

## LINGUAGEM PYTHON

**Palavra-chave:** Python, Flexibilidade, Escalabilidade

### **Autores**

Eduardo Furlan Miranda. ORCID: <https://orcid.org/0000-0003-1200-794x>

Marcelo Andrade Bortolotto. ORCID: <https://orcid.org/0009-0004-7039-433X>

Roney Cesar Alves da Silva. ORCID: <https://orcid.org/0009-0001-0642-8564>

### **Categoria de Trabalho**

Pesquisa

### **Instituição**

Faculdade Anhanguera

### **Introdução**

A simplicidade não compromete em nada a capacidade do Python uma vez que ela conta com uma estrutura de dados avançado, com listas, dicionários complexos, além de um acervo de módulos prontos para o uso. Além dos recursos embutidos a comunidade Python é bem ativa compartilhando frameworks e bibliotecas no qual permite entender as funcionalidades conforme vão surgindo novas necessidades. Com grande flexibilidade para resolução de problemas complexos o Python oferece recursos como geradores, introspecção e persistência, sendo que a linguagem acaba suportando diversos paradigmas de programação como por exemplo: como funcional, modular e orientação a objetos. Mesmo tipos de dados básicos são tratados como objetos permitindo criar uma abordagem mais unificada para programar.

Python é uma linguagem interpretada, isso significa que o seu código é traduzido por **bytecode** sendo executado por um VM (**Máquina Virtual**) que facilita a portabilidade do código permitindo rodar em diferentes plataformas sem a necessidade de recompilar. Por ser versátil permite que desenvolvedores tenha mais flexibilidade durante o seu desenvolvimento e teste do código sem esquecer a grande comunidade ativa e o fato de ser um modelo de código aberto (**open source**), desde a sua criação em 1990 por Guido van Rossum.

A linguagem Python não é apenas utilizada no desenvolvimento de sistemas, mas também utilizada para automação e como linguagem de script para adicionar funcionalidade, rotinas dentre outras funções para outros softwares. Um exemplo a criação de extensões e scripts para aplicativos como Blender, MySQL Connector/Python, Visual Studio Code, Slack, MongoDB entre outros. Outro ponto relevante é a possibilidade que o Python permite de integração de com outras linguagens, como por exemplo C e Fortran. Isso possibilita que desenvolvedores usem as bibliotecas e recursos escritos nessas outras linguagens.

## Resultado e Discursões

Com a necessidade de desenvolver novos sistemas diariamente que fazem parte do nosso cotidiano, os desenvolvedores acabam procurando linguagem de programação que acelerem o tempo de desenvolvimento é que permita que eles sejam capazes de serem mais rápidos e integrados. No exemplo podemos notar de que forma podemos manipular uma lista e buscar informações de um determinado dado.

*Buscar um nome em uma lista*

```
# Lista de nomes para a busca
nomes = ["Alice", "Bruno", "Carla", "Diego", "Elaine"]

# Nome a ser procurado
nome_procurado = "Carla"

# Variável para indicar se o nome foi encontrado
encontrado = False

# Loop para buscar o nome na lista
for nome in nomes:
    if nome == nome_procurado:
        encontrado = True
        break # Se o nome for encontrado, saia do loop

# Resultado da busca
if encontrado:
    print(f"O nome '{nome_procurado}' foi encontrado na lista.")
else:
    print(f"O nome '{nome_procurado}' não foi encontrado na lista.")
```

Notamos o quanto a sua sintaxe e de fácil entendimento característica forte predominante no Python não esquecendo de mencionar a sua integração de bibliotecas e frameworks com mais diversas aplicações. Além disso o Python é uma linguagem usada frequentemente para aplicar funções de script, ou seja, uma linguagem de script orientada a objeto em resumo quer dizer que combina suporte POO com orientação global voltada para funções script. Uma observação pertinente é que o termo script tem significados diferentes para diferentes observadores no qual mito não preferem usar este termo. Uma vez que tendem a pensar em duas definições diferentes quando ouvem sobre Python por exemplo:

- **Ferramenta Shell:** desenvolvida para script orientados a sistemas operacionais, que são frequentemente executados a partir de linhas de comandos no console que realizam tarefas como processamento de arquivos de texto e inicialização de outros programas. realizando tarefas. Podem servir para tais funções tendo uma dezena de domínio de aplicações.
- **Linguagem de controle:** uma camada usada para controlar outros componentes do aplicativo isso é um “script”. Para testar dispositivos de hardware ou outros programas o Python pode chamar componentes que dão o acesso de baixo nível para um dispositivo. Que permite que programas executem trechos de códigos Python em pontos estratégicos para suportar a personalização de produtos para usuários finais.

Mesmo com tantas vantagens o Python apresenta certas desvantagens na sua utilização no qual podemos destacar o seu desempenho tendo em vista por ser uma linguagem interpretada pode ser mais lento do que linguagem compiladas como por exemplo: C ou C++.

No Python a escalabilidade refere-se à sua capacidade de lidar com crescentes cargas de trabalho, dados, ou tráfego sem cair significativamente o seu desempenho. Podemos citar:

- Escalabilidade horizontal: que adiciona mais serviços ou instâncias.
- Escalabilidade vertical: o aumento de recursos de um servidor.

Tabela 1 Escalabilidade



<https://www.alura.com.br/artigos/big-data>

Global Interpreter Lock (GIL), é um mecanismo específico de implementação do (CPython) que impede que uma ou mais thread acabe executado o Python ao mesmo tempo. É a principal implementação de interpretador do Python. Podemos dar como exemplo:  $x = 10$  onde cria um objeto para o número 10 é X contendo uma referência. Ao redefinir o X objeto morto None o objeto anterior (10) acaba perdendo sua referência considerado um objeto morto pelo Garbage Collector (coletor de lixo).

Devido a isso o Gil aplica a regra de execução de qualquer código Python a single lock prevenido qualquer Deadlock que transforma o código em single-thread.

Exemplo 1 single\_thread:

```

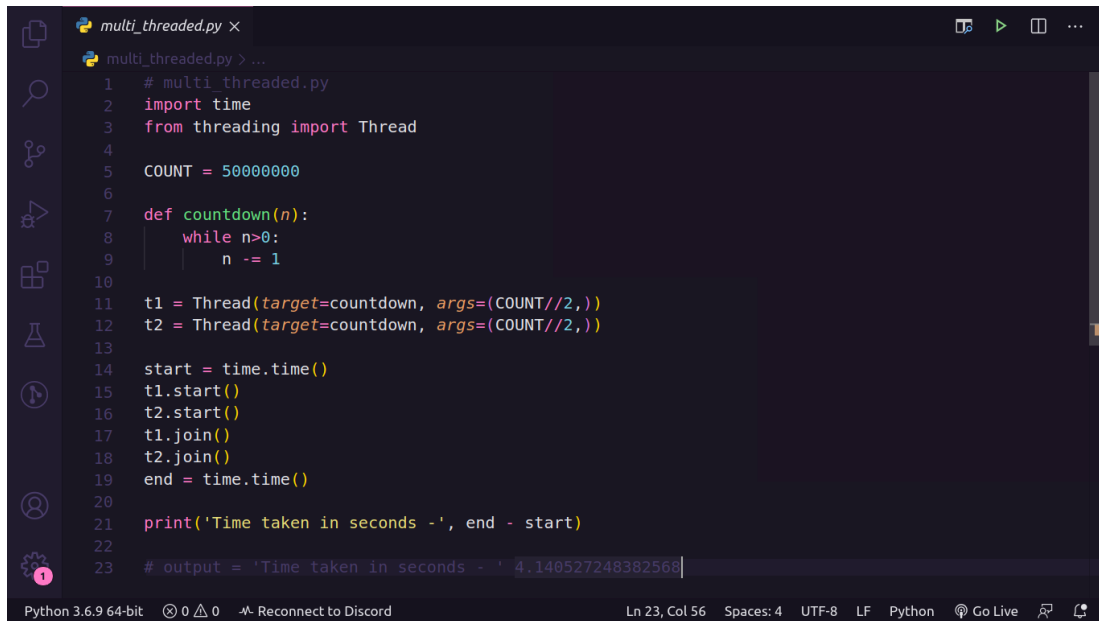
single_thread.py X
single_thread.py > ...
1 # single_threaded.py
2 import time
3 from threading import Thread
4
5 COUNT = 50000000
6
7 def countdown(n):
8     while n>0:
9         n -= 1
10
11 start = time.time()
12 countdown(COUNT)
13 end = time.time()
14
15 print('Time taken in seconds -', end - start)
16
17
18 # output = 'Time taken in seconds - ' 3.3819386959875928

```

Python 3.6.9 64-bit 0 0 Reconnect to Discord Ln 18, Col 57 Spaces: 4 UTF-8 LF Python Go Live

<https://github.com/wellington-tinho/GIL-Python>

### *Multi\_threaded: 2 threads em paralelo*

A screenshot of a code editor window titled 'multi\_threaded.py'. The code is a Python script demonstrating multi-threading with two threads. It includes imports for 'time' and 'Thread' from the 'threading' module. A constant 'COUNT' is set to 50000000. A function 'countdown(n)' is defined, which decrements 'n' until it reaches 0. Two threads, 't1' and 't2', are created, each taking 'countdown' as a target and 'COUNT//2' as an argument. The main thread records the start time, starts both threads, waits for them to finish using 'join()', and records the end time. Finally, it prints the time taken in seconds. The output line shows the time as 4.140527248382568.

```
1 # multi_threaded.py
2 import time
3 from threading import Thread
4
5 COUNT = 50000000
6
7 def countdown(n):
8     while n>0:
9         n -= 1
10
11 t1 = Thread(target=countdown, args=(COUNT//2,))
12 t2 = Thread(target=countdown, args=(COUNT//2,))
13
14 start = time.time()
15 t1.start()
16 t2.start()
17 t1.join()
18 t2.join()
19 end = time.time()
20
21 print('Time taken in seconds -', end - start)
22
23 # output = 'Time taken in seconds - ' 4.140527248382568
```

<https://github.com/wellington-tinho/GIL-Python>

Podemos notar que os exemplos acima apresentaram praticamente o mesmo tempo de execução, com a multithread demorando um pouco e causando o GIL que impediu que múltiplas threads com foco na CPU sejam executadas em paralelo.

## Conclusão

O Python vem mantendo o seu crescimento contínuo devido a facilidade de uso, a grande variedade de aplicações que pode ser implementado e não deixando de mencionar o engajamento que a comunidade implementando novidades e atualizações.

Mesmo com questões de desempenho, escalabilidade, tipagem dinâmica são assuntos que vão mantendo debate é a medida que a linguagem acaba evoluindo vai se mitigando e encontrando soluções para estes problemas.

## Código fonte

O repositório de armazenamento desta tese está armazenado no endereço **ORCID:** <https://orcid.org/0009-0001-0642-8564> registrado sobre o número de **DOI:** **10.5281/zenodo.10936031**

## Referência

ANÁLISE COMPARATIVA DE LINGUAGENS DE PROGRAMAÇÃO, Estudo computacional envolvendo linguagens de programação, São Paulo, 21, jul. 2020. Disponível, <https://revistas.unifacs.br/index.php/rsc/article/download/5133/3488> Acesso 06 mai, 2024.

O QUE É GIL, Resumo sobre Gil e exemplo, Disponível na plataforma Github, <https://github.com/wellington-tinho/GIL-Python> 06, mai, 2024.

BORGED, LUIZ EDUARDO. Python para Desenvolvedores: Érica, Rio de Janeiro, Edição do Autor, 2010. Disponível, <https://ark4n.wordpress.com/python/> .Acesso 06, mar, 2024.