



Curso **Discussão** Progresso Anotações Calendário Datas

☰ Todos os tópicos

Pesquisar todas as postagens **Busca**

O acesso à auditoria expira há 5 anos, 2020

Você perde todo o acesso a este curso, incluindo o seu progresso, em 5 anos 2020 .

[Tutorial] Tente-Exceto-Outro-Finalmente: Nos Bastidores

debate postado 3 days ago por [Kiara-Elizabeth](#) (Community TA)



Tente-Exceto-Outro-Finalmente: Nos Bastidores

Oi! Bem-vindo a este breve tutorial, onde você aprenderá como o **tratamento de exceções** funciona nos bastidores quando você usa blocos **tentar-exceto-mais-finalmente** .

Vamos começar! 

Essa é a estrutura básica usada para lidar com exceções no Python (diagrama abaixo).

Temos quatro blocos:

- **try:** este é o código que você tentará executar. O que acontece depois disso depende se uma exceção foi lançada ou não.
- **exceto:** esse bloco lida com a exceção se ocorrer quando o bloco try for executado.
- **mais:** esse bloco de código é executado apenas se nenhuma exceção foi lançada no bloco try.
- **finalmente:** esse bloco sempre é executado, mesmo se houver exceções no bloco try. Conforme observado na palestra, esse bloco é muito útil para "limpar o código", como fechar arquivos.

💡 **Nota:** os blocos `else` e `finally` são opcionais. Se houver um bloco final, mais deverá ser incluído antes do bloco final.

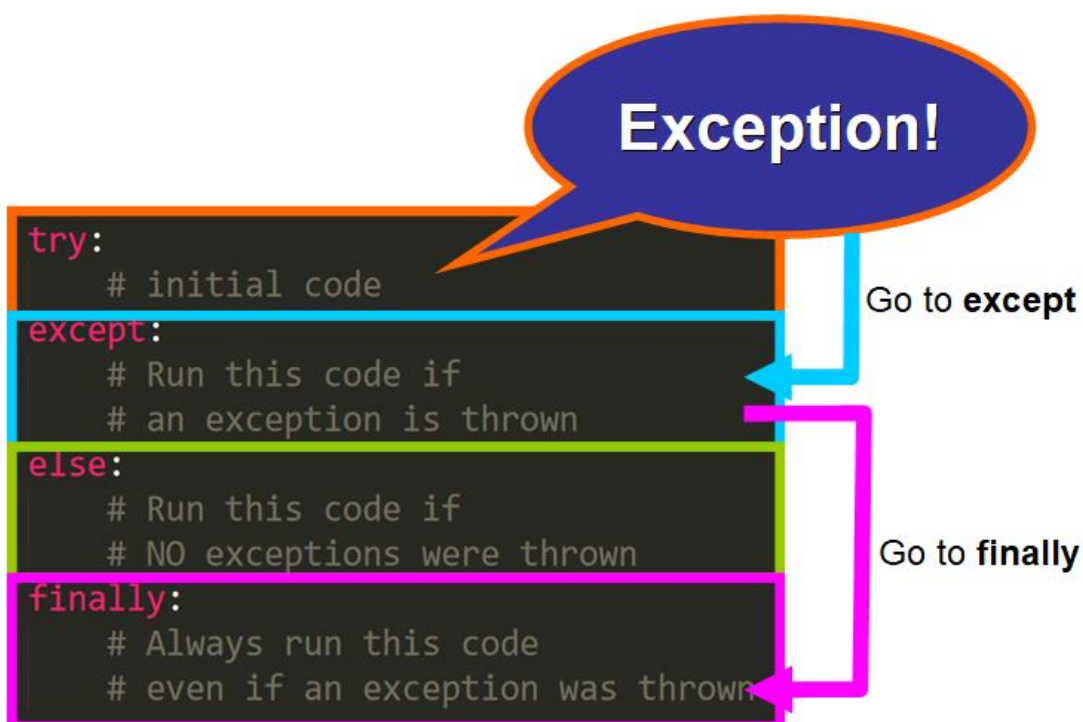
```
try:
    # initial code
except:
    # Run this code if
    # an exception is thrown
else:
    # Run this code if
    # NO exceptions were thrown
finally:
    # Always run this code
    # even if an exception was thrown
```

▶ Exceção lançada no bloco Try

Vamos ver o que acontece se uma exceção for lançada no bloco `try`.

Primeiro, o código no bloco `try` é executado. Uma exceção é lançada. O programa saltaria imediatamente para a cláusula `except`, interrompendo a execução do bloco `try`. O bloco de exceção seria executado e, em seguida, a cláusula `Finalmente` seria executada.

A sequência seria: `try -> except -> finally`



Aqui você pode ver um exemplo disso no shell Python:

Tentar dividir por 0 lança uma exceção no bloco try. O programa "salta" para o bloco de exceção e o executa. Uma vez concluído, o programa "salta" novamente para o bloco final.

```
>>> def divide(a, b):  
    try:  
        print("Running code...")  
        print(a / b)  
    except:  
        print("An exception was thrown")  
    else:  
        print("No exceptions were thrown!")  
    finally:  
        print("Final code...")  
  
>>> divide(5, 0)  
Running code...  
An exception was thrown  
Final code...
```

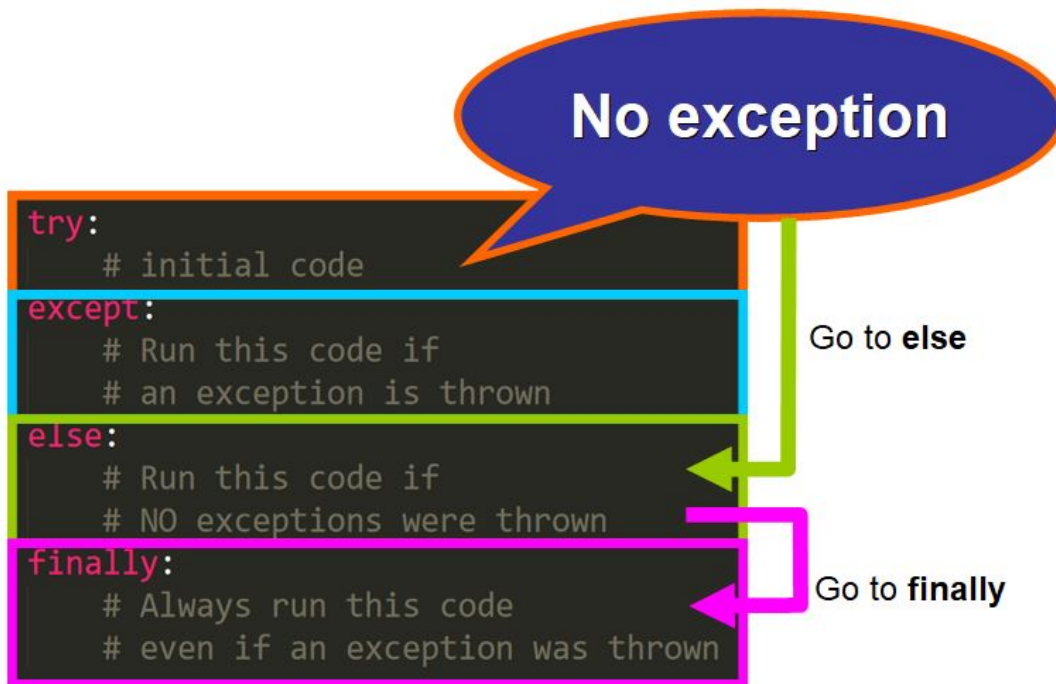
```
def divide(a, b):  
    try:  
        print("Running code...")  
        print(a / b)  
    except:  
        print("An exception was thrown")  
    else:  
        print("No exceptions were thrown!")  
    finally:  
        print("Final code...")
```

Exception!

▶ Nenhuma exceção lançada no bloco Try

Se o código no bloco try for executado sem gerar nenhuma exceção, o programa "salta" para o bloco else, executa esse bloco de código e "salta" novamente para o bloco final.

A sequência seria: `try -> else -> finally`

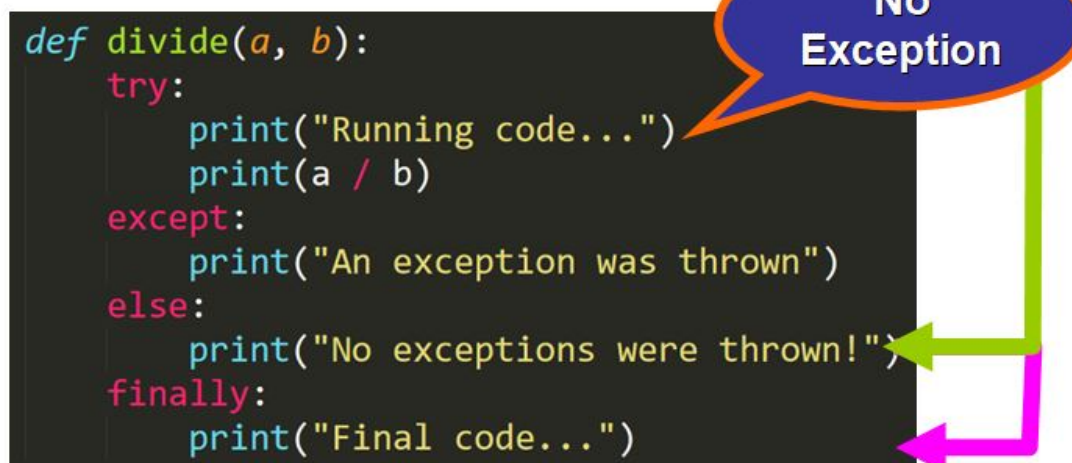


Aqui você pode ver um exemplo disso. O bloco try é executado primeiro, depois o fluxo de execução "salta" para o bloco else e a cláusula last é executada por último.

```

>>> def divide(a, b):
    try:
        print("Running code...")
        print(a / b)
    except:
        print("An exception was thrown")
    else:
        print("No exceptions were thrown!")
    finally:
        print("Final code...")

>>> divide(5, 1)
Running code...
5.0
No exceptions were thrown!
Final code...
  
```



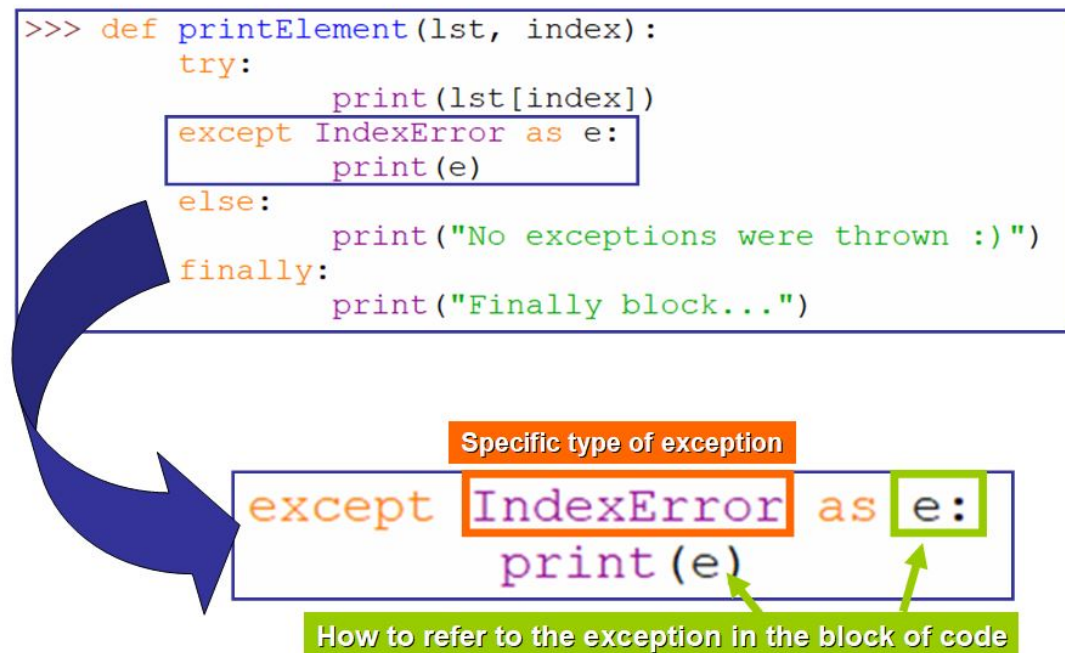
▶ Usando "Exceção como e" no bloco Except

No bloco de exceção, você pode acessar as informações da exceção lançada no bloco try usando esta sintaxe:

```
except <exception_type> as <variable_to_hold_exception>:
```

In the example below, the function specifically handles the `IndexError` exception in the except block and assigns it to the variable `e` that you can use within the except block. In this case, I decided to print the message of the exception, which you can see in the second image below.

Note: you can catch a specific type of exception such as `IndexError` or you can use a more generic form, `Exception`, which will catch most types of exceptions. This is because most of the built-in exceptions that you will use are subtypes of `Exception` (you can think of `Exception` as being a general form used to refer to an exception). You can read more about this “hierarchy” in the Python documentation: <https://docs.python.org/3/library/exceptions.html#exception-hierarchy>.



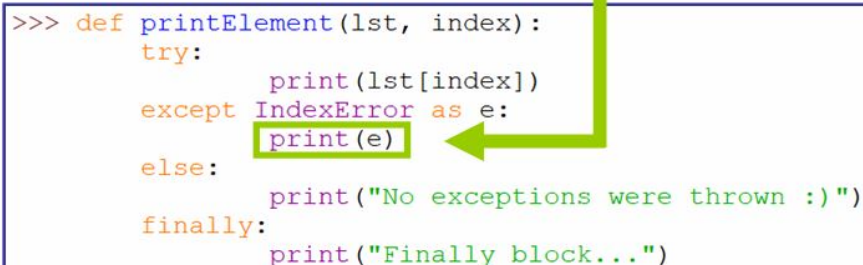
```
>>> def printElement(lst, index):
    try:
        print(lst[index])
    except IndexError as e:
        print(e)
    else:
        print("No exceptions were thrown :)")
    finally:
        print("Finally block...")
```

Specific type of exception

```
except IndexError as e:
    print(e)
```

How to refer to the exception in the block of code

```
>>> printElement([1, 2, 3, 4], 50)
list index out of range
Finally block...
```



```
>>> def printElement(lst, index):
    try:
        print(lst[index])
    except IndexError as e:
        print(e)
    else:
        print("No exceptions were thrown :)")
    finally:
        print("Finally block...")
```


Eu realmente espero que você tenha gostado deste tutorial e tenha sido útil. 😊
Gostaria de oferecer meus sinceros parabéns pelas últimas semanas de trabalho duro e dedicação. 🌟

Por favor, não hesite em perguntar nos fóruns de discussão ou logo abaixo deste post se tiver alguma dúvida. Os ATs da comunidade e seus colegas de classe estarão lá para ajudá-lo.

Stephanie.

Relacionado a: [Palestra 8 / Vídeo: Exceções](#)

Esta postagem é visível para todos.

[bilal_mu_ahmad](#)

2 days ago



Muito útil Estefania, muito obrigado. Realmente aprecio a qualidade e o esforço que você e sua equipe colocam nesses tutoriais.



Sim, muito obrigada !

Postado 2 days ago por [leopold_bernard_leo](#)

[PedroBiel](#)

2 days ago



Muito Obrigado!

Exibindo todas as respostas

Filtrar tópicos

filtrar tópicos



Todas as Discussões

★ Publicações que estou seguindo

Geral

Instalar

Recursos