CS 160

Python Programming Style Guideline

Comments

The comments should describe what is happening, what parameters and variables mean, any restrictions or bugs, etc. Avoid comments that are clear from the code. Don't write comments that disagree with the code. Short comments should be *what* comments, such as "compute mean value", rather than *how* comments such as "sum of values divided by n". Putting a comment at the top of a 3-10 line section telling what it does overall is often more useful than a comment on each line describing micrologic.

Programs should include a program header at the very top of the file, and EVERY function should be preceded by headers blocks, which describe the purpose, usage, and type of any parameters, as well as any pre-conditions and post-conditions. For those unclear on these concepts, a pre-condition is a condition that must hold prior to beginning the function, and post-conditions are conditions that must hold upon exiting the function. As code is updated, these may change, and must be updated accordingly.

```
##############################################################################
# Program Filename:
# Author:
# Date:
# Description:
# Input:
# Output:
##############################################################################


##############################################################################
# Function:
# Description:
# Parameters:
# Return values:
# Pre-Conditions:
# Post-Conditions:
##############################################################################
```

Comments that describe algorithms, data structures, etc. should be in block comment form.

```
# Here is a block comment.
# The comment text should be tabbed or spaced over uniformly.
# The pound symbol followed by a space is used for commenting a block…

# Or for commenting a line or in line with code.
x = x + 1     # Increment x
```

Block comments inside a function are appropriate, and they should be spaced over to the same indentation setting as the code that they describe. One-line comments alone on a line should be indented to the same space setting of the code that follows.

```
import math

# Read radius value from user
radius = float(input("Enter the circle’s radius value: "))

#Print area of circle to user
print("Area of your circle is: ", 2 * math.pi * radius, "\n")
```

Very short comments may appear on the same line as the code they describe, and should be spaced over to separate them from the statements.

```
if a == EXCEPTION:
    b = TRUE        # special case
else:
    b = isprime(a)  # works only for odd a
```

Whitespace

Use vertical and horizontal whitespace generously. Indentation and spacing should reflect the block structure of the code; e.g., there should be at least 2 blank lines between the end of one function and the comments for the next. For indentation, you can use tabs or spaces, but BE CONSISTENT. If you choose to use spaces, then I suggest you use at least 3 spaces for indenting because it is hard to see 1 or 2 spaces.

It is a good idea to have spaces after commas in argument/variable lists to help separate the arguments/variables and statements visually.

```
print(length, width)

def calculate_rectangle_area(length, width):

for i in range(0, MAX_SIZE):
```

Operators should be surrounded by a space. For example, use

```
z = x + y
```

rather than

```
z=x+y
```

This greatly enhances readability, and makes it significantly easier to spot operators within an expression. Prefix and postfix increment and decrement are not considered operators in this context.

Compound Statements

A compound statement is a list of statements followed by a colon. Regarding spacing, be consistent with your local standard, if you have one, or pick one and use it consistently. When editing someone else's code, always use the style used in that code.

```
control:
          statement
          statement
```

Naming Conventions

Individual projects will no doubt have their own naming conventions. There are some general rules however.

- Names with leading and trailing underscores are reserved for special methods in classes. Function and variable names, as well as method and object names, should be in lower case, with words separated by an underscore.
- Avoid names that differ only in case, like foo and FOO. Similarly, avoid foobar and foo_bar. The potential for confusion is considerable. Similarly, avoid names that look like each other. On many terminals and printers, 'l', '1' and 'I' look quite similar. A variable named 'l' is particularly bad because it looks so much like the constant '1'.
- Variables used as constants should be in all CAPS.

Constants

Even though Python doesn't give you way to create a constant, you may want to create a variable at the top of your program that you use as a constant. Numerical constants should not be coded directly. Symbolic constants make the code easier to read. Defining the value in one place at the top also makes it easier to administer large programs, since the constant value can be changed uniformly by changing only the assignment statement.

- There are some cases where the constants 0 and 1 may appear as themselves instead of as a constant defined. For example if a for loop indexes through an array:

  ```
  for i in range(0, ARYBOUND):
  ```

- Even simple boolean values like 1 or 0 are often better expressed using defines like TRUE and FALSE (sometimes YES and NO read better).

Line Length

Lines should be 80 characters in length or shorter (this includes the newline character). Try not to have your code wrap on a terminal because it lowers readability.