



# TRATAMENTO DE ERROS E EXCEÇÕES NO PYTHON COM TRY/EXCEPT

Guia de try/except: tratamento de erros, except específico vs genérico, else, finally, raise, custom exceptions, casos práticos (input, arquivos, APIs).

# 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** 

 **Atualizado para Python 3.13 (Dezembro 2025)**

Try/except para código robusto e confiável.

Salve salve Pythonistas!

Nem sempre os programas que desenvolvemos fazem aquilo que queremos, **não é mesmo?**

Quem nunca supôs que determinada função iria retornar um dado de certo tipo, e se surpreendeu quando viu que o tipo de retorno era outro? 😰

Ou quem nunca fez uma conversão de tipos de maneira incorreta e teve um `ValueError` sendo lançado? 😰

Por esses e outros motivos que devemos **sempre** tratar as exceções, programando de forma **defensiva** em Python!

E para isso, utilizamos o bloco **try/except**.

Na verdade, é o bloco **try/except/else/finally**.

Mas calma que veremos tudo nesse post!

Então sem enrolação, já prepara o café e **bora nessa!**

## O tratamento de exceções no Python

O bloco básico de código que realiza o tratamento de exceções no *Python* é feito da seguinte forma:

```
try:  
    # Bloco de código a ser executado  
  
except {exceção}:  
    # Código que será executado caso a {exceção} seja capturada  
  
else:  
    # Código que será executado caso nenhuma exceção tenha sido  
    # lançada ou capturada  
  
finally:  
    # Código que será executado independente se alguma exceção for cap-  
    # turada ou não
```

Apenas as cláusulas `try` e `except` são obrigatórias, sendo `else` e `finally` opcionais!

Podemos ter múltiplas cláusulas `except`, capturando Exceções diferentes.

Mas vamos primeiro aos exemplos para facilitar seu entendimento!

## Exemplos de utilização

Suponha que você queira abrir um arquivo e caso ele não exista, seu programa irá criá-lo.

Uma forma de fazer isso seria:

```
nome_arquivo = 'nome_arquivo.txt'

try:
    arquivo = open(nome_arquivo, 'r')

except FileNotFoundError:
    arquivo = open(nome_arquivo, 'a')

else:
    print(f"Arquivo {nome_arquivo} já existe")

finally:
    # Realiza algum processamento no arquivo
    processa_arquivo(arquivo)

    # Fecha o arquivo
    arquivo.close()
```

Dessa forma, caso o arquivo não exista, uma exceção `FileNotFoundException` será lançada e será capturada na cláusula `except`.

Ali dentro, a função `open` com o parâmetro “a” (de “*append*”) será chamada, criando então o arquivo.

Em seguida, temos a cláusula `finally`, que será executada independentemente de uma exceção ter sido lançada ou não.

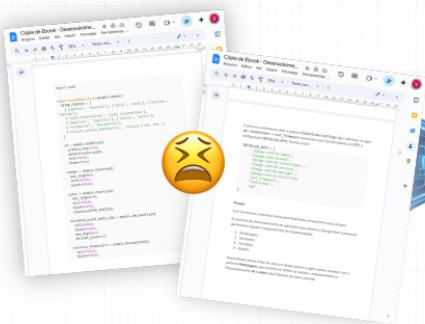
Como se trata de um arquivo, é sempre aconselhado realizar seu fechamento, e um bom lugar para fazer isso é dentro do `finally`.



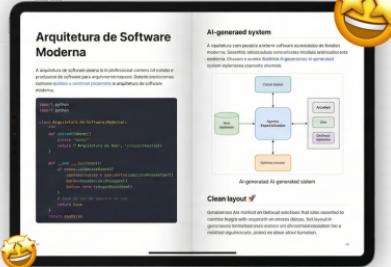
*Estou desenvolvendo o **DevBook**, uma plataforma que usa IA para gerar ebooks técnicos profissionais. Te convido a conhecer!*

## Crie Ebooks técnicos incríveis em minutos com IA

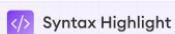
Conheça a 1ª IA Especializada na criação de Ebooks **com código!**



Chega de formatar código no Google Docs



Deixe que nossa IA faça o trabalho pesado



Syntax Highlight



Adicione Banners Promocionais



Edit in Markdown em Tempo Real



Infográficos feitos por IA

**TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS**

## Capturando múltiplas exceções

Podemos capturar múltiplas exceções de duas formas:

A primeira, com múltiplas cláusulas `except`, assim:

```
try:
    dividendo = int(input("Digite o dividendo: "))
    divisor = int(input("Digite o divisor: "))
    resultado = dividendo/divisor

except ValueError:
    print("Número digitado inválido")

except ZeroDivisionError:
    print("Divisão por zero não permitida")
```

Assim, caso o erro `ValueError` seja lançado, este será capturado pelo primeiro `except`.

E caso haja uma divisão por zero, a exceção `ZeroDivisionError` será lançada e capturada pelo segundo `except`.

Podemos também agrupar exceções! Dessa forma, o mesmo tratamento será dado independente de qual exceção tenha sido lançado, dessa forma:

```
try:  
    dividendo = int(input("Digite o dividendo: "))  
    divisor = int(input("Digite o divisor: "))  
    resultado = dividendo/divisor  
  
except(ZeroDivisionError, ValueError):  
    print("Erro de conversão ou divisor igual à zero")
```

Viu como é fácil tratar erros e exceções no Python?!

Agora você não tem mais desculpa para programar da **maneira correta!**

## Conclusão

Nesse post você viu como pode fazer o tratamento de erros e exceções no Python com o bloco `try / except` e ainda como utilizar as cláusulas `else` e `finally`!

Programar de maneira defensiva - capturando e tratando os erros dos programas que desenvolve - é algo que você precisa passar a fazer para progredir como Dev Python!

Espero que tenha curtido esse conteúdo e nos vemos na próxima 😊

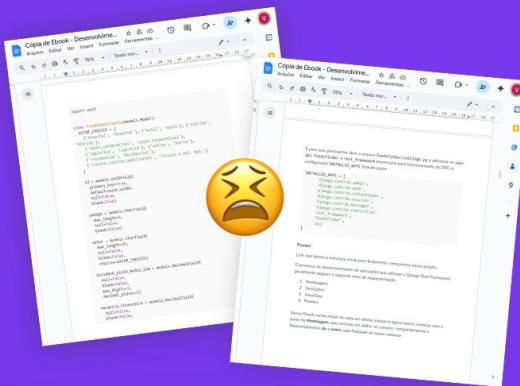
Não se esqueça de conferir!



# 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



*Adicione Banners Promocionais*



• Infográficos feitos para...

Deixe que nossa IA faça o trabalho pesado



 Edite em Markdown em Tempo Real

**TESTE AGORA**



 PRIMEIRO CAPÍTULO 100% GRÁTIS