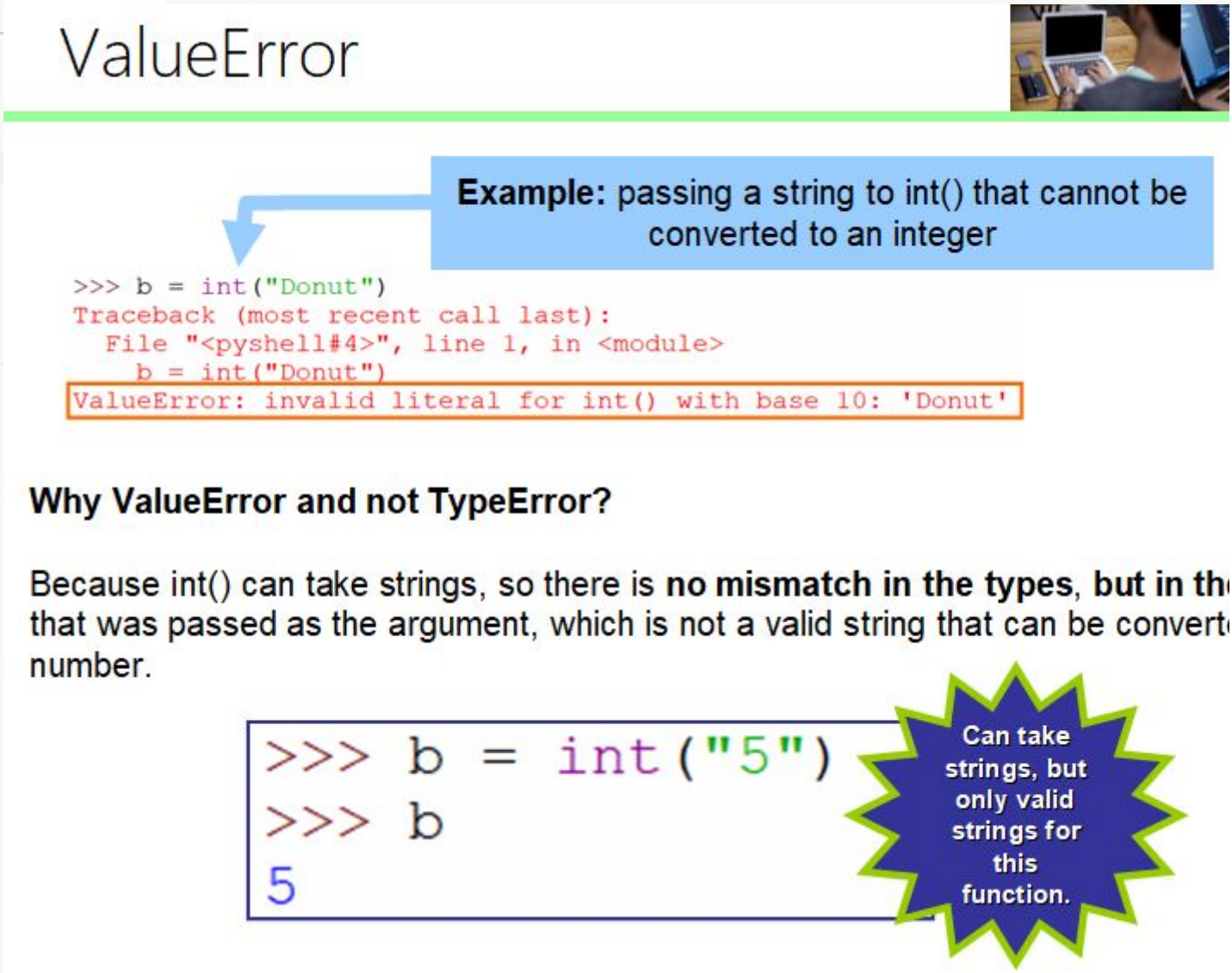
erro de valor 1



Este tópico é essencial para o processo de depuração. Quando há erros ou bugs em um programa (sintaxe ou erros lógicos), geralmente é possível detectá-los e corrigi-los lendo a mensagem de erro exibida. É como trabalhar como detetive. Você precisa ler a mensagem de erro, interpretá-la e tentar adivinhar o que pode estar causando o problema. É realmente muito divertido se você pensar sobre isso. Você pode consertar o programa resolvendo o "quebra-cabeça" e geralmente aprendemos muito com longas sessões de depuração. Você precisa pesquisar, experimentar, testar e corrigir, para que sejam oportunidades incríveis de aprender.

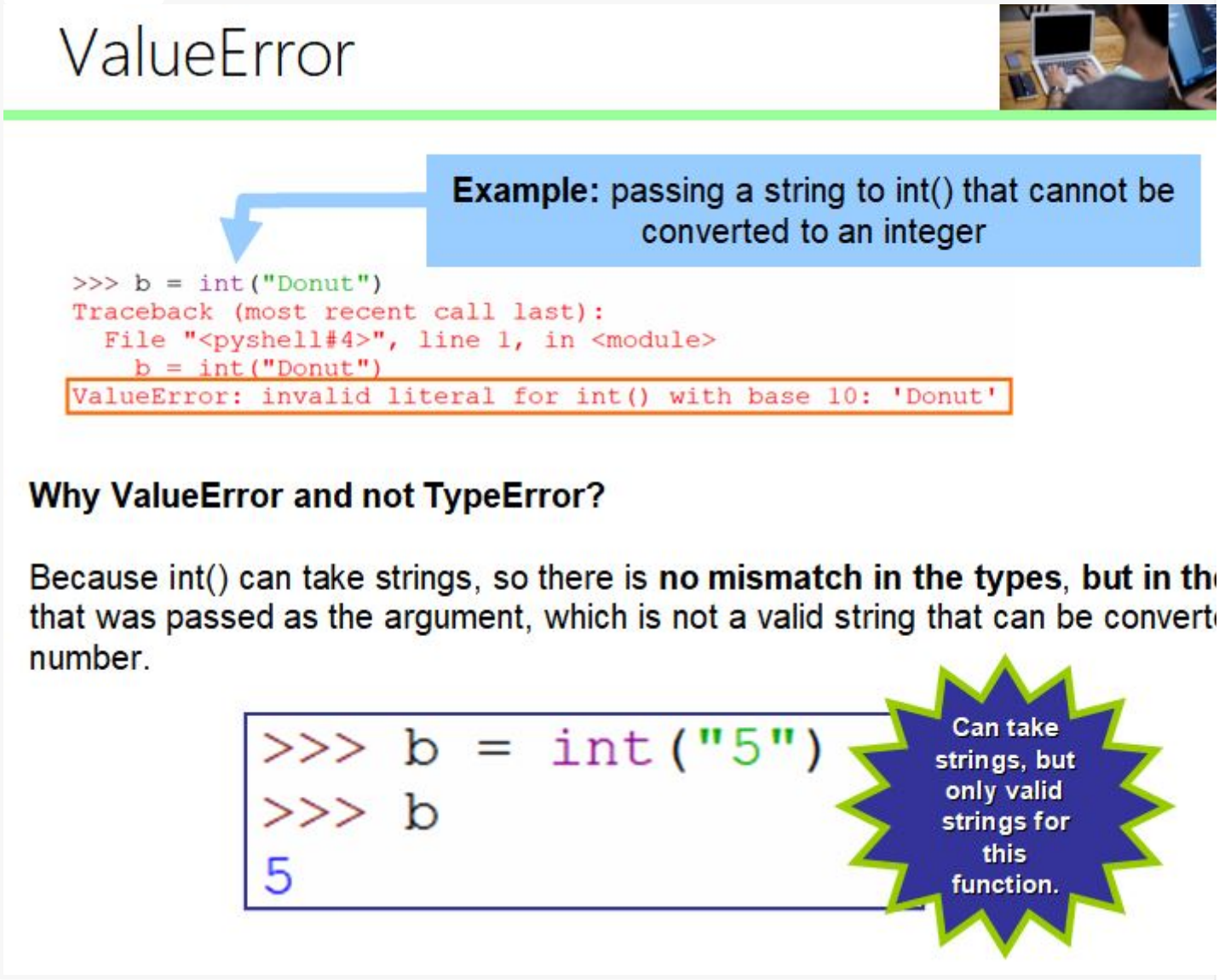
Tudo começa com uma mensagem de erro simples, e é isso que discutiremos nesta breve introdução, alguns dos diferentes tipos de mensagens de erro que você pode receber em um programa Python. Vamos começar!

Vocabulário

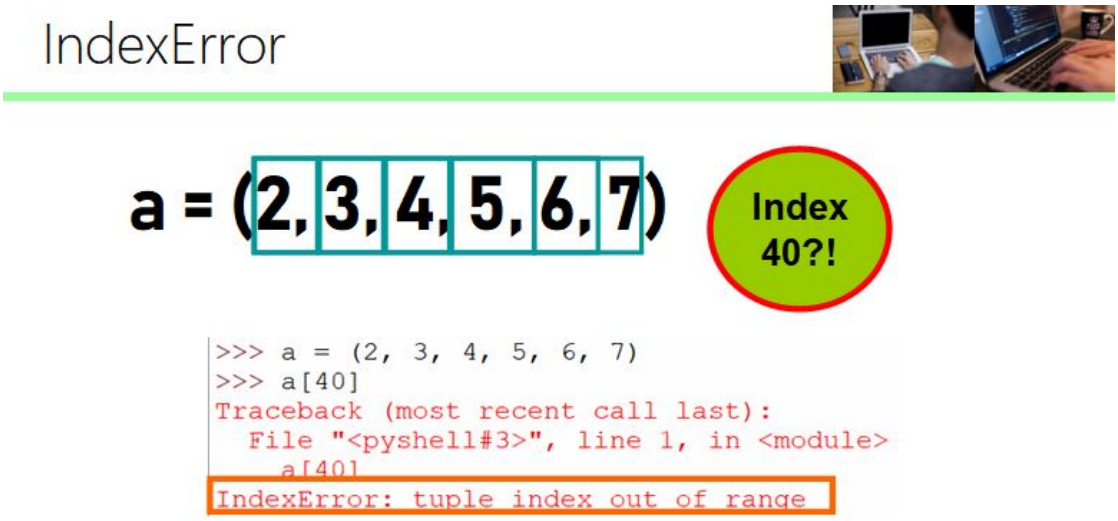
- **Exceção** : uma exceção é um erro ou evento inesperado que ocorre enquanto o programa está em execução e altera o fluxo de execução.
- **Bug** : erro lógico em um programa.

- **Depuração** : o processo de encontrar e corrigir erros no seu código.

erro de valor 1



Este é um exemplo com uma tupla:



🏳 TypeError

Um TypeError é uma exceção criada quando uma operação é aplicada a um valor de um tipo inadequado.

Por exemplo, você poderá ver essa exceção se passar um argumento do tipo de dados errado para uma função interna do Python, como na imagem abaixo, onde uma lista como argumento para int(). Essa função converte o argumento

erro de valor 1

ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

Example: passing a number to the built-in function len()

```
>>> len(5)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    len(5)
TypeError: object of type 'int' has no len()
```

Aqui você pode ver um exemplo dessa exceção quando tentamos concatenar um número inteiro com uma tupla. O tipo de dados do operando não é apropriado para a operação.

TypeError

erro de valor 1

ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

permitido.

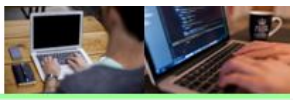
De acordo com a documentação do Python:

"Gerado quando uma operação ou função recebe um argumento que tem o tipo certo, mas um valor inadequado, e a situação não é descrita por uma exceção mais precisa, como IndexError".
https://docs.python.org/3/library/exceptions.html#ValueError

Por exemplo, na imagem abaixo, vemos essa exceção porque passamos a string "Donut" para a função interna int (). Você pode perguntar, por que não obtemos um TypeError? Isso ocorre porque a função int () pode receber cadeias de caracteres, mas essas cadeias precisam representar números para serem convertidos (convertidos) em números inteiros (por exemplo, “56”). Nesse caso, a sequência (valor) não é válida.

ValueError

erro de valor 1



ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

items than the ones contained in a list.

```
>>> name, age, address = ["Amy", 5]
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    name, age, address = ["Amy", 5]
ValueError: not enough values to unpack (expected 3, got 2)
```

NameError

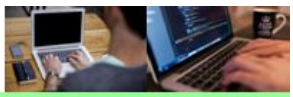
This exception, NameError, is raised when a local or global variable is used or accessed without being defined first, when you are trying to access it before it even exists in the scope.

According to the Python documentation:

“Raised when a local or global name is not found. This applies only to unqualified names. The associated value is an error message that includes the name that could not be found.” https://docs.python.org/3/library/exceptions.html#NameError

NameError

erro de valor 1



ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

Not following Python syntax rules

```
>>> * 5 6
SyntaxError: invalid syntax
```

Eu realmente espero que você tenha gostado deste tutorial e tenha sido útil 😀. 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.

1 resposta

leopold_bernard_leo

erro de valor 1

ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

Lendo

Vídeo: Variáveis Globais

Leitura 1

Vídeo: Idiomas

Exercício 10

Vídeo: Variáveis

edX

Sobre

erro de valor 1 siness

ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

erro de valor 1

ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

erro de valor 1

ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

erro de valor 1

ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

erro de valor 1

ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

erro de valor 1

ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.

erro de valor 1

ValueError

Example: passing a string to int() that cannot be converted to an integer

```
>>> b = int("Donut")
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    b = int("Donut")
ValueError: invalid literal for int() with base 10: 'Donut'
```

Why ValueError and not TypeError?

Because int() can take strings, so there is **no mismatch in the types**, **but in th** that was passed as the argument, which is not a valid string that can be convert number.

```
>>> b = int("5")
>>> b
5
```

Can take strings, but only valid strings for this function.