Why is good coding style useful and important?

“Structure and document your program the way you wish other programmers would.” ("Toward Developing Good Programming Style", McCann) This simple phrase is telling on many levels. Up to this point, you may not have thought of anyone besides yourself while writing code, but in reality you should be writing it so others not only understand it, but can do so quickly. There is much collaboration in the world of programming, so anything that can be done to help coders understand each other’s work is a plus. If you reach out to someone to help fix your code, but they first have to spend an extreme amount of time figuring out what your program is trying to do, time is wasted. Some of the things you will read about you may already do, and some may give you ideas to better write future programs.

There are numerous different ways that one can go about having good and readable code.

1. Program organization - When you write your program you want to group similar aspects of the code together.
   a. If your code does more than one thing gather all of the component pieces of each process together so that readers, users, and you don’t have to go hunting for certain details that will make or break your program.
   b. Another important aspect of organization is to delete obsolete and old pieces of code that are no longer necessary to run the whole program.

2. Effective variable names - When you name your variables there are certain things to consider to make your information easy to read
   a. Names should reflect the purpose or values stored within them
   b. They should follow a naming convention
      i. thisIsEasyToRead
      ii. so-is-this
      iii. and_also_this
   c. Choose the one that is the easiest for you to type and read
   d. Stay consistent
      i. doNotDo-this_or-thisOrThis

3. Indent and use blank space
a. Indents should be consistent length
b. They should be used to keep conditionals and loops organized
c. If you are using nested loops you may want to standardize where the loops are so that you don’t end up with a very limited work space.
   i. Please
      1. Don’t
         a. Make
            i. Your
               1. Code
                  a. Do
                     i. This because there will be no room for all of your code and you will be stuck out here for line after line and I hope this proves my point

4. Comments
   a. Always include a header comment before your code that includes dates written and a basic overview of the purpose of your code. Be sure to note any errors/bugs that exist in your code as well.
   b. Make comments in moderation - people who will read your code will have at a minimum a basic understanding of coding.
      i. You won’t need to explain every line
   c. When you comment make sure that it shows the higher levels of your program
      i. Make comments for chunks of code or entire functions
      ii. Use them to explain your purpose, internal thoughts, or summarize large blocks of code.

```
29  void bounce(){ //changes direction when contact screen edge
30  if(x<0 || x>width){
31     xspeed=-xspeed;
32 }
```

d. If there will be a large number of comments explaining the purpose of your functions an external readme file may be beneficial rather than bogging your program down with comments.

5. Other useful tips
a. Use variables to store one way flags - these don’t change back ever
   i. If a condition is met it stays met for the remainder of the program
b. Use counters
   i. Counters will let you keep track of individual pieces. Refresh or create a new counter as necessary
c. Don’t make your conditional expect the Else statement
   i. There are times where there are a few specific cases where we get a wanted outcome and then using the Else condition to catch all of the other values is acceptable.
   ii. If you find yourself with a single condition that causes your Else statement to happen, use that as your primary condition instead.