Previous ✓ Next

2.1.6 Coding style guide

Bookmark this page

The following is a short set of tips we highly recommend you adopt in writing computer code, in particular for computer codes used in CSE applications. As we get into the course, we will introduce some other best practices in particular with respect to object-oriented programming.

1. **Do not hardcode parameter values.**

   For example, to compute the force of gravity for a 100 kg weight using $f = mg$ you should not write

   ```
   f = 100*9.81
   ```

   Instead, it is preferable to write

   ```
   g = 9.81 # [m/s^2]
   m = 100 # [kg]
   f = m*g
   ```

   This makes the code more comprehensible where there are complicated equations. Moreover this practice allows for better code re-use when parameters need to be changed.

2. **Use indicative and meaningful variable names.**

   Sometimes in CSE it is ok to use single letter variables provided the context and connection to the mathematical equations are obvious beyond all doubt. For example, in the example above it was ok to name the variable `f` rather than `force` since it was clear we were implementing the example $f = mg$. This is a judgement call. Typically if you have to ask whether a variable name is obvious then you should probably just use a more indicative variable name.

3. **Sometimes it is better not to simplify equations!**

   Although it is tempting to do so, the minute computational speedups gained from simplifying equations usually do not outweigh the loss in code comprehensibility, or code extensibility. At best you are saving $O(1)$ flops.

   For example, if we had to compute the ratio of areas between two circles, we prefer

   ```
   # not as good
   ratio = (r1/r2)**2

   # good
   ```

Discussions

All posts sorted by recent activity

Rounding error In subsection 7, I believe the computed
tolgayilmaz
☆ 2

```
area1 = math.pi*r1**2
area2 = math.pi*r2**2
ratio = area1/area2
```

Why? Perhaps because later on we might be interested in other geometric objects, in which case code-modification in the latter case is easy since:

```
area1 = Circle.area()
area2 = Rectangle.area()
ratio = area1/area2
```

4. **Code defensively against expensive function evaluations.**

   Throughout this class we will often refer to some function `f` that relates to our physical system. You should always have the mindset that calls to `f` are expensive, in memory and in wall time. For instance, if `f` corresponds to a numerical weather simulation then it could take days to compute a single call.

   Try to cache previous computations whenever possible, though within reason – learning what is reasonable is a judgement call you will learn to g

< Previous Next >

   For example, suppose we had to compute the squared error between a prediction coming form a

© All Rights Reserved

edX

[About](#)
[Affiliates](#)
[edX for Business](#)
[Open edX](#)
[Careers](#)
[News](#)

Legal

[Terms of Service & Honor Code](#)
[Privacy Policy](#)
[Accessibility Policy](#)
[Trademark Policy](#)
[Sitemap](#)
[Cookie Policy](#)
Your Privacy Choices

Connect

[Idea Hub](#)

Contact Us

Help Center

Security

Media Kit

64-bit floating point numbers.

```
# good!
```