

## Style Guidelines for 15-110

Your programs are graded on more than just their input-output behavior. It is not enough to have programs that happen to work: they need to clearly state what they do, have some empirical evidence that they work as advertised, and be easy for other people to read and reason about.

**You may lose style points on assignments unless you follow these style guidelines:**

- Use informative names for variables.
- Use comments when the code is not straightforward.
- Use spaces between operators and assignments to improve readability (e.g. `minutes = hours * 60` instead of `minutes=hours*60`).
- Keep imports at the beginning of the file (you only need to import libraries once).
- Use empty lines in functions only if they improve readability. One empty line is enough.
- If you repeat a sequence of commands in a function, check if the code can be reorganized so that this sequence occurs once. For example:

GOOD PRACTICE	BAD PRACTICE
<pre>def getDigit(num, pos):     if (num &lt; 0):         num = num * -1     x = 10 ** pos     y = 10 ** (pos - 1)     mn = (num % x) // y     return mn</pre>	<pre>def getDigit(num, pos):     if (num &lt; 0):         num = num * -1         x = 10 ** pos         y = 10 ** (pos - 1)         mn = (num % x) // y         return mn     else:         x = 10 ** pos         y = 10 ** (pos - 1)         mn = (num % x) // y     return mn</pre>

- Distinguished cases that are not special should not be distinguished.  
For example:

GOOD PRACTICE	BAD PRACTICE
<pre>def add(m,n):     return m + n</pre>	<pre>def add(m,n):     if m == 0:         return n     else:         return m + n</pre>

Keep in mind that the first version of a code is rarely the best one. Once you have a working program, look at it again and try to make it more concise and elegant.