,9 MB	0 MB/s	0 Mbps	Moderada
Windows Explorer		0%	28,9 MB	0 MB/s	0 Mbps	Muito baixo
Gerenciador de Tarefas		1,0%	18,8 MB	0 MB/s	0 Mbps	Muito baixo
Bloco de notas		0%	1,9 MB	0 MB/s	0 Mbps	Muito baixo

Windows

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

Jogador 86

J

- Em alguns momentos foi possível visualizar o pico de consumo de 98% da CPU, mas esse valor não permaneceu por muito tempo, retornando aos valores médios que foram apresentados.



Comparação do tempo de execução com 10.000 jogadores.

```
5166 | python3 | Terminate | Kill | Interrupt | close |
0% Core1 | Status: | Elapsed: | Parent: | User: |
          | dead | 0:00:28 | bash | fernando |
          | Memory: 0.2% | ..... | 10.3 MiB |
          | C |
          | M | python3 SoBET.py
          | D |
4 | proc | f SoBET del | per-core | reverse | tree | < cpu lazy > |
  | Pid: Program: | Arguments: | Threads: User: | MemB | Cpu% |
```

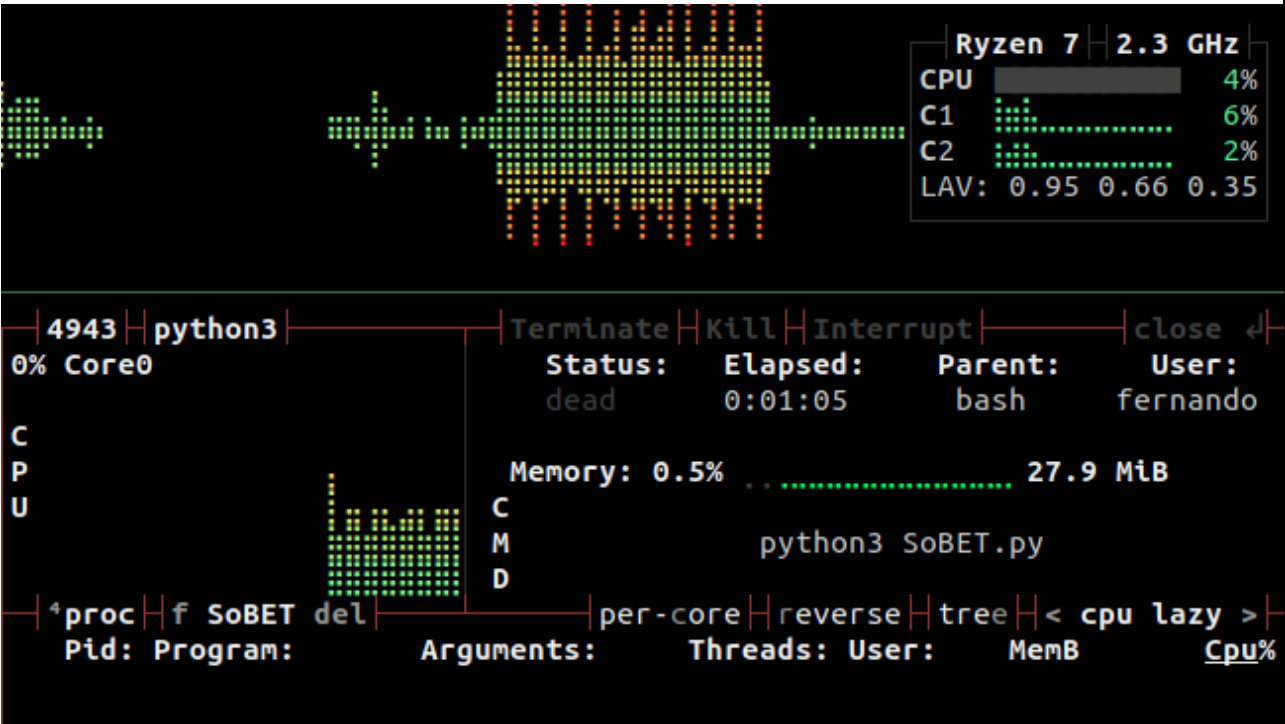
- No Ubuntu o programa com 10.000 jogadores foi executado e finalizado em torno de 28 segundos.

```
Jogador 9996 [21, 18, 7, 6, 28, 11, 38, 48, 15, 16] 3
Jogador 9997 [27, 10, 40, 44, 13, 22, 36, 45, 9, 43] 3
Jogador 9998 [7, 44, 23, 49, 31, 47, 6, 14, 21, 42] 5
Jogador 9999 [50, 12, 22, 44, 26, 11, 1, 17, 32, 15] 5
Jogador 10000 [15, 40, 6, 13, 19, 14, 45, 20, 7, 48] 5

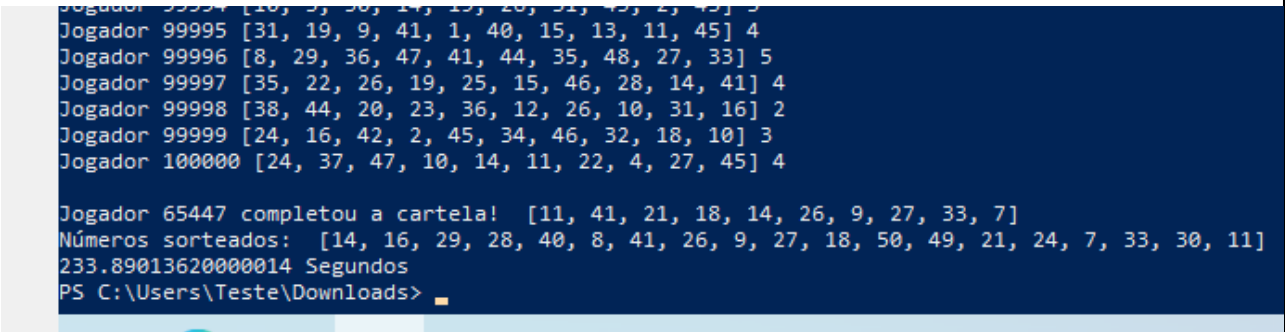
Jogador 2078 completou a cartela! [31, 44, 14, 4, 50, 43, 1, 5, 34, 20]
Números sorteados: [27, 14, 44, 43, 6, 29, 32, 17, 19, 35, 31, 4, 7, 1, 34, 5, 38, 50, 20]
92.97512579999966 Segundos
PS C:\Users\Teste\Downloads>
```

- No Windows o programa com 10.000 jogadores foi executado e finalizado em torno de 92 segundos. (OBS. Foi usando um comando em Python para medir o tempo utilizado ao executar o código no Windows, o comando em questão foi o “time.perf\_counter()” da biblioteca time.)

Comparação do tempo de execução com 100.000 jogadores.



- No Ubuntu o programa com 100.000 jogadores foi executado e finalizado em torno de 1 minuto e 05 segundos.



- No Windows o programa com 100.000 jogadores foi executado e finalizado em 233 segundos, quase 4 minutos.

#### 4) RESPONDER

A pesquisa se baseou em testar o comportamento de um mesmo programa python em dois ambientes diferentes. Foi tentado ao máximo igualar a quantidade de recursos direcionados para os ambientes em questão, para que a comparação fosse a mais próxima possível.

Como a proposta era visualizar o comportamento do processo em um ambiente que não possui o python nativo em seu sistema, o Windows foi a escolha e junto com ele, foi necessário instalar o python3 para que o teste pudesse ocorrer. Na tentativa de ter uma execução mais semelhante ao Linux, foi utilizado o powershell para que o código fosse executado e acompanhado com mais precisão.

Como mostrado nos resultados, o programa, em ambas as condições 10.000 e 100.000 jogadores, executado em um ambiente Linux, teve o uso de recursos bem inferior ao ambiente Windows. Foi utilizado o comando `time.sleep()` para que o intervalo entre o sorteio do programa fosse um pouco maior, para que a observação do comportamento do processo fosse mais detalhada. Com isso foi possível ver que naturalmente um Script em Python, executado em escalas menores ou maiores, são executados e utilizam menos recursos em ambientes como o Linux-Ubuntu que possuem o python nativo em seu sistema. Acrescentando um adendo ao tempo de execução, que se mostrou bem menor no ambiente Linux, comparando a execução dos dois processos em 100.000 jogadores, vemos a diferença de 168 segundos, com o Linux finalizando o programa em 65 segundos e o Windows em torno de 233.

Em suma. Os dois sistemas operacionais conseguiram executar o programa e manter sua execução até o final. Foram usadas ferramentas como o Powershell + Gerenciador de Tarefas + python3, para que a verificação no ambiente Windows fosse realizada, testes que apresentaram certo resultado ao identificar a utilização de recursos pela máquina. Claramente esse programa sendo executado em uma máquina que tivesse mais recursos disponíveis, como mais Núcleos ou mais Memória RAM, obteríamos resultados melhores, mas como a questão era a comparar com um ambiente Linux que possui as mesmas especificações, apenas utilizando o programa bpytop + python3 nativo do sistema, foi possível comprovar o comportamento mais “fluído”, onde no monitoramento o sistema apresentou uma execução com menos recursos e bem mais rápida pelo ambiente Linux-Ubuntu.