

Live de Python #60

Exceções

Ajude a Live de Python

apoia.se/LiveDePytho

picPay: @livedepython

Roteiro

- LBYL x EAFP
- O que são exceções?
- Lidando com exceções
 - Try / except
 - else
 - finally
- O que gera uma exceção?
- Exceções nativas
- Criando minhas próprias exceções

Referências:

1. <https://docs.python.org/3/tutorial/errors.html>
2. <https://docs.python.org/3/library/exceptions.html>
3. <https://docs.python.org/3/glossary.html>
4. Python eficaz - Brett Slatkin (Cap 1.13) - Novatec

LBYL x EAFP

- look before you leap
 - Olhe antes de saltar
- Easier to ask for forgiveness than permission
 - Mais fácil tentar do que pedir permissão

LBYL x EAFP

- look before you leap
 - Olhe antes de saltar
- Easier to ask for forgiveness than permission
 - Mais fácil tentar do que pedir permissão

```
In [1]: dicionario = {}  
  
In [2]: # LBYL  
  
In [3]: if 'a' in dicionario:  
...:     print(dicionario['a'])  
...: else:  
...:     print('a não está no dicionário')  
...:  
a não está no dicionário
```

LBYL x EAFP

- look before you leap
 - Olhe antes de saltar
- Easier to ask for forgiveness than permission
 - Mais fácil tentar do que pedir permissão

```
In [1]: dicionario = {}  
  
In [2]: # EAFP  
  
In [3]: try:  
...:     print(dicionario['a'])  
...: except:  
...:     print('a não está no dicionário')  
...:  
a não está no dicionário
```

O que são exceções?

Diferentemente de erros, temos exceções. Exceções são disparadas mesmo quando a sintaxe está correta.

Exceções são levantadas quando não é possível fazer aquilo que se espera.

O que são exceções? (exemplos)

Perguntaram ao guido uma vez, qual a exceção preferida dele. Ele respondeu:

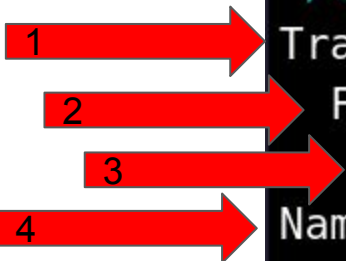
KeyboardInterrupt

Então, se é a preferida do guido, vamos começar por ela. SUAUSAHSUAH

O que são exceções?

```
~ >>> python 18:49:22
Python 3.6.6 (default, Jun 27 2018, 13:11:40)
[GCC 8.1.1 20180531] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
KeyboardInterrupt
```

O que são exceções?



```
~/live60 >>> python exemplo_1.py
Traceback (most recent call last):
  File "exemplo_1.py", line 1, in <module>
    name # NameError
NameError: name 'name' is not defined
```

1 Traceback (most recent call last):

2 File "exemplo_1.py", line 1, in <module>

3 name # NameError

4 NameError: name 'name' is not defined

O que são exceções?

```
~/live60 >>> python exemplo_1.py  
Traceback (most recent call last):  
  File "exemplo_1.py", line 1, in <module>  
    name # NameError  
NameError: name 'name' is not defined
```

O que são exceções?

```
~/live60 >>> python exemplo_1.py  
Traceback (most recent call last):  
  File "exemplo_1.py", line 1, in <module>  
    name # NameError  
NameError: name 'name' is not defined
```

Lidando com exceções

É possível lidar com exceções genéricas e com exceções específicas usando as palavras reservadas **try** e **except**.

```
while True:
    try:
        nome = input('Digite seu nome: ')
        print(f'Olá {nome}')
    except:
        print('\nTudo bem, já entendi, você não quer brincar')
        exit()
```

Lidando com exceções

É possível lidar com exceções genéricas e com exceções específicas usando as palavras reservadas **try** e **except**.

```
while True:
    try:
        nome = input('Nome: ')
        print(f'Olá, {nome}!')
    except:
        print('\nTudo bem, já entendi, você não quer brincar')
        exit()
```

Forma genérica

Lidando com exceções

É possível lidar com exceções genéricas e com exceções específicas usando as palavras reservadas **try** e **except**.

```
while True:
    try:
        nome = input('Digite o nome: ')
        print(f'Olá {nome}!')
    except KeyboardInterrupt:
        print('\nTudo bem, já entendi, você não quer brincar')
        exit()
```

**Forma para
específicas**

Lidando com exceções

É possível lidar com exceções genéricas e com exceções específicas usando as palavras reservadas **try** e **except**.

```
while True:
    try:
        nome = input('Digite seu nome: ')
        print(f'Olá {nome}')
        0/0
    except KeyboardInterrupt:
        print('\nTudo bem, já é')
        exit()
    except ZeroDivisionError as ex:
        print(ex)
```

**Captura a mensagem
de erro**

Fluxo do try

1. O bloco try/except é invocado
2. Caso não ocorra uma exceção em try, except é ignorado
3. Caso ocorra um erro mapeado por algum except, ele será executado
4. Caso a exceção não seja mapeada, o sistema vai apresentar um erro

Else

A cláusula **else** existente no try, será executada caso tudo ocorra com sucesso.

Por exemplo:

```
try:
    f = open('arquivo')
except FileNotFoundError:
    print('Arquivo não existe')
else:
    print(f.read())
    f.close()
```

finally

A cláusula **finally**, diferente da **else** será executada independente da execução dar certo ou não

```
try:
    f = open('arquivo')
except FileNotFoundError:
    print('Arquivo não existe')
else:
    print(f.read())
    f.close()
finally:
    print('Vou executar com erro ou sem')
```

O que gera uma exceção?

Para forçar uma exceção você pode usar a instrução **raise**. Ela é responsável por levantar qualquer tipo de exceção. Seja ela nativa ou criada por você (calma, vamos chegar lá)

```
raise NameError('Seu nome não é Luiza, aquela do Canadá?')
```

```
~/live60 >>> python exemplo_4.py
Traceback (most recent call last):
  File "exemplo_4.py", line 1, in <module>
    raise NameError('Seu nome não é Luiza, aquela do Canadá?')
NameError: Seu nome não é Luiza, aquela do Canadá?
```

Criando minhas próprias exceções

Uma exceção customizada, só precisa herdar de **Exception**. E uma classe de exceção pode fazer toda e qualquer coisa. Sua instância será retornada no **as** do **except**

```
class MeuErro(Exception):  
    def __str__(self):  
        return 'Eu quis dizer, você não quis escutar'
```

```
~/live60 >>> python exemplo_5.py  
Traceback (most recent call last):  
  File "exemplo_5.py", line 6, in <module>  
    raise MeuErro  
__main__.MeuErro: Eu quis dizer, você não quis escutar
```

BÔNUS

Debugger