Tratamento de Exceções em Python

Tempo estimado necessário: 10 Minutos

Objetivos

1. Compreender Exceções
2. Distinguir Erros de Exceções
3. Familiaridade com Exceções Comuns do Python
4. Gerenciar Exceções de Forma Eficaz

Quais são as exceções?

Exceções são alertas quando algo inesperado acontece durante a execução de um programa. Pode ser um erro no código ou uma situação que não foi planejada. O Python pode gerar esses alertas automaticamente, mas também podemos acioná-los intencionalmente usando o comando raise. A parte interessante é que podemos evitar que nosso programa falhe ao tratar exceções.

Erros vs. Exceções

Espere, qual é a diferença entre erros e exceções? Bem, `erros` geralmente são grandes problemas que vêm do computador ou do sistema. Eles costumam fazer o programa parar de funcionar completamente. Por outro lado, `exceções` são mais como problemas que podemos controlar. Elas acontecem por causa de algo que fizemos em nosso código e geralmente podem ser corrigidas, permitindo que o programa continue.

Aqui está a diferença entre `Erros e exceções`:-

Aspecto	Erros	Exceções
Origem	Erros são tipicamente causados pelo ambiente, hardware ou sistema operacional.	Exceções geralmente são o resultado da execução problemática do código dentro do programa.
Natureza	Erros são frequentemente severos e podem levar a falhas do programa ou término anormal.	Exceções são geralmente menos severas e podem ser capturadas e tratadas para evitar a terminação do programa.
Tratamento	Erros não costumam ser capturados ou tratados pelo próprio programa.	Exceções podem ser capturadas usando blocos try-except e tratadas de forma elegante, permitindo que o programa continue a execução.
Exemplos	Exemplos incluem “SyntaxError” devido a uma sintaxe incorreta ou “NameError” quando uma variável não está definida.	Exemplos incluem “ZeroDivisionError” ao dividir por zero, ou “FileNotFoundError” ao tentar abrir um arquivo inexistente.
Classificação	Erros não são classificados em categorias.	Exceções são categorizadas em várias classes, como “ArithmeticError,” “IOError,” “ValueError,” etc., com base em sua natureza.

Exceções Comuns em Python

Aqui estão alguns exemplos de exceções que frequentemente encontramos e podemos tratar usando esta ferramenta:

- **ZeroDivisionError:** Este erro ocorre quando se tenta dividir um número por zero. A divisão por zero é indefinida na matemática, causando um erro aritmético.
  Por exemplo:

```
result = 10 / 0
print(result)
# Raises ZeroDivisionError
```

- **ValueError:** Este erro ocorre quando um valor inadequado é usado dentro do código. Um exemplo disso é quando se tenta converter uma string não numérica em um inteiro:
  Por exemplo:

```
num = int("abc")
# Raises ValueError
```

- **FileNotFoundError:** Esta exceção é encontrada quando se tenta acessar um arquivo que não existe.
  Por exemplo:

```
with open("nonexistent_file.txt", "r") as file:
        content = file.read()   # Raises FileNotFoundError
```

- **IndexError:** Um IndexError ocorre quando um índice é usado para acessar um elemento em uma lista que está fora do intervalo de índices válidos. Por exemplo:

```
my_list = [1, 2, 3]
value = my_list[1]  # No IndexError, within range
missing = my_list[5]  # Raises IndexError
```

- **KeyError:** O KeyError ocorre quando se tenta acessar uma chave inexistente em um dicionário.
  Por exemplo:

```
my_dict = {"name": "Alice", "age": 30}
value = my_dict.get("city")  # Sem KeyError, usando o método .get()
missing = my_dict["city"]  # Levanta KeyError
```

- **TypeError:** O TypeError ocorre quando um objeto é usado de maneira incompatível. Um exemplo inclui tentar concatenar uma string e um inteiro:
  Por exemplo:

```
result = "hello" + 5
# Raises TypeError
```

- **AttributeError:** Um AttributeError ocorre quando um atributo ou método é acessado em um objeto que não possui esse atributo ou método específico. Por exemplo:

```
text = "example"
length = len(text)  # No AttributeError, correct method usage
missing = text.some_method()  # Raises AttributeError
```

- **ImportError:** Este erro é encontrado quando se tenta importar um módulo que não está disponível. Por exemplo: `import non_existent_module`

Nota: **Por favor, lembre-se, as exceções que você encontrará não estão limitadas apenas a estas. Existem muitas outras em Python. No entanto, não há necessidade de se preocupar. Usando a técnica fornecida abaixo e seguindo a sintaxe correta, você será capaz de lidar com quaisquer exceções que surgirem.**

Tratamento de Exceções:

Python possui uma ferramenta útil chamada `try and except` que nos ajuda a gerenciar exceções.

Try and Except: Você pode usar os blocos try e except para evitar que seu programa falhe devido a exceções.

Aqui está como eles funcionam:

1. O código que pode resultar em uma exceção está contido no bloco try.
2. Se uma exceção ocorrer, o código salta diretamente para o bloco except.
3. No bloco except, você pode definir como lidar com a exceção de forma elegante, como exibindo uma mensagem de erro ou tomando ações alternativas.
4. Após o bloco except, o programa continua executando o restante do código.

Exemplo: Tentando dividir por zero

```
# usando Try- except
try:
    # Tentando dividir 10 por 0
    result = 10 / 0
except ZeroDivisionError:
    # Tratando o ZeroDivisionError e imprimindo uma mensagem de erro
    print("Erro: Não é possível dividir por zero")
# Esta linha será executada independentemente de uma exceção ter ocorrido
print("fora do bloco try e except")
```

Next Step

As we finish up this reading, you are ready to move on to the next part where you will practice handling errors. For better learning, try out different types of data in the lab. This way, you will encounter various errors and learn how to deal with them effectively. This knowledge will help you write stronger and more reliable code in the future.

Author(s)

Akansha Yadav