# Python - Bloco de tentativa aninhado

Em um programa Python, se houver outra construção **try-except** dentro de um bloco **try** ou dentro de seu bloco **except** , ele é conhecido como bloco try aninhado. Isso é necessário quando blocos diferentes, como externo e interno, podem causar erros diferentes. Para lidar com eles, precisamos de blocos try aninhados.

Começamos com um exemplo com uma única construção "tentar - exceto - finalmente". Se as instruções dentro de try encontrarem uma exceção, ela será tratada pelo bloco except. Com ou sem exceção, o bloco final é sempre executado.

## Exemplo 1

Aqui, o bloco **try** tem a situação de "divisão por 0", portanto o bloco **except** entra em jogo. Está equipado para lidar com a exceção genérica com a classe Exception.

```
a=10
b=0
try:
   print (a/b)
except Exception:
   print ("General Exception")
finally:
   print ("inside outer finally block")
```

Ele produzirá a seguinte **saída** -

```
General Exception
inside outer finally block
```

## Exemplo 2

Vamos agora ver como aninhar as construções **try** . Colocamos outros blocos "try - exceto - finalmente" dentro do bloco try existente. A palavra-chave exceto para tentativa interna agora lida com exceção genérica, enquanto solicitamos ao bloco exceto de tentativa externa para lidar com ZeroDivisionError.

Como a exceção não ocorre no bloco **try** interno , seu Except genérico correspondente não é chamado. A situação de divisão por 0 é tratada pela cláusula outer except.

```
a=10
b=0
```

```
try:
   print (a/b)
   try:
      print ("This is inner try block")
   except Exception:
      print ("General exception")
   finally:
      print ("inside inner finally block")

except ZeroDivisionError:
   print ("Division by 0")
finally:
   print ("inside outer finally block")
```

Ele produzirá a seguinte **saída** -

```
Division by 0
inside outer finally block
```

## Exemplo 3

Agora invertemos a situação. Dos blocos **try** aninhados , o externo não tem nenhuma exceção levantada, mas a instrução que causa a divisão por 0 está dentro do try interno e, portanto, a exceção tratada pelo bloco inner **except** . Obviamente, a parte **except** correspondente a outer **try** : não será chamada.

```
a=10
b=0
try:
   print ("This is outer try block")
   try:
      print (a/b)
   except ZeroDivisionError:
      print ("Division by 0")
   finally:
      print ("inside inner finally block")

except Exception:
   print ("General Exception")
finally:
   print ("inside outer finally block")
```

Ele produzirá a seguinte **saída** -

```
This is outer try block
Division by 0
inside inner finally block
inside outer finally block
```

Ao final, vamos discutir outra situação que pode ocorrer no caso de blocos aninhados. Embora não haja nenhuma exceção no **try** : externo, não há um bloco except adequado para lidar com aquele dentro do bloco **try** : interno.

## Exemplo 4

No exemplo a seguir, o **try** interno : enfrenta "divisão por 0", mas seu correspondente except: está procurando por KeyError em vez de ZeroDivisionError. Conseqüentemente, o objeto de exceção é passado para o bloco except: da instrução except subsequente que corresponde à instrução try: externa. Lá, a exceção zeroDivisionError é capturada e tratada.

```
a=10
b=0
try:
   print ("This is outer try block")
   try:
      print (a/b)
   except KeyError:
      print ("Key Error")
   finally:
      print ("inside inner finally block")

except ZeroDivisionError:
   print ("Division by 0")
finally:
   print ("inside outer finally block")
```

Ele produzirá a seguinte **saída** -

```
This is outer try block
inside inner finally block
Division by 0
inside outer finally block
```