Programming Style

General Advice

- Never hard-code literals. Not even pi. Assign to variables.
  - You'll thank me the first time you have to change some fundamental constant and only have to do it once.
- Try to do at least basic sanity checks on your input.
- Never expect a compiler or interpreter to read your mind. If you intend something to be a float, declare it as such (if required by your language) or, for constants, put the period at the end. Use explicit casts just to be explicit.
- Don't try to be fancier than necessary. Just because you *can* do something doesn't mean you *have* to do it.

Keep it Neat

- Indent properly. Code at the same level should have the same indentation. Indent each level a fixed number of spaces (3 is usually good). Don't use tabs.
- Never have repeated blocks of code. Write a function/ subroutine/procedure.
- Put a short documentation block at the top of every function/subroutine/procedure.
- My personal taste: don't document excessively. Only document lines that aren't obvious to a casual reader (who knows the language). It's distracting to have too many comments.
  - I often actually write pseudocode without commenting it out, that way it will throw a syntax error if I don't finish so I know that it's incomplete.

Make Code Self-Documenting

- Choose good variable names. It's not necessary to make them extremely long, but they should be self-explanatory. Learn to type if you are hunting and pecking!
- Similarly, choose good function/subroutine/ procedure names. Don't use e.g. "compute" as the name of your most important routine. Compute what? It must do *something* more specific.
- Once again, INDENT PROPERLY!

Code Organization

- Put some thought into selecting your fundamental data structures. Often a good structure/class layout is the difference between a readable, maintainable code and a "spaghetti"code.
- Use whatever packaging method is available in your language. Keep your "packages" as short and self---contained as possible. USE SEPARATE FILES for each package/module even if not required by the language.

Guidelines for Style

- From "Coding Guidelines: Finding the Art in the Science" by Robert Green and Henry Ledgard, *Communications of the ACM* **54**, p. 57.
  - Use fixed-width fonts
  - Variable names should be nouns or noun phrases
  - Functions that act like variables should also be nouns, or if Boolean/logical like adjectives
  - Procedures/subroutines should be verbs or verb phrases
  - Try to make names that are pronounceable

Guidelines (Continued)

- Routine variables such as array or loop indices should be short (i,j,k and so forth) and may be reused. Less commonly used variables, or variables corresponding to mathematical quantities, should have a longer, descriptive name. Use normal English ordering, e.g. *SetName* rather than *NameSet*.
- Use spaces to show the layout. Align horizontally and vertically.
- Focus on code, not comments (i.e. don't overcomment)
- http://www.eng.utoledo.edu/eecs/faculty_web/hledgard/softe/upload/index.php?&direction=0&order=&directory=Miscellaneous
  - File is Professional Guidelines-2009.pdf
  - Examples and a few of the guidelines are specific to C++, but most of the advice is of general applicability.

Example (Python Syntax)

- Not so good:

```
import math
"""This script computes the surface area of a sphere of radius r.
   The value of r is input by the user. """”
#Enter radius
r = float(raw_input("Enter the radius r:"))
#Compute area
A = 4*math.pi*r**2
print "r= %10.3f A=%10.3e" % (r,A)
```

Better

```
import math
"""This script computes the surface area of a
sphere of a specified radius.  The value of the radius
is input by the user.
Author: Jim Dandy
Date:  January 2012 """

radius = float(raw_input("Enter the radius:"))
surface_area = 4.*math.pi*radius**2
print "radius= %10.3f Surface Area=%10.3e" % \
                                    (radius,surface_area)
```