Python Style Rules

Introduction

Programming style deals with the appearance of your program. Documentation consists of explanatory remarks and comments for the program. Good programming style and appropriate documentation reduce the chance of errors and make programs easy to read.

Appropriate Comments and Comment Styles

Comments should be complete sentences. If a comment is a phrase or sentence, its first word should be capitalized, unless it is an identifier that begins with a lower case letter (never alter the case of identifiers!).

If a comment is short, the period at the end can be omitted. Block comments generally consist of one or more paragraphs built out of complete sentences, and each sentence should end in a period.

Header Comments

Header comments appear at the top of a file. These lines typically include the filename, author, date, version number, and a description of what the file is for and what it contains. For class assignments, headers should also include such things as course name, number, section, and assignment number.

Example:

```
'''
 * Program: Assignment name
 * Programmer: Your name
 * Due: 7/3/2016
 * Description: (Give a brief description for Assignment)
'''
# Display three messages
print("Welcome to Python")
print("Python is fun")
print("Problem Driven")
```

Inline Comments

Use inline comments sparingly.

An inline comment is a comment on the same line as a statement. Inline comments should be separated by at least two spaces from the statement. They should start with a # and a single space.

Inline comments are unnecessary and in fact distracting if they state the obvious. Don't do this:

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

But sometimes, this is useful:

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

Naming Conventions

Make sure that the meanings of the descriptive names you choose for variables, constants, classes, and methods are straightforward. Names are case-sensitive. Listed below are the conventions for naming variables, functions, and classes.

- For variables and functions, always use lowercase. If the name consists of several words, concatenate them into one, making the first word lowercase and capitalizing the first letter of each subsequent word in the name.

```
student_count = 0          # Number of students in class
fahrenheit_float = 0.0     # Temperature in Fahrenheit
```

- For class names, capitalize the first letter of each word in the name.

```
class SalariedEmployee
 class HourlyEmployee
```

- Use singulars for variables representing single items such as student and count. Use plurals for arrays or collections.
- You can never have a space within a name.
  eg.

```
sumOfSquares
printHappyBirthday
totalApples
```

Constants

All letters in constants should be capitalized, and underscores should be used between words.

```
HEAT_OF_FUSION = 79.71
```

Proper Indentation and Spacing

Use **4 spaces** per indentation level.

A consistent indentation style makes programs clear and easy to read. Indentation is used to illustrate structural relationships among the program's components or statements. You should indent each subcomponent or statement ***four*** spaces more than the structure within which it is nested.

Use a space to separate parameters in a function. Do not leave spaces before or after parentheses in a method. For example, aMethod(a1, a2) is preferred, whereas aMethod (a1, a2) is not a good style.

A single space should be added on both sides of a binary operator, as shown in the following statement:

```
b = 3 + 4 * 4 > 5 * (4 + 3) - i
```

A single space line should be used to separate segments of the code to make the program easier to read.

Tabs or Spaces?

Never mix tabs and spaces.

The most popular way of indenting Python is with spaces only. By default, IDLE will convert new tabs to spaces

Semicolons

Do not terminate your lines with semi-colons and do not use semi-colons to put two commands on the same line.

Parentheses

Use parentheses sparingly.
Do not use them in return statements or conditional statements unless using parentheses for implied line continuation.

It is however fine to use parentheses around tuples.

```
if foo:
    bar()
while x:
    x = bar()
if x and y:
    bar()
if not x:
    bar()
return foo
for (x, y) in dict.items(): ...
```

Maximum Line Length

Source code lines will be no more than 80 characters long. Break long lines into multiple shorter lines using the continuation character ("\").

Only one statement will be included on a given line of the source code.

Blank lines and spaces will be used to enhance the readability of both source statements and comments.

Blank Lines

Separate top-level function and class definitions with two blank lines.
Method definitions inside a class are separated by a single blank line.
Use blank lines in functions, sparingly, to indicate logical sections.

Imports

All import statements will appear after the source code header and before any other Python code in the program.

Imports should usually be on separate lines, e.g.:

Yes:
```
import os
import sys
```

No:
```
import sys, os
```