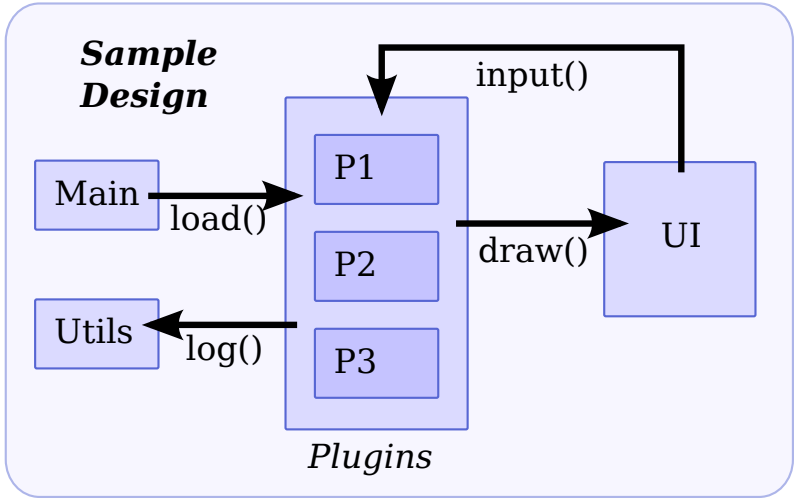
Templates

The software design should let us reason about groups of related code, such as plugins, drivers, widgets, etc



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 plugins input() is lost during coding.

We would also like all plugins to be somewhat consistently implemented.

Templates act as super interfaces by providing more information about the code that uses the template.

Syntactic templates

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():
```

Flag bad comment

Flag out of order defs

Semantic templates

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

      # Initial draw
      calls UI.draw()
```

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

Static checks for method existance and also method behavior