Coding Conventions

Due to this being a collaborative software project among new developers, it's important to make sure that the code you write is clean and usable by others. We predominantly use two languages in this project, C++ and Python. Below are a list of coding conventions for each respective language in the project. There may be exceptions to these rules, but otherwise they should be followed. A code linter will be created to ensure that code conforms to there standards.

roslint

To check if you code passes standard checks, compile using:

```
rsmake roslint
```

This will invoke the compiler with the code linter turned on. The C++ linter will run first and will fail with an error if the linter fails and the Python linter will not be run. If the C++ code passes the linter then the Python linter will be run.

All Languages

Indentation

Spaces shall be used for indentation, no tabs.
Rational: mixing tabs and spaces can cause major bugs in Python, and can create ugly formatted code in all other languages, so consistency is important. Although I prefer all tabs in Python, spaces makes more sense for other languages.

Line Length

The maximum length of a line shall be 80 characters
Rational: this results in code that is readable on most screens without wrapping lines

Trailing Spaces

Trailing whitespace at the end of a line is not allowed.
Rational: most text editors will actually automatically chop the trailing whitespace when you open a file. If people aren't paying attention, this can result in them commiting a file where they made no functional changes other than to cut the whitespace, resulting in a confusing source control history. In addition, it's just a cleanliness thing.

Extra Newlines

More than one blank line in a row is not allowed.
Rational: If you need to break up your code into logical chunks it is better to use a single newline and a comment describing the next code block.

C++

running

```
astyle -A1 -N -n <file name>
```

on your C++ files should clean them up nicely.

Python

Nodes should be defined in classes with the constructor called in the "main" section.

```
def Node():
    def __init__():
        # define pubs and subs here
    ...

if __name__=='__main__':
    rospy.init_node()
    Node()
    rospy.spin()
```