# 1 - Faça um tutorial sobre o Vs Code o Github.

Exercícios:

1) Olá, usuário!

Enunciado: Leia o nome do usuário e exiba uma saudação.

Requisitos: usar input() e print().

2) Soma de dois números

Enunciado: Leia dois números e mostre a soma.

Requisitos: converter para float ou int.

3) Média de 3 notas

Enunciado: Leia 3 notas e calcule a média.

Requisitos: usar variáveis e operadores.

4) Par ou ímpar

Enunciado: Leia um número inteiro e informe se é par ou ímpar.

Requisitos: usar operador % e if/else.

5) Tabuada

Enunciado: Leia um número e imprima a tabuada dele (1 a 10).

Requisitos: usar for.

Extra: permitir que o usuário escolha o intervalo (ex.: até 20).

6) Contador de vogais

Enunciado: Leia uma frase e conte quantas vogais existem.

Requisitos: usar for e checagem em conjunto/string (in).

Extra: contar separadamente a, e, i, o, u.

7) Lista de números: maior, menor e média

Enunciado: Peça ao usuário uma quantidade N e leia N números, guardando em uma lista. Depois mostre: maior, menor e média.

Requisitos: usar lista e sum, max, min.

Extra: validar N > 0.

8) Jogo de adivinhação

Enunciado: O programa escolhe um número aleatório de 1 a 100 e o usuário tenta adivinhar.

Requisitos: usar random.randint, while, contador de tentativas.

Extra: dar dicas: “maior” ou “menor”.

9) Funções: calculadora simples

Enunciado: Crie funções somar, subtrair, multiplicar, dividir e um menu para o usuário escolher a operação.

Requisitos: usar def, return, if/elif.

Extra: tratar divisão por zero.

# 10) Arquivo (TXT): salvar e ler tarefas

**Enunciado:** Faça um programa que permita:

1. adicionar uma tarefa (salvar em `tarefas.txt`)
2. listar tarefas (ler do arquivo e mostrar)
3. sair
   **Requisitos:** usar `open(..., "a")`, `open(..., "r")`, menu em `while`.
   **Extra:** numerar as tarefas ao exibir.

Exercícios sobre gerenciamento de Sistema Operacional

# ) Monitor simples de memória RAM

**Enunciado:**
Crie um programa em Python que mostre:

- quantidade total de memória RAM,
- memória sendo usada,
- memória livre (% e em MB).

**Requisitos:**

- Atualizar as informações a cada 2 segundos.
- Exibir de forma organizada no terminal.

**Dica:** usar `psutil.virtual_memory()` e `time.sleep()`.

---

# 2) Monitor de uso de memória RAM com alerta

**Enunciado:**
A partir do exercício anterior, crie um programa que **exiba um alerta** caso o uso de RAM ultrapasse um valor definido pelo usuário (por exemplo, 80%).

**Requisitos:**

- Ler do usuário o limite (em %).

- Exibir mensagem de alerta se o uso passar do limite.
- Continuar rodando em loop até o usuário interromper (Ctrl+C).

---

# 3) Gerenciador de espaço em disco (todas as partições)

**Enunciado:**
Crie um programa que liste **todas as partições de disco** do computador e mostre para cada uma:

- ponto de montagem (ex.: `C:\`, `/`, `/home`),
- espaço total,
- espaço usado,
- espaço livre,
- porcentagem usada.

**Requisitos:**

- Exibir os dados em forma de tabela no terminal.

**Dica:** usar `psutil.disk_partitions()` e `psutil.disk_usage()`.

---

# 4) Alerta de pouco espaço em disco

**Enunciado:**
Crie um programa que, com base no exercício 3, exiba um **alerta** para cada partição cujo espaço livre seja menor que, por exemplo, **10%**.

**Requisitos:**

- Mostrar apenas as partições "críticas".
- Limite (%) pode ser constante no código ou lido do usuário.

---

# 5) Monitor de tráfego de rede (download e upload)

**Enunciado:**
Crie um programa que mostre o tráfego de rede em tempo real (aproximado), exibindo a quantidade de bytes enviados e recebidos por segundo.

**Requisitos:**

- Medir `bytes_recv` e `bytes_sent` em um intervalo (por exemplo, 1 segundo).
- Calcular a diferença entre duas leituras para estimar "por segundo".

- Exibir no terminal algo como:
  `Download: X kB/s | Upload: Y kB/s`

**Dica:** usar `psutil.net_io_counters()` e `time.sleep()`.

---

# 6) Monitor de desempenho da CPU

**Enunciado:**
Crie um programa que mostre o uso da CPU em tempo real:

- uso total da CPU (%),
- uso por núcleo (%).

**Requisitos:**

- Atualizar, por exemplo, a cada 1 segundo.
- Exibir no terminal algo como:
  ```
  CPU Total: 35%
  Núcleo 0: 30% | Núcleo 1: 40% | ...
  ```

**Dica:** usar `psutil.cpu_percent(interval=1, percpu=True)`.

---

# 7) Histórico de uso da CPU (simples “logger”)

**Enunciado:**
Crie um programa que registre o uso da CPU em um arquivo `cpu_log.txt` a cada 5 segundos.

**Requisitos:**

- Em cada linha do arquivo, salvar: data/hora e porcentagem de uso da CPU.
- Ex.: `2026-02-19 19:30:12 - CPU: 42%`
- O programa pode rodar em loop até ser interrompido.

**Dica:** usar `datetime` e `psutil.cpu_percent()`.

---

# 8) Informações detalhadas do processador (incluindo número de série, se possível)

**Enunciado:**
Crie um programa que exiba informações sobre o processador do computador, como:

- nome/modelo da CPU,
- número de núcleos físicos e lógicos,
- frequência (em MHz/GHz),
- **tentar** obter o número de série da CPU (quando o sistema operacional permitir).

**Requisitos:**

- Exibir todas as informações de forma organizada.
- Se não conseguir obter o número de série, exibir mensagem explicando que não é possível no sistema atual.

**Dicas:**

- `psutil.cpu_count()`, `psutil.cpu_freq()`.
- Para modelo/identificação, pode usar `platform` ou bibliotecas como `cpuinfo` (`pip install py-cpuinfo`), dependendo do nível que você quiser exigir dos alunos.
- Número de série costuma ser **difícil ou impossível** de obter de forma portável; isso é um ótimo gancho didático para discutir **limitações de software vs hardware e sistema operacional**.

---

# 9) Listagem de dispositivos de entrada e saída (conceitual + prático)

**Enunciado:**
Crie um programa que:

1. Liste os dispositivos de armazenamento (HDs/SSDs/pendrives) conectados como possíveis dispositivos de entrada/saída de dados.
2. Permita ao usuário escolher um dispositivo/partição e mostrar informações detalhadas daquele dispositivo/partição.

**Requisitos:**

- Listar pelo menos as partições (similar ao exercício 3).
- Discutir, em comentário no código, exemplos de outros dispositivos de E/S: teclado, mouse, monitor, impressora etc., mesmo que não sejam detectados via código.

**Dica:** reutilizar `psutil.disk_partitions()` e acrescentar algum texto explicativo nos comentários sobre dispositivos de E/S.

---

# 10) Painel integrado de monitoramento (RAM + CPU + Disco + Rede)

**Enunciado:**
Crie um programa em Python que funciona como um **painel de monitoramento** simples no terminal, mostrando na mesma tela:

- Uso de RAM (% e MB usados),
- Uso da CPU (total),
- Espaço livre na partição principal (por exemplo, onde o sistema está instalado),
- Taxa de download/upload aproximada (como no exercício 5).

**Requisitos:**

- Atualizar as informações a cada 2 segundos.
- Organizar o layout de forma legível (linhas separadas, talvez limpar a tela a cada atualização com `os.system("cls")` no Windows ou `os.system("clear")` no Linux).
- Rodar em loop até o usuário interromper.

**Dicas:**

- Combinar tudo que foi feito nos exercícios anteriores.
- Usar `psutil` para RAM, CPU, disco e rede.

Python - Escreva seus primeiros programas - Casa do Codigo.pdf