



PYTHON
ACADEMY



A BIBLIOTECA OS DO PYTHON

Obtenha informações do sistema em Python com o módulo os. Aprenda a interagir com o sistema operacional para criar aplicativos poderosos e flexíveis.

[PYTHONACADEMY.COM.BR](https://pythonacademy.com.br)

Este ebook foi gerado por




Crie Ebooks técnicos incríveis em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**




Chega de formatar código no Google Docs

Deixe que nossa IA faça o trabalho pesado

 Syntax Highlight

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

TESTE AGORA 

 PRIMEIRO CAPÍTULO 100% GRÁTIS

Salve salve Pythonista 🙌

Neste artigo, vamos explorar o módulo `os` do Python e como ele pode ser utilizado para obter informações do sistema operacional.

Entender como o Python interage com o sistema operacional é fundamental para desenvolver programas mais poderosos e flexíveis.

Portanto, é importante conhecer as funcionalidades que o módulo `os` oferece.

Então bora nessa!

Introdução

O módulo `os` em Python é uma biblioteca padrão muito útil quando se trata de interagir com o sistema operacional.

Ele nos fornece uma série de funcionalidades para executar ações específicas, como navegar por diretórios, criar novos diretórios, executar comandos no terminal e obter informações do sistema.

Essas informações podem ser vitais para realizar ações específicas dependendo do sistema operacional em que o código Python está sendo executado.

Descobrimos informações do sistema

O módulo `os` nos fornece uma maneira fácil de obter informações do sistema operacional com apenas uma linha de código.

Por exemplo, para obter o nome do sistema operacional, podemos usar a função `os.name`.

```
import os

print(os.name)
```

A saída desse código será o nome do sistema operacional em que o código Python está sendo executado.

Para sistemas baseados em UNIX, a saída será `posix`, enquanto para Windows, a saída será `nt`.

Essas informações podem ser úteis para realizar ações condicionais dependendo do sistema operacional.

Outro método importante fornecido pelo módulo `os` é o `os.environ`.

Ele retorna um dicionário com as variáveis de ambiente do sistema. Podemos usá-lo para acessar variáveis específicas e seus valores.

```
import os

username = os.environ.get('USERNAME')
print(f"Username: {username}")
```

Ao executar esse código, será impresso o nome de usuário do sistema operacional em que o código está sendo executado.

Aqui no meu PC por exemplo, a saída é:

```
viniciusramos
```

Navegando por diretórios

O módulo `os` também nos permite navegar pelo sistema de arquivos e manipular diretórios.

Podemos utilizar a função `os.getcwd()` para obter o diretório atual em que o código está sendo executado.

```
import os

diretorio_atual = os.getcwd()
print(f"Diretório atual: {diretorio_atual}")
```

Essa função retorna o caminho completo do diretório atual, o que pode ser útil para realizar operações específicas em arquivos ou diretórios nesse local.

Além disso, podemos utilizar as funções `os.chdir()` e `os.mkdir()` para mudar o diretório atual e criar novos diretórios, respectivamente.

```
import os

os.chdir('caminho/para/diretorio')
os.mkdir('novo_diretorio')
```

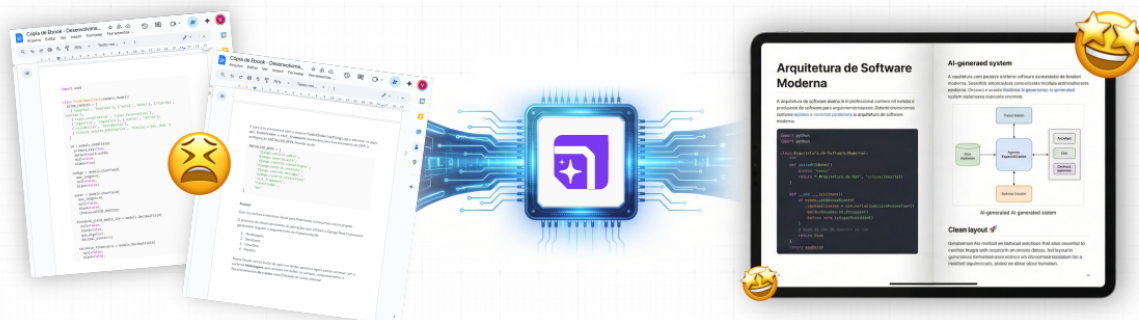
Essas operações são úteis quando precisamos manipular arquivos em diretórios específicos ou criar novos diretórios para armazenar arquivos gerados pelo nosso código.



*Estou desenvolvendo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Não deixe de conferir!*


Crie Ebooks técnicos incríveis em minutos com IA

Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs

Deixe que nossa IA faça o trabalho pesado

 Syntax Highlight

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS 

Executando comandos no terminal

O módulo `os` também nos permite executar comandos diretamente no terminal. Podemos utilizar a função `os.system()` para isso.

```
import os

os.system('ls') # comando do terminal no Linux ou macOS
```

Esse código executa o comando `ls` no terminal e exibe o resultado.

No exemplo acima, estamos usando o comando `ls` do Linux ou macOS.

Para sistemas operacionais Windows, podemos substituir o comando por algo como `dir`.

Conclusão

O módulo `os` em Python é extremamente útil para interagir com o sistema operacional de maneira eficiente.

Com ele, podemos obter informações do sistema, navegar por diretórios, criar novos diretórios e executar comandos no terminal.

Com isso, podemos criar aplicativos mais poderosos e flexíveis, adaptando-os às necessidades específicas do ambiente em que serão executados.

Neste artigo, exploramos apenas algumas funcionalidades básicas do módulo `os`.

No entanto, existem muitas outras funcionalidades disponíveis para você explorar.

Recomendo que você consulte a documentação oficial do Python para obter mais informações sobre o módulo `os` e suas possibilidades.

Agora que você possui esse conhecimento, aproveite para melhorar suas habilidades em Python e desenvolver aplicações ainda mais robustas!

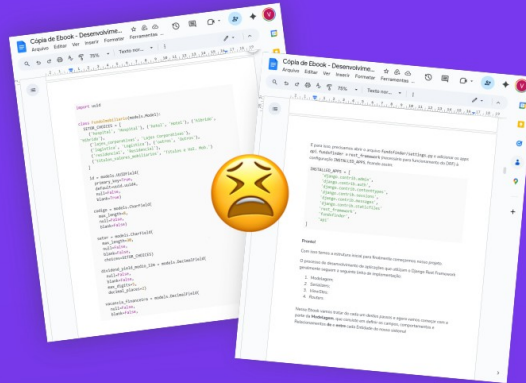
Não se esqueça de conferir!



DevBook

Crie Ebooks técnicos em minutos com IA


Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



 Syntax Highlight

 Infográficos feitos por IA

 Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado

 Edite em Markdown em Tempo Real

TESTE AGORA 

 PRIMEIRO CAPÍTULO 100% GRÁTIS