Usar rastreamentos para localizar erros

5 minutos

As exceções do Python são um recurso principal da linguagem. Você pode ficar surpreso ao ler que algo que produz erros é realçado como um recurso principal. Essa surpresa provavelmente se deve ao fato de que normalmente as ferramentas de software robustas não geram um rastreamento completo em caso de falha (várias linhas de texto que indicam como o erro foi iniciado e encerrado).

Mas as exceções são úteis porque ajudam na tomada de decisões produzindo mensagens de erro descritivas. Eles podem ajudar você a lidar com problemas esperados e inesperados.

Rastreamentos

Um *rastreamento* é o corpo do texto que pode apontar para a origem (e o término) de um erro não tratado. Noções básicas sobre os componentes de um rastreamento tornarão você mais eficaz quando estiver corrigindo erros ou depurando um programa que não está funcionando bem.

Na primeira vez que se deparar com exceções no Python, você poderá ficar tentado a evitar o erro suprimindo-o. Quando um programa apresenta um erro sem tratamento, um rastreamento é exibido como saída. Como você verá neste módulo, os rastreamentos são muito úteis. Há maneiras de lidar corretamente com os erros para que eles não apareçam ou mostrem apenas informações úteis.

Abra uma sessão interativa do Python e tente iniciar um arquivo inexistente:

Python

```
open("/path/to/mars.jpg")
```

Output

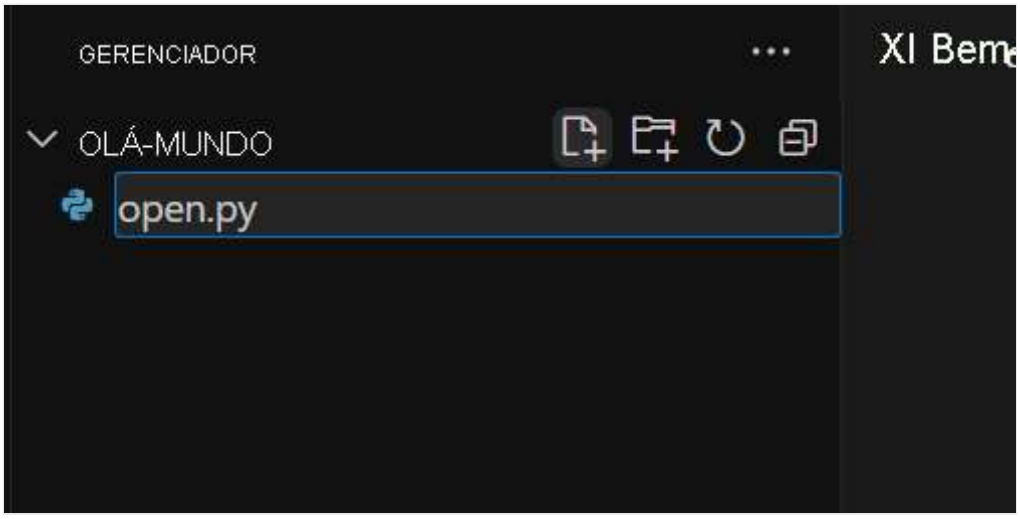
```
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: '/path/to/mars.jpg'
```

Essa saída tem várias partes principais. Primeiro, o rastreamento menciona a ordem da saída. Em seguida, ele informa que o arquivo é `stdin` (entrada no terminal interativo) na primeira

linha da entrada. O erro é `FileNotFoundError` (o nome da exceção), o que significa que o arquivo não existe ou talvez o diretório para ele não exista.

São muitas informações. Pode ser difícil entender por que a linha 1 é significativa ou o que significa `Errno 2`.

Abra o diretório desejado no Visual Studio Code e crie um arquivo Python chamado *open.py*.



Adicione o seguinte conteúdo ao arquivo e salve-o:

Python

```
def main():
    open("/path/to/mars.jpg")

if __name__ == '__main__':
    main()
```

É apenas uma função `main()` que abre o arquivo inexistente, assim como antes. No final, essa função usa um auxiliar do Python que informa ao interpretador para executar a função `main()` quando ela é chamada no terminal. Execute o comando em um terminal Bash com Python e verifique a mensagem de erro:

Bash

```
python3 open.py
```

Output

```
Traceback (most recent call last):
  File "/tmp/open.py", line 5, in <module>
    main()
  File "/tmp/open.py", line 2, in main
```

```
    open("/path/to/mars.jpg")
FileNotFoundError: [Errno 2] No such file or directory: '/path/to/mars.jpg'
```

A saída do erro faz mais sentido agora. Os caminhos estão apontando para um arquivo chamado *open.py*. A saída menciona que o erro começa na linha 5, que inclui a chamada à `main()`. Em seguida, a saída segue o erro para a linha 2 na chamada à função `open()`. E, por fim, `FileNotFoundError` novamente relata que o arquivo ou o diretório não existe.

Os rastreamentos quase sempre incluem as seguintes informações:

- Todos os caminhos de arquivo envolvidos para cada chamada a cada função.
- Números de linha associados a cada caminho de arquivo.
- Os nomes de funções, métodos ou classes envolvidas na geração de uma exceção.
- O nome da exceção que foi gerada.

Unidade seguinte: Tratar exceções

Continuar >