

Organização de dados e criação de conjuntos de dados – Coleta dados de máquina (PI)

Antes fazer qualquer análise de dados, vamos coletar dados de interesse e que estejam integrados a aplicação do projeto de PI.

O PI tem por objetivo o monitoramento de no mínimo CPU, memória e Hard disk.

Usaremos estes dados para criar nossos conjuntos e assim aplicar as técnicas da disciplina de cálculo.

Para capturar dados de máquina, o python usa a biblioteca psutil.

A **psutil** (sistema python e utilitários de processo) é uma biblioteca de **plataforma cruzada** para recuperar informações sobre processos em execução e utilização do sistema (CPU, memória, discos, rede, sensores) em Python. É útil principalmente para monitoramento de sistema, criação de perfil, limitação de recursos de processo e gerenciamento de processos em execução.

Mas o que é plataforma cruzada?

Aplicação Nativa: Criada especificamente para cada sistema (ex.: Android; IOs)

Aplicação em plataforma cruzada: Surgiu da utilização de plataformas e linguagens WEB para o desenvolvimento Mobile (para Android, IOs, Windows Phone). É considerada como uma única versão que roda para todos os sistemas.

Baixar a psutil-5.9.1-cp310-cp310-win_amd64.whl ou relativa aos bits de seu processador. <https://pypi.org/project/psutil/>

The screenshot shows the PyPI project page for psutil 5.9.1. The page has a blue header with the project name and version, a search bar, and navigation links. Below the header, there's a section for the project description with various statistics like downloads, stars, forks, contributors, and coverage. A sidebar on the left contains navigation links like 'Project description', 'Release history', 'Download files', 'Project links', and 'Statistics'. The 'Quick links' section on the right includes links to the home page, install, documentation, download (highlighted with a red box), forum, and StackOverflow.

Vá em downloads e procure a api relacionada a sua máquina:

Cross-platform lib for process and system monitoring in Python.

Navigation

- Project description
- Release history
- Download files**

Project links

- Homepage

Statistics

GitHub statistics:

Download files

Download the file for your platform. If you're not sure which to choose, learn more about [installing packages](#).


Source Distribution

- [psutil-5.9.1.tar.gz](#) (479.1 kB [view hashes](#))
Uploaded May 20, 2022 [source](#)

Built Distributions

- [psutil-5.9.1-cp310-cp310-win_amd64.whl](#) (245.8 kB [view hashes](#))
Uploaded May 20, 2022 [cp310](#)
- [psutil-5.9.1-cp310-cp310-win32.whl](#) (241.7 kB [view hashes](#))
Uploaded May 20, 2022 [cp310](#)

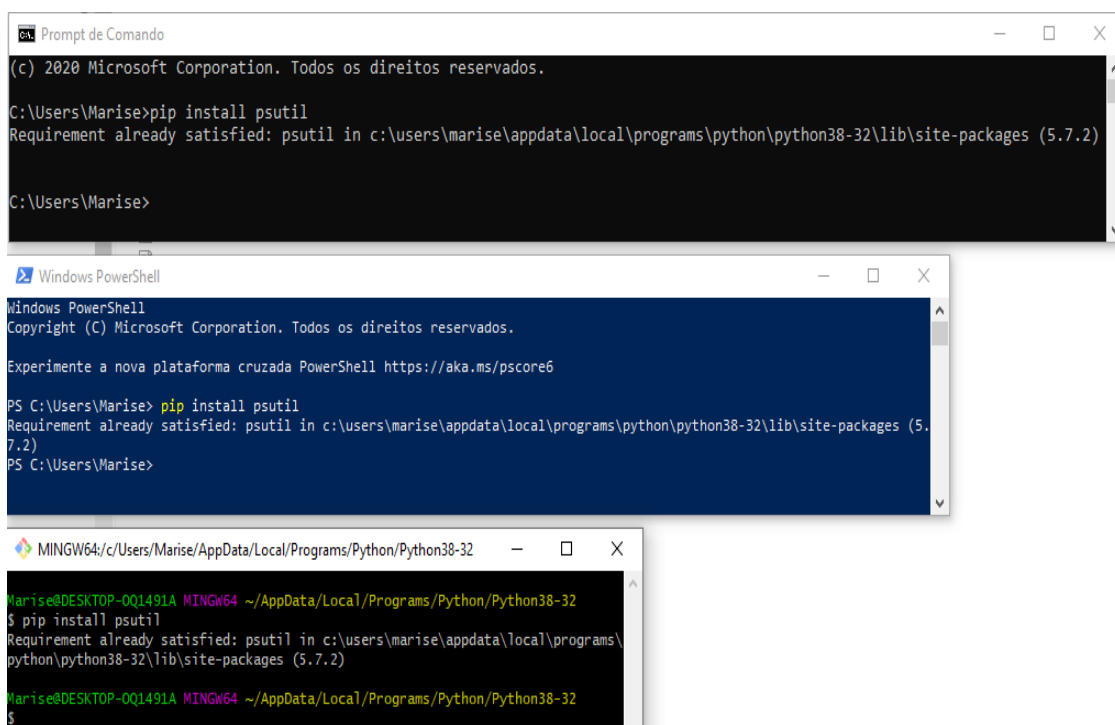
Em downloads procure a API

 **psutil-5.9.1-cp310-cp310-win_amd64**

Clique nela para executar, vai abrir o terminal e fechar. Agora vamos conferir se a lib pode ser instalada:

Pode fazer isso no cmd, shell ou gitbash (neste caso precisa ser no local onde o Python está instalado (Veja o path em users, na figura).

Comando: **pip install psutil**



```

Prompt de Comando
(c) 2020 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Marise>pip install psutil
Requirement already satisfied: psutil in c:\users\marise\appdata\local\programs\python\python38-32\lib\site-packages (5.7.2)

C:\Users\Marise>

Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

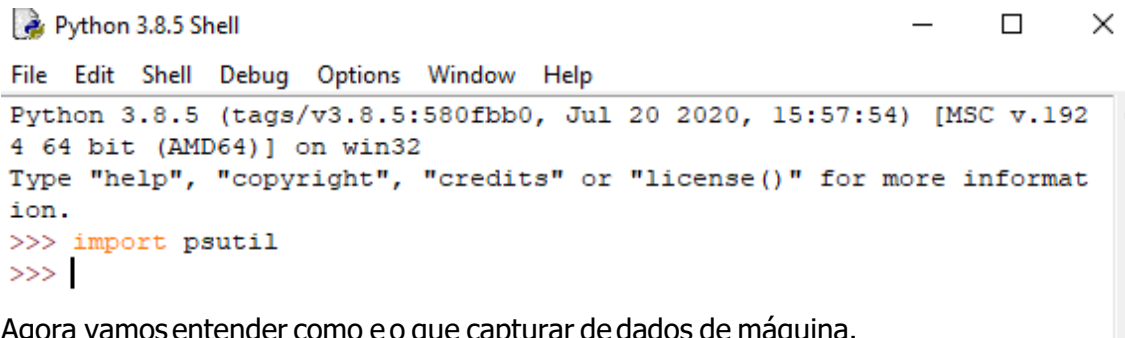
Experimente a nova plataforma cruzada PowerShell https://aka.ms/pscore6

PS C:\Users\Marise> pip install psutil
Requirement already satisfied: psutil in c:\users\marise\appdata\local\programs\python\python38-32\lib\site-packages (5.7.2)
PS C:\Users\Marise>

MINGW64/c/Users/Marise/AppData/Local/Programs/Python/Python38-32
Marise@DESKTOP-QQ1491A MINGW64 ~/AppData/Local/Programs/Python/Python38-32
$ pip install psutil
Requirement already satisfied: psutil in c:\users\marise\appdata\local\programs\python\python38-32\lib\site-packages (5.7.2)
Marise@DESKTOP-QQ1491A MINGW64 ~/AppData/Local/Programs/Python/Python38-32
$
```

Abra o interpretador IDLE

Comando: `import psutil`



```
Python 3.8.5 Shell
File Edit Shell Debug Options Window Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.192
4 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more informat
ion.
>>> import psutil
>>> |
```

Agora vamos entender como e o que capturar de dados de máquina.

CPU

`psutil.cpu_times(percpu = False)`

Vamos testar no Windows:

Retorna os tempos de CPU do sistema como uma tupla nomeada. Cada atributo representa os segundos que a CPU gastou no modo determinado. A disponibilidade dos atributos varia de acordo com a plataforma:

- **Usuário (user):** tempo gasto pelos processos normais em execução no modo usuário; no Linux, isso também inclui tempo de visitante
- **Sistema (system):** tempo gasto pelos processos em execução no modo kernel
- **Ocioso (idle):** tempo gasto sem fazer nada

```
>>> psutil.cpu_times()
```

```
scputimes(user=9854.578125,system=4752.328124999996, idle=27844.5, interrupt=227.703125, dpc=816.546875)
```

scputimes: estatística de tempo de cpu

user: tempo gasto pelo usuário

system: tempo gasto pelo sistema

idle: tempo de cpu ociosa

dpc (deferred procedures call – chamadas de procedimento adiada) - é a porcentagem de tempo que o processador gasta recebendo e atendendo chamadas de procedimento adiado. São chamadas ao processador que não responde ao processamento solicitado.

Veja que para várias solicitações da `cpu_times` os valores vão mudando. Podemos então criar uma tabela com essas informações.

```
>>> psutil.cpu_times()
scputimes(user=9854.578125, system=4752.328124999996, idle=27844.5, interrupt=227.703125, dpc=816.546875)
>>> psutil.cpu_times()
scputimes(user=10432.90625, system=4887.765625, idle=29819.234375, interrupt=233.875, dpc=842.484375)
>>> psutil.cpu_times()
scputimes(user=10433.359375, system=4888.4375, idle=29842.359375, interrupt=233.953125, dpc=842.53125)
>>> psutil.cpu_times()
scputimes(user=10434.0625, system=4889.546875, idle=29868.859375, interrupt=234.03125, dpc=842.640625)
>>> psutil.cpu_times()
scputimes(user=10435.625, system=4891.921875, idle=29916.109375, interrupt=234.203125, dpc=842.984375)
>>> psutil.cpu_times()
scputimes(user=10441.765625, system=4900.390625, idle=30034.375, interrupt=234.59375, dpc=844.625)
...
```

psutil.cpu_percent(interval = None, percpu = False)

Retorne um float que representa a utilização atual da CPU de todo o sistema como uma porcentagem. Quando o intervalo é > 0.0 compara os tempos de CPU do sistema decorridos antes e depois do intervalo (bloqueio). Quando percpu é True retorna uma lista de flutuações que representam a utilização como uma porcentagem para cada CPU. O primeiro elemento da lista se refere à primeira CPU, o segundo elemento à segunda CPU e assim por diante. A ordem da lista é consistente nas chamadas.

Faça os comandos a seguir:

```
>>> psutil.cpu_percent(interval=1)
```

29.4

```
>>> psutil.cpu_percent(interval=None)
```

36.3

```
>>> psutil.cpu_percent(interval=1, percpu=True)
```

[40.3, 26.2, 33.8, 23.1]

psutil.cpu_count(lógico = verdadeiro)

```
>>> psutil.cpu_count()
```

4

```
>>> psutil.cpu_count(logical=False)
```

2

Retorne o número de CPUs lógicas no sistema (o mesmo que os `cpu_count` no Python 3.4) ou **none** se indeterminado. Núcleos lógicos significa o número de núcleos físicos multiplicado pelo número de threads que podem ser executados em cada núcleo (isso é conhecido como Hyper Threading). Se lógicas é **false** retorna o número de núcleos físicos apenas

psutil.cpu_freq(percpu = False)

```
>>> psutil.cpu_freq()
```

scpu_freq(current=2511.0, min=0.0, max=2712.0)

Retorna a frequência da CPU como um nome duplo, incluindo as frequências atual, mínima e máxima expressas em Mhz.

psutil.cpu_freq(percpu=True)

Façam a parte de Disco e Memória. Veja as dicas a seguir:

psutil.disk_partitions()

```
[sdiskpart(device='C:\\', mountpoint='C:\\', fstype='NTFS', opts='rw,fixed'), sdiskpart(device='D:\\', mountpoint='D:\\', fstype='NTFS', opts='rw,fixed')]
```

```
>>> psutil.disk_usage('C:\\')
```

```
sdiskusage(total=261480951808, used=229084987392, free=32395964416, percent=87.6)
```

psutil.disk_usage(caminho)

Retorna estatísticas de uso de disco sobre a partição que contém o *caminho* dado como uma tupla nomeada incluindo espaço **total** , **usado** e **livre** expresso em bytes, mais a **porcentagem** de uso.

psutil.disk_partitions (all = False)

psutil.virtual_memory()