I believe the best coding standards to follow at all times for an object-oriented program are the SOLID principles. The first SOLID principle, is the S - single responsibility principle. This principle defines that any class or function should only be responsible to accomplish one goal. A class should not do more than one thing otherwise code become less clear and harder to handle if changes are needed. This is a vital part of OOP.

The second principle is the O - open-closed principle. This principle defines that an Object should be able to be extended, extended meaning to be used or used in new circumstances, without having to be modified. This is usually achieved by making Objects more generic and putting function which cant be generic in the Object itself.

L - Liskov substitution principle, this principle says that when a child object extends a parent object and the child object is called it should be possible to replace the child objects call with the parent one and not receive any sort of error.

I - Interface Segregation. This means that a user or client should not be forced to use or depend on a method they do not use. This makes the creation and use of an Interface considerably easier as if a user does not want to use a certain method they don't need to write useless code.

D - Dependency Inversion. Dependency inversion is making code depend on abstractions rather than concretions. This means that rather than making a function, class or whatever depend on an exact type/kind it should depend on a an abstraction. For example rather than making my function depend on my Banana object I will make it simply depend on an Object.

Some other practices/standards I strongly believe in:

- If you don't need it now, don’t write it.

- DRY: Don’t repeat yourself EVER