Python PEP8 Style Guide

Table of contents

Variable and function names 1
Spacing 1
Breaking long computation 2
Comments 2

PEP stands for **Python Enhancement Proposal**. A PEP is a design document providing information on how to **style** our code. This course follows **Google's PEP**, which is a variant of **PEP8**. Give it a browse.

Variable and function names

Names **must start with a letter** and must not include special characters excepting the underscore `_`.

Function and variable names should be lowercase, with words separated by underscores as necessary to improve readability:

Yes	No
`descriptive_variable_name`	`DescriptiveVariableName`

Spacing

Put single spaces around the *loosest-binding* operators in an expression and close up the tighter ones, so the spacing mirrors the precedence. A bracketed subexpression is treated as an atom whose interior is closed up.

Yes	No
`i = i + 1`	`i=i+1`
`x = x*2 - 1`	`x = x * 2 - 1`
`hypot2 = x*x + y*y`	`hypot2 = x * x + y * y`
`c = (a+b) * (a-b)`	`c = (a + b) * (a - b)`

Breaking long computation

According to PEP8 all lines must be **strictly less than 80 characters** wide.

```
>>> income = (gross_wages
...           + taxable_interest
...           + (dividends - qualified_dividends)
...           - ira_deduction
...           - student_loan_interest)
```

The **redundant bracket that encloses the entire expression** is what enables the expression to be broken across multiple lines — usually, hitting enter would evaluate the expression instead.

Comments

Anything following a `#` will be ignored by Python. In addition to docstrings (see python-functions[1]) you can — and *should* — include comments in your code to explain something not obvious in your design.

Not helpful:

```
x = x + 1                     # Increment x
```

Helpful:

```
x = x + 1                     # Compensate for border
```

The first comment restates what the code already says. The second says *why*.

[1] courses/csse1001/python-functions.pdf