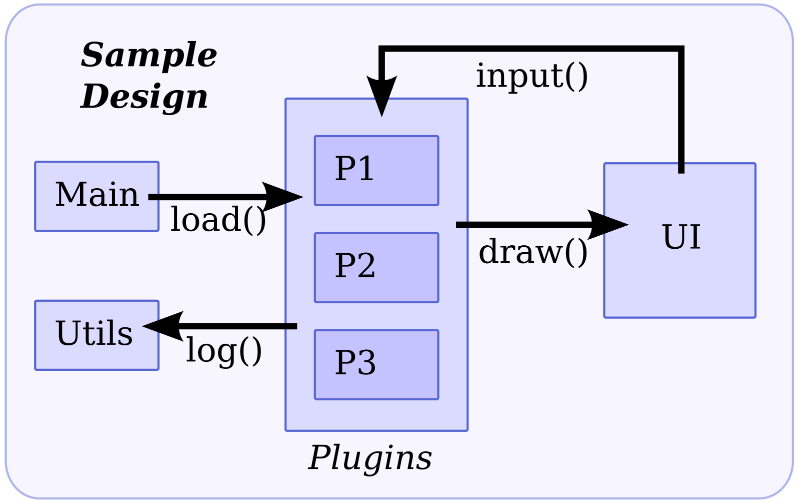
# Templates

Software design groups related code, such as plugins or widgets, so that large system can be reasoned about.



**Interfaces** and **classes** only capture part of this software design. They tell us that Plugins have load and input methods and UI has a draw method.

The fact that plugins should call the draw() method or that UI calls the plugin's input() is lost during coding.

We would also like all plugins to be implemented consistently.

Like interfaces, **templates** list the methods that an object provide, but they also help with syntactic formatting and provide information about the behavior of the object.

## Consistent Formatting

Provide easy creation and syncronization of code

```
# Generic plugin
template Plugin:

    # Load Plugin
    def load():

    # Handle UI Input
    def input():
```

```
# Plugin Number One
class P1 from Plugin:

    # Handle Output
    def input():
        process()
        ...
    # Load Plugin
    def load():
```

Inconsistent comment

Flag out of order defs

## Behavioral information

```
template Plugin:
    def load():
        # Add widgets
        calls UI.add

        # Initial draw
        calls UI.draw
```

```
# Main class
def main():
    # Add and draw
    # all widgets
    new P1().load()
    new P2().load()
    new P3().load()
```

Static checks for method existance and also method behavior