**Ninja irony**: Make the code as short as possible. Show how smart you are.

**Meaning**: do not sacrifice code clarity for brevity.

use single-letter variable names everywhere.

**Meaning**: use a meaning word when naming variables and functions. That makes the

code easy to follow and understand. But on some positions is better to use one

letter, like using i in the for loop.

Use abbreviations.

Meaning: abbreviation words will make it very hard to understanding what is the variable using for or the structure of the program.

When choosing a name, try to use words describe the data type (str, num, var, chara or data).

**Meaning**: like these names do not contains information that help to understand what

Is the variable do.

Using same name in different position as much as you can

**Meaning:** do not use similar name for different variables that make your code hard to understand.

Using same common prefix to unusual structure, like printName for showing a name on a window.

**Meaning**: try to follow the common prefix of the team in the same position that what they expected.

Add a new variable only when necessary. Instead, reuse existing names. Just write new values into them. In a function try to use only variables passed as parameters.

**Meaning**: That would make it really hard to identify what’s exactly in the variable now. And, where it comes from. The purpose is to develop the intuition and memory of a person reading the code. A person with weak intuition would have to analyze the code line-by-line and track the changes through every code branch.

Put underscores before variable names without meaning.

**Meaning**: underscore should be in a particular place with specific meaning. Usually, the team agree to use it for specific variable. For example, sometime using underscore at the begging of each local object.

Using unmeaning name to declaration (super, mega, …)

**Meaning**: the name must be related to the using of the variable, it should describe the using of the variable.

Overlap outer variables.

**Meaning**: avoid overlap a global variable with a local variable.

Side-effects everywhere!

**Meaning**: each function should have only one task, do not make any secondary mission.