

Exceptions

E porque não são só para erros

```
In [2]: def foo(){  
        return 'bar'  
        }
```

```
File "<ipython-input-2-82e397173a8e>", line 1
```

```
def foo(){  
    ^
```

```
SyntaxError: invalid syntax
```

Ok, SyntaxError é um erro

**São um sinal de que um comportamento não
esperado ocorreu**

In [3]: *# Um exemplo de estrutura de dados, uma lista com 2 elementos chave-valor*

```
pokemons = [  
    {'nome': 'Bulbassauro'},  
    {'nome': 'Ivyssauro'},  
]
```

In [4]:

```
def nome_segundo_pokemon(pokemons):  
    if len(pokemons) >= 2:  
        if 'nome' in pokemons[1]:  
            return pokemons[1]['nome']  
        else:  
            logger.info("Não encontro o nome dos pokemons!")  
    else:  
        logger.info("Não há pokemons suficientes para procurar!")
```

Esse código funciona?

É longo? é muito difícil de entender?

Segue a PEP8?

```
In [5]: pokedex_vazia = [{}]  
nome_segundo_pokemon(pokedex_vazia)
```

Não há pokemons suficientes para procurar!

```
In [6]: dados_errados = [  
        {'pokemon': 'Bulbassauro'},  
        {'pokemon': 'Ivyssauro'},  
    ]  
nome_segundo_pokemon(dados_errados)
```

Não encontro o nome dos pokemons!

```
In [7]: nome_segundo_pokemon(pokemons)
```

```
Out[7]: 'Ivyssauro'
```

commit, push... profit?

EAFP

It is easier to Ask for Forgiveness than Permission

<https://docs.python.org/3/glossary.html> (<https://docs.python.org/3/glossary.html>).



```
In [8]: def nome_segundo_pokemon(pokemons):  
    # a entrada tem o tamanho que eu espero?  
    if len(pokemons) >= 2:  
        # tem a chave que eu estou procurando?  
        if 'nome' in pokemons[1]:  
            # ufa, então posso acessar o que estou procurando  
            return pokemons[1]['nome']  
  
        # não tem a chave? eita.  
        else:  
            logger.info("Não encontro o nome dos pokemons!")  
  
    # não tem o tamanho correto? vish.  
    else:  
        logger.info("Não há pokemons suficientes para procurar!")
```


“Deve haver um modo melhor!”

-- Raymond Hettinger

```
In [9]: def nome_segundo_pokemon(pokemons):  
        try:  
            return pokemons[1]['nome']  
  
        except IndexError:  
            logger.info('Não há pokemons suficientes para procurar!')  
  
        except KeyError:  
            logger.info('Não encontro o nome dos pokemons!')
```

- O objetivo da função já está na sua segunda linha
- Não existem vários níveis de indentação
- Caso não exista o segundo elemento da lista, ocorre um `IndexError`
- Caso não exista a chave "nome", ocorre um `KeyError`

```
In [10]: pokedex_vazia = [{}]  
         nome_segundo_pokemon(pokedex_vazia)
```

Não há pokemons suficientes para procurar!

```
In [11]: ainda_vazia = [{}, {}]  
         nome_segundo_pokemon(ainda_vazia)
```

Não encontro o nome dos pokemons!

```
In [12]: nome_segundo_pokemon(pokemons)
```

```
Out[12]: 'Ivyssauro'
```

E se eu precisar fazer algo a mais?

```
In [13]: def nome_segundo_pokemon(pokemons):  
    try:  
        logger.info(pokemons[1]['nome'])  
        logger.info('Vá capturar mais!')  
  
    except IndexError:  
        logger.info('Não há pokemons suficientes para procurar!')  
        logger.info('Vá capturar mais!') # de novo?  
  
    except KeyError:  
        logger.info('Não encontro o nome dos pokemons!')  
        logger.info('Vá capturar mais!') # outra vez?
```

Finally!

```
In [14]: def nome_segundo_pokemon(pokemons):  
    try:  
        logger.info(pokemons[1]['nome'])  
  
    except IndexError:  
        logger.info('Não há pokemons suficientes para procurar!')  
  
    except KeyError:  
        logger.info('Não encontro o nome dos pokemons!')  
  
    finally:  
        logger.info('Vá capturar mais!') # só uma vez \o/
```

```
In [15]: nome_segundo_pokemon(pokemons)
```

Ivyssauro
Vá capturar mais!

O bloco finally sempre é executado mesmo que qualquer exceção ocorra

```
In [16]: def log_my_work():  
    timer.start()  
  
    try:  
        lots_of_work()  
  
    except TooMuchWork:  
        logger.info('ops, something went wrong')  
  
    finally:  
        # mesmo que qualquer outra exception ocorra, será executado  
        timer.stop()
```

Não confundir com o bloco "else"

O bloco else é executado após o try, caso não ocorra uma exception.

```
In [17]: def nome_segundo_pokemon(pokemons):  
    try:  
        return pokemons[1]['nome']  
  
    except IndexError:  
        logger.info('Não há pokemons suficientes para procurar!')  
  
    except KeyError:  
        logger.info('Não encontro o nome dos pokemons!')  
  
    else:  
        # não vai executar se não tiver o Ivyssauo  
        logger.info('Vá capturar mais!')
```

```
In [18]: pokedex_vazia = [{}]  
nome_segundo_pokemon(pokedex_vazia)
```

Não há pokemons suficientes para procurar!


```
In [19]: def nome_segundo_pokemon(pokemons):  
    try:  
        logger.info(pokemons[1]['nome'])  
  
    except IndexError:  
        logger.info('Não há pokemons suficientes para procurar!')  
  
    except KeyError:  
        logger.info('Não encontro o nome dos pokemons!')  
    else:  
        logger.info('E tem também o %s!' % pokemons[0]['nome'])
```

```
In [20]: nome_segundo_pokemon(pokemons)
```

Ivyssauro
E tem também o Bulbassauro!

```
In [21]: def nome_segundo_pokemon(pokemons):  
        try:  
            logger.info(pokemons[1]['nome'])  
  
        except IndexError:  
            logger.info('Não há pokemons suficientes para procurar!')  
  
        except KeyError:  
            logger.info('Não encontro o nome dos pokemons!')  
  
        else:  
            # Só quero que execute se o try tiver sucesso  
            logger.info('E tem também o %s!' % pokemons[2]['nome'])
```

```
In [22]: nome_segundo_pokemon(pokemons)
```

Ivyssauro

```
-----  
IndexError                                Traceback (most recent call last)  
<ipython-input-22-c3123dd79692> in <module>()  
----> 1 nome_segundo_pokemon(pokemons)  
  
<ipython-input-21-e93e902b65db> in nome_segundo_pokemon(pokemons)  
    11     else:  
    12         # Só quero que execute se o try tiver sucesso  
----> 13         logger.info('E tem também o %s!' % pokemons[2]['nome'])  
  
IndexError: list index out of range
```

Use o try..else para casos onde gostaria de adicionar no bloco "try" no fim do bloco, mas que não quer que seja tratado pelos blocos except.

Tracebacks

- Trazem o "stack trace" da execução que gerou aquela exceção
- "Que ponto do código gerou aquela exceção, e quais chamadas levaram até aquele ponto"

```
In [23]: def a():
          return b(0)

          def b(z):
              error(z)

          a()
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-23-3d1e3144282a> in <module>()
      5     error(z)
      6
----> 7 a()

<ipython-input-23-3d1e3144282a> in a()
      1 def a():
----> 2     return b(0)
      3
      4 def b(z):
      5     error(z)

<ipython-input-23-3d1e3144282a> in b(z)
      3
      4 def b(z):
----> 5     error(z)
      6
      7 a()

NameError: name 'error' is not defined
```

```
In [24]: try:
          a()
        except:
          exception_info = sys.exc_info()
          logger.info(exception_info)
```

```
(<class 'NameError'>, NameError("name 'error' is not defined",), <traceback ob  
ject at 0x7eff642bbb88>)
```

```
In [25]: try:
          a()
        except:
            exception_info = sys.exc_info()
            logger.info('logando minha exception', exc_info=True)
```

```
logando minha exception
Traceback (most recent call last):
  File "<ipython-input-25-d54910996e5f>", line 2, in <module>
    a()
  File "<ipython-input-23-3d1e3144282a>", line 2, in a
    return b(0)
  File "<ipython-input-23-3d1e3144282a>", line 5, in b
    error(z)
NameError: name 'error' is not defined
```

```
In [26]: try:
          a()
        except:
            exception_info = sys.exc_info()
            logger.exception('logando minha exception')
```

```
logando minha exception
Traceback (most recent call last):
  File "<ipython-input-26-a636fe381183>", line 2, in <module>
    a()
  File "<ipython-input-23-3d1e3144282a>", line 2, in a
    return b(0)
  File "<ipython-input-23-3d1e3144282a>", line 5, in b
    error(z)
NameError: name 'error' is not defined
```

sentry_jira/plugin.py in _get_all_users_for_project at line 242 | application



```
237.
238.     def _get_all_users_for_project(self, client, project):
239.         users = []
240.         for user in client.get_users_for_project(project).json:
241.             users.append({
242.                 'value': user['name'],
243.                 'display': '%s - %s (%s)' % (user['displayName'], user['emailAddress'], user['name']),
244.                 'needsRender': True,
245.                 'q': '',
246.             })
247.         return JsonResponse({'users': users})
```

project

u'UNI'

self

<sentry_jira.plugin.JIRAPLugin object at 0x5883328>

client

<sentry_jira.jira.JIRAClient object at 0x7fa020b2d650>

user

u'errorMessages'

users

[]

Como modelar uma estrutura de Exceptions útil?

Exceptions do Python

BaseException

- +-- SystemExit
- +-- KeyboardInterrupt
- +-- GeneratorExit
- +-- Exception
 - +-- StopIteration
 - +-- StopAsyncIteration
 - +-- ArithmeticError
 - | +-- FloatingPointError
 - | +-- OverflowError
 - | +-- ZeroDivisionError
 - +-- AssertionError
 - +-- AttributeError
 - +-- BufferError
 - +-- EOFError
 - +-- ImportError
 - +-- ModuleNotFoundError
 - +-- LookupError
 - | +-- IndexError
 - | +-- KeyError
 - +-- MemoryError

Exceptions da biblioteca requests

```
RequestException
+-- ConnectionError
|   +-- ProxyError
|   +-- SSLError
|
+-- Timeout
    +-- ConnectTimeout(ConnectionError, Timeout):
    +-- ReadTimeout(Timeout):

+-- RequestsWarning
    +-- FileModeWarning(RequestsWarning, DeprecationWarning):
    +-- RequestsDependencyWarning
```

Capturar uma exception mais genérica pega todas as específicas

```
In [27]: from requests import Timeout

try:
    use_requests()
except Timeout:
    logger.info("catches ConnectionTimeout and ReadTimeout")
```

catches ConnectionTimeout and ReadTimeout

```
In [28]: # pokedex/exceptions.py
class PokedexException(Exception):
    pass

class NotEnoughPokemonError(PokedexException):
    """Não há pokemons suficientes para procurar!"""

class CantFindPokemonError(PokedexException):
    """Não encontro o nome dos pokemons!"""

# pokedex/functions.py

def nome_segundo_pokemon(pokemons):
    try:
        return pokemons[1]['nome']

    except IndexError:
        raise NotEnoughPokemonError

    except KeyError:
        raise CantFindPokemonError
```

```
In [29]: pokedex_vazia = [{}]  
nome_segundo_pokemon(pokedex_vazia)
```

```
-----  
IndexError                                Traceback (most recent call last)  
<ipython-input-28-d1ab70dda423> in nome_segundo_pokemon(pokemons)  
    14     try:  
----> 15         return pokemons[1]['nome']  
    16
```

IndexError: list index out of range

During handling of the above exception, another exception occurred:

```
NotEnoughPokemonError                    Traceback (most recent call last)  
<ipython-input-29-70c6b1a2745f> in <module>()  
      1 pokedex_vazia = [{}]  
----> 2 nome_segundo_pokemon(pokedex_vazia)  
  
<ipython-input-28-d1ab70dda423> in nome_segundo_pokemon(pokemons)  
    16  
    17     except IndexError:  
----> 18         raise NotEnoughPokemonError  
    19  
    20     except KeyError:
```

NotEnoughPokemonError:

```
In [30]: pokedex_vazia = [{}]  
try:  
    nome_segundo_pokemon(pokedex_vazia)  
except PokedexException as exc:  
    logger.info('Something went wrong')
```

Something went wrong

E se nós não nos importarmos com os erros que ocorrerem?

```
In [31]: from contextlib import suppress  
  
with suppress(RequestException):  
    use_requests()  
  
logger.info('continue...')  
  
continue...
```



```
In [34]: try:
          pokemons()
        except: # Gotta catch 'em all!
          pass
```

@daneoshiga

Obrigado!

Perguntas?

- <https://bit.ly/olist-tech> (<https://bit.ly/olist-tech>) (newsletter)
- https://t.me/olist_learning (https://t.me/olist_learning) (@olist_learning)