

Live de Python #39

Padronização de código e guias de estilo (PEPs)

Roteiro

- Guias de estilo
- PEP-8
- PEP-3107
- PEP-257

Guias de estilo

PEP - 8

<https://www.python.org/dev/peps/pep-0008/>

<http://pep8.org/>

PEP-8 Imports

Os imports em python tem quatro regras simples e bem definidas:

1. Não devemos importar dois módulos na mesma linha
2. A ordem dos imports importa, SIM
3. Não se deve importar usando wildcard (*)
4. Imports não devem substituir módulos builtin

PEP-8 Imports

Os imports em python tem quatro regras simples e bem definidas:

1. **Não devemos importar dois módulos na mesma linha**
2. A ordem dos imports importa, SIM
3. Não se deve importar usando wildcard
4. Imports não devem substituir módulos builtin

```
# Errado  
  
import os, sys  
  
# certo  
  
import os  
  
import sys
```

PEP-8 Imports

Os imports em python tem quatro regras simples e bem definidas:

1. Não devemos importar dois módulos na mesma linha
2. **A ordem dos imports importa, SIM**
 - a. Primeiro o futuro (`__future__`)
 - b. Depois a bibliotecas padrões
 - c. Bibliotecas de terceiros
 - d. Suas bibliotecas
3. Não se deve importar usando wildcard
4. Imports não devem substituir módulos builtin

PEP-8 Imports

Os imports em python tem quatro regras simples e bem definidas:

1. Não devemos importar dois módulos na mesma linha

2. A ordem dos imports importa, SIM

- a. Primeiro o futuro (`__future__`)
- b. Depois a bibliotecas padrões
- c. Bibliotecas de terceiros
- d. Suas bibliotecas

3. Não se deve `import __future__` # *proximas implementações*

4. Imports não `import os` # *biblioteca padrão*

```
import requests # biblioteca de terceiros
```

```
import minha_lib # bibliotecas proprias
```


PEP-8 Imports (espaços)

```
# proximas implementações
import __future__

# biblioteca padrão
import os
import sys

# biblioteca de terceiros
import requests
import funcy

# bibliotecas proprias
import minha_lib
import minha_outra_lib
```

PEP-8 Imports

Os imports em python tem quatro regras simples e bem definidas:

1. Não devemos importar dois módulos na mesma linha
2. A ordem dos imports importa, SIM
3. **Não se deve importar usando wildcard (*)**
4. Imports não devem substituir módulos builtin

```
from re import *  
  
findall(r'\w+', 'eduardo foi a escola')  
# ['eduardo', 'foi', 'a', 'escola']
```

PEP-8 Imports

Os imports em python tem quatro regras simples e bem definidas:

1. Não devemos importar dois módulos na mesma linha
2. A ordem dos imports importa, SIM
3. **Não se deve importar usando wildcard (*)**
4. Imports não devem substituir módulos builtin

```
from re import findall
```

```
findall(r'\w+', 'eduardo foi a escola')  
# ['eduardo', 'foi', 'a', 'escola']
```

```
from re import *
```

```
findall(r'\w+', 'eduardo foi a escola')  
# ['eduardo', 'foi', 'a', 'escola']
```



PEP-8 Imports

Os imports em python tem quatro regras simples e bem definidas:


1. Não devemos importar dois módulos na mesma linha
2. A ordem dos imports importa, SIM
3. Não se deve importar usando wildcard (*)
4. **Imports não devem substituir módulos builtin**

```
from re import *  
from re import compile  
compile() # Default ou import?
```

PEP-8 Imports

Os imports em python tem quatro regras simples e bem definidas:

1. Não devemos importar dois módulos na mesma linha
2. A ordem dos imports importa, SIM
3. Não se deve importar usando wildcard (*)
4. **Imports não devem substituir módulos builtin**



```
from re import compile
from re import compile
compile()
```

```
import re
from re import compile as re_compile
re_compile()
re.compile()
```

PEP - 8 Tamanho das linhas

Toda linha de código deve ter no máximo 80 caracteres. Segundo a PEP, alguns times usam 100 caracteres, porém o número de caracteres em comentários será mantido em 72 caracteres (Docstrings e comentários)

Agora vamos voltar aos imports com from.

```
# ERRADO  
from re import compile, findall, finditer, fullmatch, match, escape, replace, subn
```

PEP - 8 Indentação

Todo código deve usar espaços no lugar de TABs. Todo bloco deve ser definido com 4 espaços. *Caso você use TAB, não usar TAB + ESPAÇOS*

PEP - 8 Indentação - agrupamento

Existem 6 maneiras diferentes de agrupar blocos de código (Isso vale para listas, tuplas, dicionários, etc ...)

```
# CORRETO 0 - Grade
from re import (compile, findall, finditer, fullmatch,
                match, escape, replace, subn)
```


PEP - 8 Indentação - agrupamento

Existem 6 maneiras diferentes de agrupar blocos de código (Isso vale para listas, tuplas, dicionários, etc ...)

```
# CORRETO 1 - Vertical
from re import (compile,
                findall,
                finditer,
                fullmatch,
                match,
                escape,
                replace,
                subn)
```

PEP - 8 Indentação - agrupamento

Existem 6 maneiras diferentes de agrupar blocos de código (Isso vale para listas, tuplas, dicionários, etc ...)

```
# CORRETO 2 - alinhamento deslocado  
from re import \br/>    compile, findall, finditer, fullmatch, \br/>    match, escape, replace, subn
```

PEP - 8 Indentação - agrupamento

Existem 6 maneiras diferentes de agrupar blocos de código (Isso vale para listas, tuplas, dicionários, etc ...)

```
# CORRETO 3 - alinhamento deslocado na vertical
from re import (
    compile,
    findall,
    finditer,
    fullmatch,
    match,
    escape,
    replace,
    subn
)
```

PEP - 8 Indentação - agrupamento

Existem 6 maneiras diferentes de agrupar blocos de código (Isso vale para listas, tuplas, dicionários, etc ...)

```
# CORRETO 4 - Grade alinhada
from re import (
    compile, findall, finditer, fullmatch,
    match, escape, replace, subn)
```

PEP - 8 Indentação - agrupamento

Existem 6 maneiras diferentes de agrupar blocos de código (Isso vale para listas, tuplas, dicionários, etc ...)

```
# CORRETO 5 - Recuo de grade alinhada  
from re import (  
    compile, findall, finditer, fullmatch,  
    match, escape, replace, subn  
)
```

PEP - 8 Indentação - agrupamento

Existem 6 maneiras diferentes de agrupar blocos de código (Isso vale para listas, tuplas, dicionários, etc ...)

```
# Aligned with opening delimiter.
foo = long_function_name(var_one, var_two,
..... var_three, var_four)

# More indentation included to distinguish this from the rest.
def long_function_name(
..... var_one, var_two, var_three,
..... var_four):
... print(var_one)

# Hanging indents should add a level.
foo = long_function_name(
... var_one, var_two,
... var_three, var_four)
```

PEP - 8 Indentação - agrupamento

Existem 6 maneiras diferentes de agrupar blocos de código (Isso vale para listas, tuplas, dicionários, etc ...)

```
my_list = [  
    1, 2, 3,  
    4, 5, 6,  
    ]  
result = some_function_that_takes_arguments(  
    'a', 'b', 'c',  
    'd', 'e', 'f',  
    )
```

PEP - 8 Espaços em branco

Evite espaços em branco quando:

1. Dentro de parênteses, chaves, ...
2. Entre uma vírgula antes de fechar o parêntese
3. Antes de vírgulas, ponto e vírgula e dois pontos
4. Ao menos que o que venha antes seja a abertura de um slice
5. Antes de parênteses
6. Em parâmetros nomeados

PEP - 8 Espaços em branco

Evite espaços em branco quando:

1. Dentro de parênteses, chaves, ...
2. Entre uma vírgula antes de fechar o parêntese
3. Antes de vírgulas, ponto e vírgula e dois pontos
4. Ao menos que o que venha antes seja a abertura de um slice
5. Antes de parênteses
6. Em parâmetros nomeados

CERTO

```
spam(ham[1], {eggs: 2})
```

ERRADO

```
spam( ham[ 1 ], { eggs: 2 } )
```

PEP - 8 Variáveis

1. Constantes devem ter letras maiúsculas
2. Variáveis nunca devem ser 're-declaradas'

PEP - 3107

<https://www.python.org/dev/peps/pep-3107/>

PEP - 3107 Anotações de funções

1. Anotações de funções são completamente opcionais
2. As anotações de funções não são mais do que uma forma de associar expressões arbitrárias de Python a várias partes de uma função em tempo de compilação.

```
# CERTO 3107
def xpto(spam: str='eggs', *args: list) -> int:
    return None
```

PEP - 257

<https://www.python.org/dev/peps/pep-0257/>