Python Naming Convention & PEP8

Naming Convention

Modules: use lowercase letters and underscores

    Math_operations.py

Variables: Globals variable should be in uppercase with underscores

    GLOBAL_VARIABLE = 10

locals variable should follow the same convention as functions

    local_variable = 5

Classes: CapWords (or CamelCase) convention. This means that the first letter of each word in the class name should be capitalized, and there should be no underscores between words.

    class Car

Exceptions: names should end with "Error," following the CapWords convention.

    except CustomError as ce:

PEP 8

Ref: peps.python.org/pep-0008/

Indentation: Use 4 spaces per indentation level.

```
# Correct:

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

# Add 4 spaces (an extra level of indentation) to distinguish arguments 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)
```

```
# Wrong:

# Arguments on first line forbidden when not using vertical alignment.
foo = long_function_name(var_one, var_two,
    var_three, var_four)

# Further indentation required as indentation is not distinguishable.
def long_function_name(
    var_one, var_two, var_three,
    var_four):
    print(var_one)
```

The 4-space rule is optional for continuation lines.

Optional:

```
# Hanging indents *may* be indented to other than 4 spaces.
foo = long_function_name(
  var_one, var_two,
  var_three, var_four)
```

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

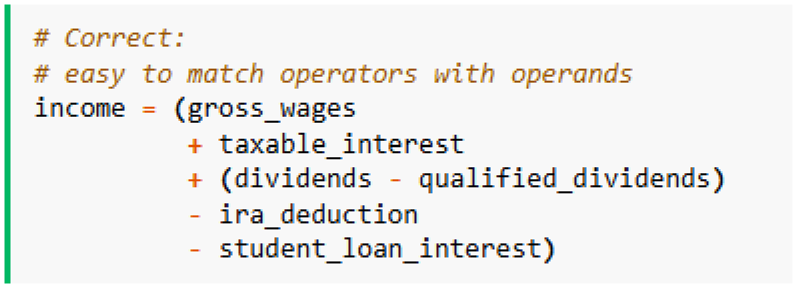
or it may be lined up under the first character of the line that starts the multil S

<u>Tabs or Spaces?</u>

Spaces are the preferred indentation method.
Tabs should be used solely to remain consistent with code that is already indented with tabs.

Maximum Line Length: The Python standard library is conservative and requires limiting lines to 79 characters (and docstrings/comments to 72).

Line break before binary operator.



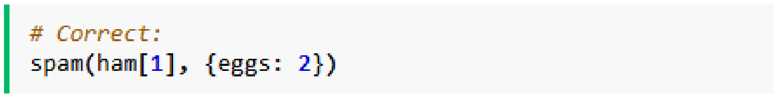
```
# Correct:
# easy to match operators with operands
income = (gross_wages
          + taxable_interest
          + (dividends - qualified_dividends)
          - ira_deduction
          - student_loan_interest)
```

Blank lines:
Surround top-level function and class definitions with two blank lines.
Method definitions inside a class are surrounded by a single blank line.

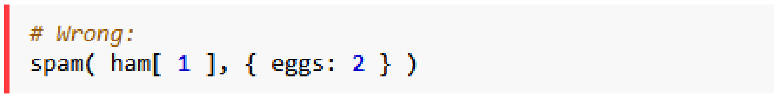
Import:
Imports should be grouped in the following order:

1. Standard library imports.
2. Related third party imports.
3. Local application/library specific imports.

Whitespace:

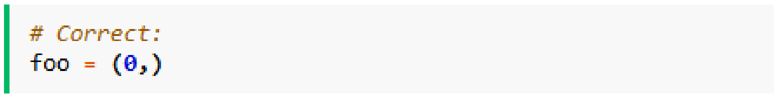


```
# Correct:
spam(ham[1], {eggs: 2})
```

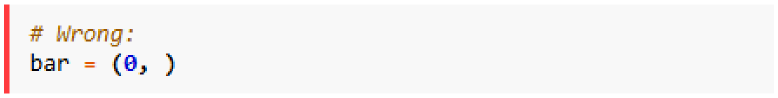


```
# Wrong:
spam( ham[ 1 ], { eggs: 2 } )
```

Between a trailing comma and a following close parenthesis:



```
# Correct:
foo = (0,)
```



```
# Wrong:
bar = (0, )
```

- Immediately before a comma, semicolon, or colon:

```
# Correct:
if x == 4: print(x, y); x, y = y, x
```

```
# Wrong:
if x == 4 : print(x , y) ; x , y = y , x
```

.

```
# Correct:
ham[1:9], ham[1:9:3], ham[:9:3], ham[1::3], ham[1:9:]
ham[lower:upper], ham[lower:upper:], ham[lower::step]
ham[lower+offset : upper+offset]
ham[: upper_fn(x) : step_fn(x)], ham[:: step_fn(x)]
ham[lower + offset : upper + offset]
```

```
# Wrong:
ham[lower + offset:upper + offset]
ham[1: 9], ham[1 :9], ham[1:9 :3]
ham[lower : : step]
ham[ : upper]
```

.

```
# Correct:
spam(1)
```

```
# Wrong:
spam (1)
```

Immediately before the open parentl

```
# Correct:
dct['key'] = lst[index]
```

```
# Wrong:
dct ['key'] = lst [index]
```

```
# Correct:
x = 1
y = 2
long_variable = 3
```

```
# Wrong:
x             = 1
y             = 2
long_variable = 3
```

```python
# Correct:
i = i + 1
submitted += 1
x = x*2 - 1
hypot2 = x*x + y*y
c = (a+b) * (a-b)
```

```python
# Wrong:
i=i+1
submitted +=1
x = x * 2 - 1
hypot2 = x * x + y * y
c = (a + b) * (a - b)
```

```python
# Correct:
def munge(input: AnyStr): ...
def munge() -> PosInt: ...
```

```python
# Wrong:
def munge(input:AnyStr): ...
def munge()->PosInt: ...
```

Don't use spaces around the `=` sign when used to i
a default value for an *unannotated* function param

```python
# Correct:
def complex(real, imag=0.0):
    return magic(r=real, i=imag)
```

```python
# Wrong:
def complex(real, imag = 0.0):
    return magic(r = real, i = imag)
```

it does not make sense to have a trailing comma on the same line

```
# Correct:
FILES = [
    'setup.cfg',
    'tox.ini',
    ]
initialize(FILES,
           error=True,
           )
```

```
# Wrong:
FILES = ['setup.cfg', 'tox.ini',]
initialize(FILES, error=True,)
```

.

Inline comments should be separated by at least two spaces from the statement.

- PEP 257 describes good docstring conventions. Note that most importantly, the `"""` that ends a multiline docstring should be on a line by itself:

  ```
  """Return a foobang

  Optional plotz says to frobnicate the bizbaz first.
  """
  ```

- For one liner docstrings, please keep the closing `"""` on the same line:

  ```
  """Return an ex-parrot."""
  ```