Coding Style
Correctness is not sufficient

- In mathematics, for example, it is simply important to get the right answer. Elegance is nice, but not necessarily required.
  - In programming, this is not the case.
  - Writing software is part science, part craft!
- Would you turn in an essay that had poor formatting, no paragraphs, incorrect spacing, etc?
  - The ‘style’ of your code is actually *qualitative*. 
Programming languages are for humans

- The reason high-level programming languages were invented was to make working with computers easier for humans.

- Not only when writing, but, more importantly when reading!
  - Code is “write-once, read-many”

- “Always code as if the person who ends up maintaining your code is a violent psychopath who knows where you live.”
  - Jeff Atwood “Coding Horror”

http://www.codinghorror.com/blog/2008/06/coding-for-violent-psychopaths.html
Good style…

- Makes you a better programmer
- Reduces bugs in your code
- Reduces debugging time
- Reduces cognitive overhead
- Improves your grades
- Earns the love and adoration of your collaborators
What is ‘good style’?

- What makes good style is somewhat subjective and is difficult to define in a complete manner.

- The following are usually considered as part of style:

  - consistent layout & indentation
  - informative, consistent naming
  - useful comments
  - sensible structure
Layout & indentation

Compare:

```java
if (hours < 24 && minutes < 60 && seconds < 60) {
    return true;
} else {
    return false;
}
```

or

```java
if (hours < 24 && minutes < 60 && seconds < 60)
{
    return true;
}
else
{
    return false;
}
```

with something like

```java
if ( hours < 24
    && minutes < 60
    && seconds < 60
)
{return     true
;}
else
{return     false
;}
```
Variable & method naming

- Name things **descriptively**!

- The reader (grader) should be able to see a reference to a variable or method and immediately know what it represents!

- Bad names
  - int tmp
  - getInfo()
  - String str2

- Good names
  - int clickCounter;
  - calculateAvg()
  - String lastName;
Comments

- What are good comments?
  - Don't write 'what' code is doing. Write 'why'.
  - Get to the point. Don't write your life story.
  - Alert readers to important code blocks.

- When do you not need comments?
  - When your code is self-documenting!
    - It is so elegant and clean that its obvious what it does.
Sensible structure

- Methods (especially main) should not be too long. They should “do one thing and do it well”.

- Rule of thumb:
  - If a method exceeds 50 lines of code your terror alert status should be at ‘elevated’.
  - As it passes 100, it should raise to ‘high’.
  - Past 200 the terror alert is ‘severe’.

- As the number of lines grows, the probability that you are conflating responsibilities of what should be separate methods does too.
Conclusion

- Code should be easy to read and neat.
- Code should be consistently formatted.
- Variables and methods should be named thoughtfully.
- Code should be self-documenting.
- Comments should explain 'why' not 'what'.
- Code should be organized into methods intelligently.