world

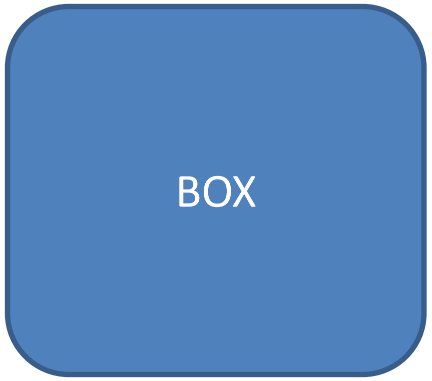
Taking over the ~~lab~~ world with Python

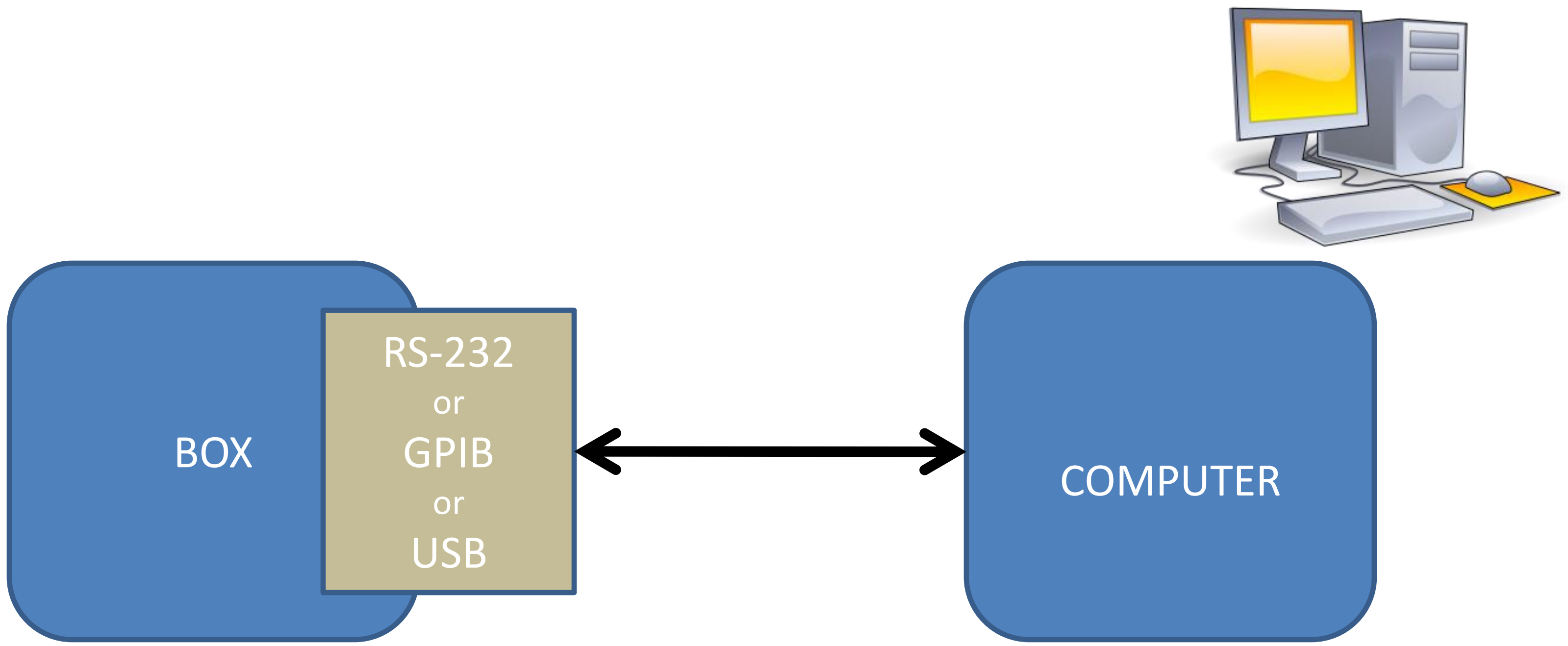
or

I want to replace LabView

Justin Lazear
GSFC Python Bootcamp
6/13/2014

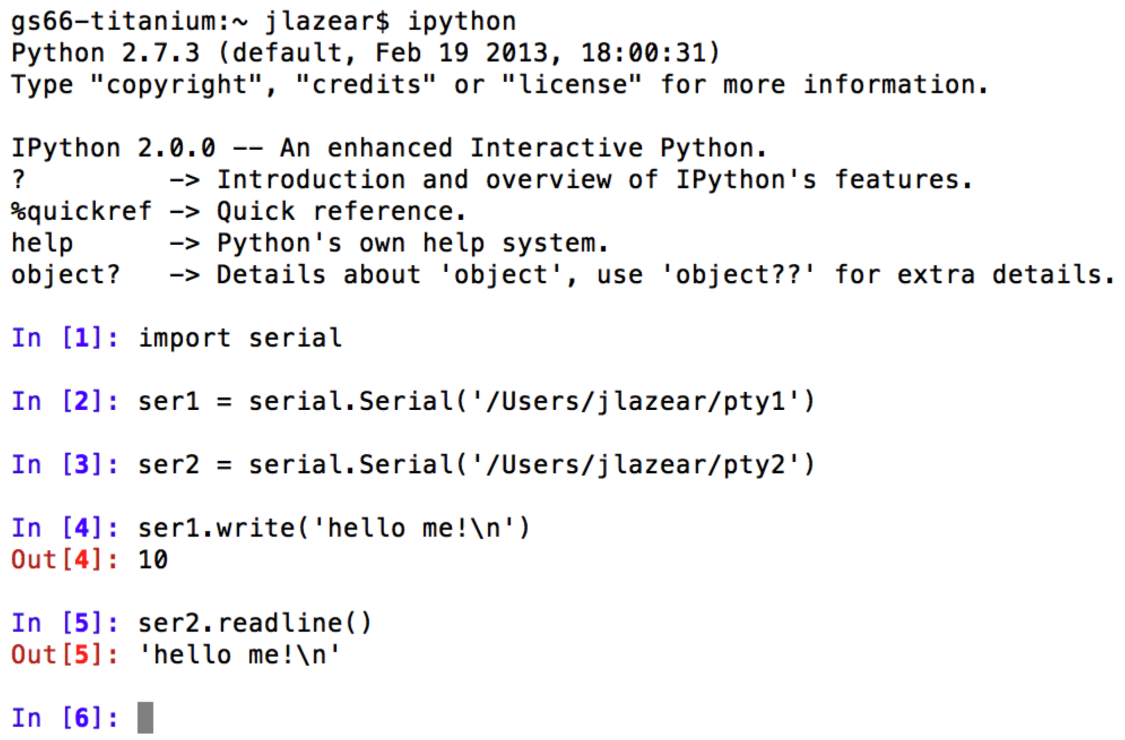
=


BOX

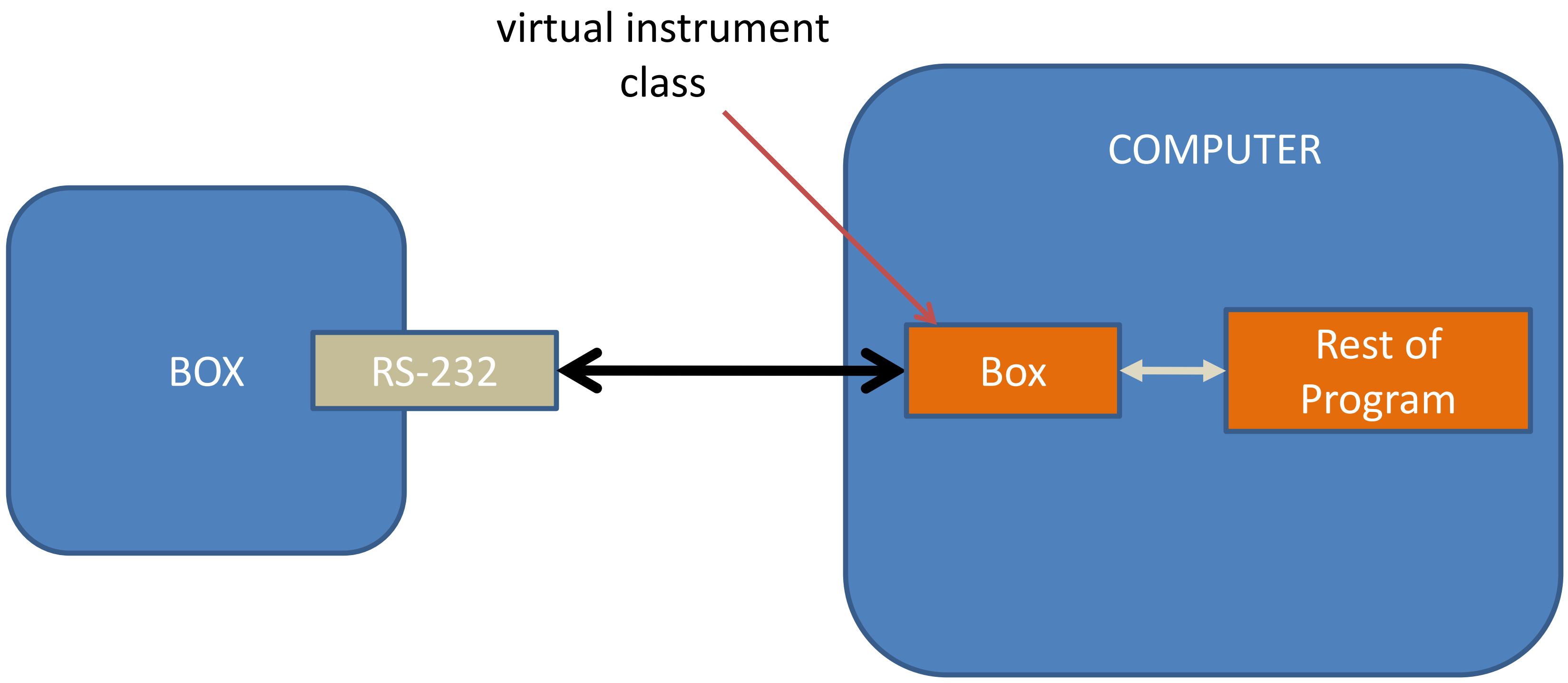


RS-232 $\rightarrow$	pyserial (http://pyserial.sourceforge.net/)
GPIB $\rightarrow$	pyVISA (http://pyvisa.readthedocs.org/en/latest/)
USB $\rightarrow$	pyUSB (http://sourceforge.net/apps/trac/pyusb/)

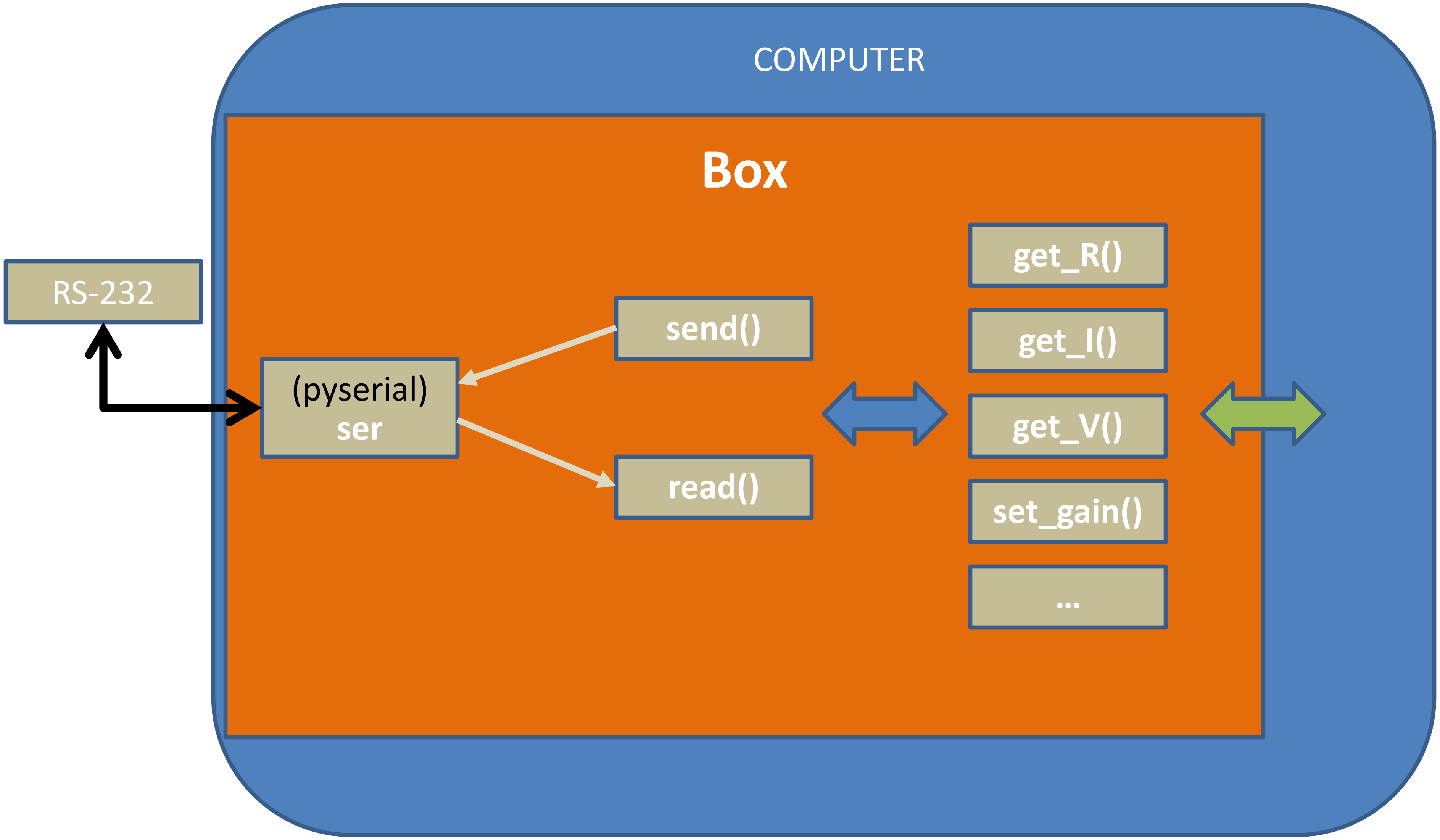
pyserial is easy

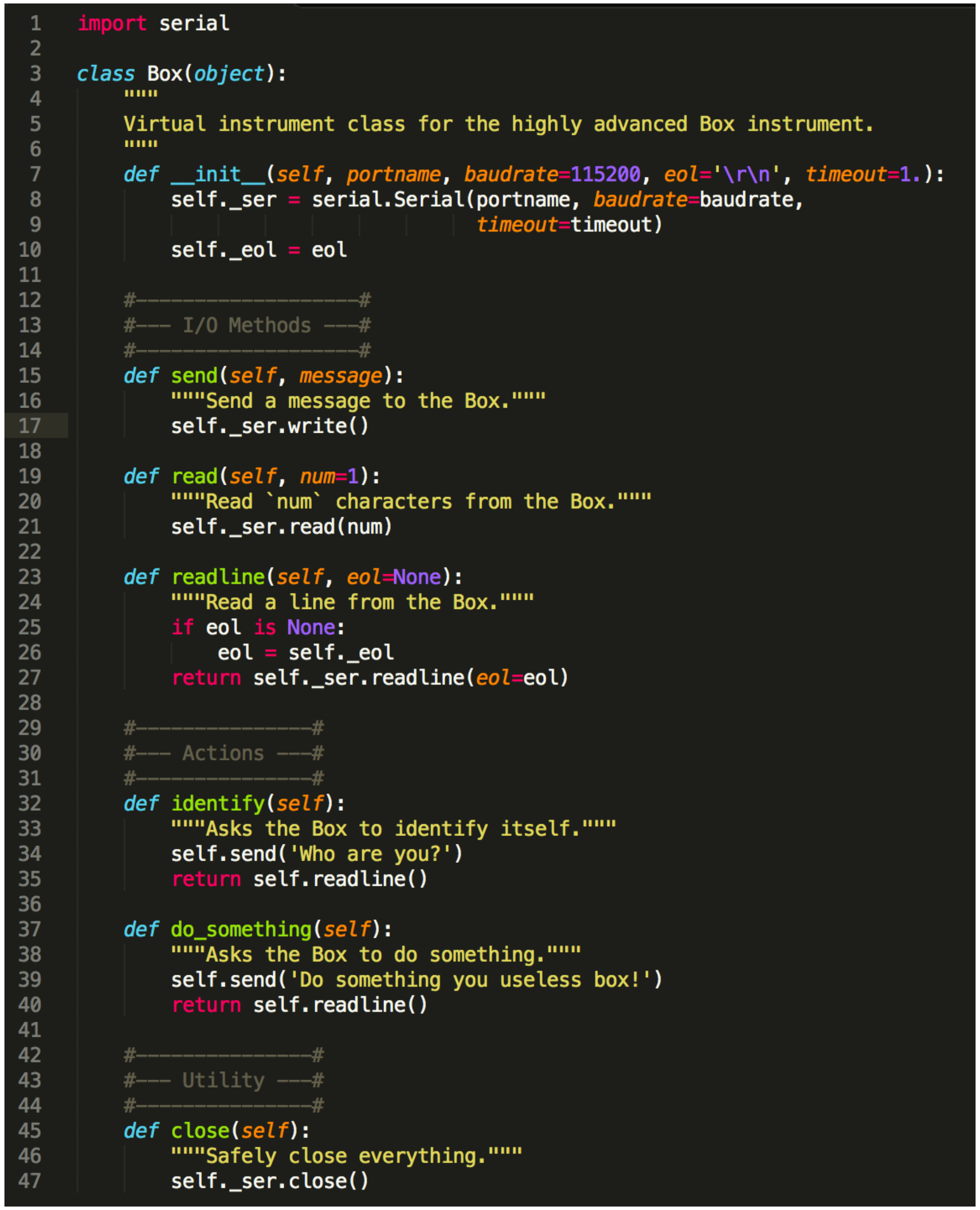


```
gs66-titanium:~ jlazear$ ipython
Python 2.7.3 (default, Feb 19 2013, 18:00:31)
Type "copyright", "credits" or "license" for more information.

IPython 2.0.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import serial

In [2]: ser1 = serial.Serial('/Users/jlazear/pty1')

In [3]: ser2 = serial.Serial('/Users/jlazear/pty2')

In [4]: ser1.write('hello me!\n')
Out[4]: 10

In [5]: ser2.readline()
Out[5]: 'hello me!\n'

In [6]: 
```

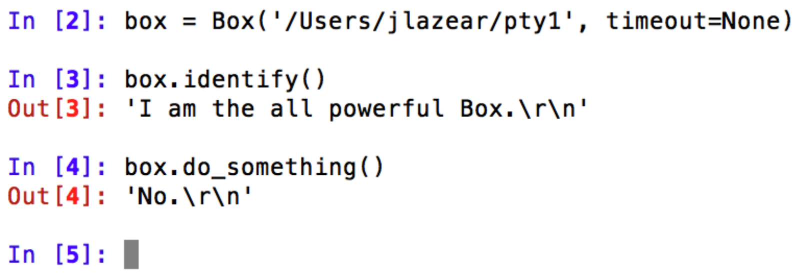


Box now looks like it's in the computer


COMPUTER
Box
RS-232
(pyserial)
ser
send()
read()
get_R()
get_I()
get_V()
set_gain()
...

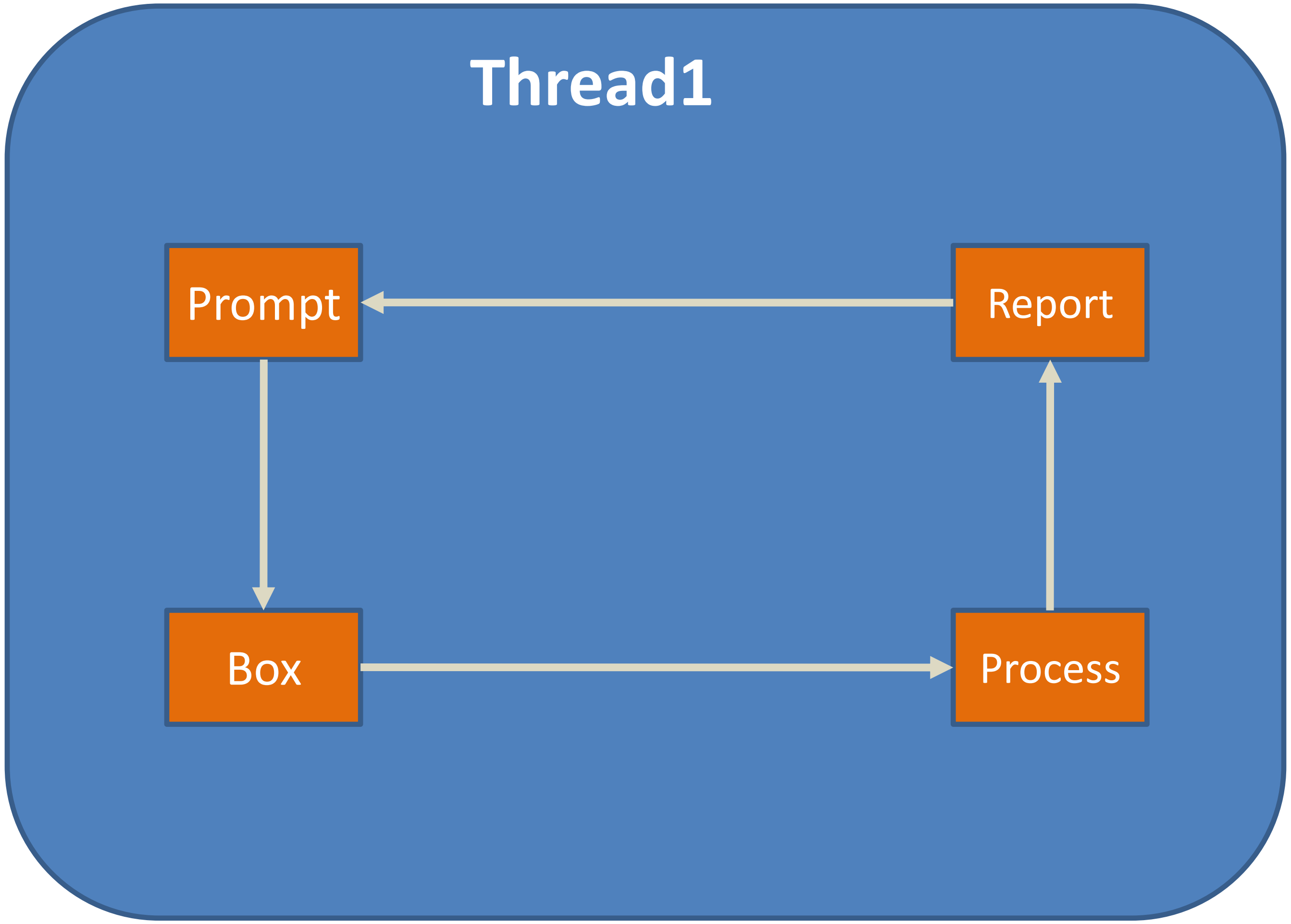


```python
import serial

class Box(object):
    """
    Virtual instrument class for the highly advanced Box instrument.
    """
    def __init__(self, portname, baudrate=115200, eol='\r\n', timeout=1.):
        self._ser = serial.Serial(portname, baudrate=baudrate,
                                  timeout=timeout)
        self._eol = eol

    #--------------------#
    #---- I/O Methods ---#
    #--------------------#
    def send(self, message):
        """Send a message to the Box."""
        self._ser.write()

    def read(self, num=1):
        """Read `num` characters from the Box."""
        self._ser.read(num)

    def readline(self, eol=None):
        """Read a line from the Box."""
        if eol is None:
            eol = self._eol
        return self._ser.readline(eol=eol)

    #----------------#
    #---- Actions ---#
    #----------------#
    def identify(self):
        """Asks the Box to identify itself."""
        self.send('Who are you?')
        return self.readline()

    def do_something(self):
        """Asks the Box to do something."""
        self.send('Do something you useless box!')
        return self.readline()

    #----------------#
    #---- Utility ---#
    #----------------#
    def close(self):
        """Safely close everything."""
        self._ser.close()
```

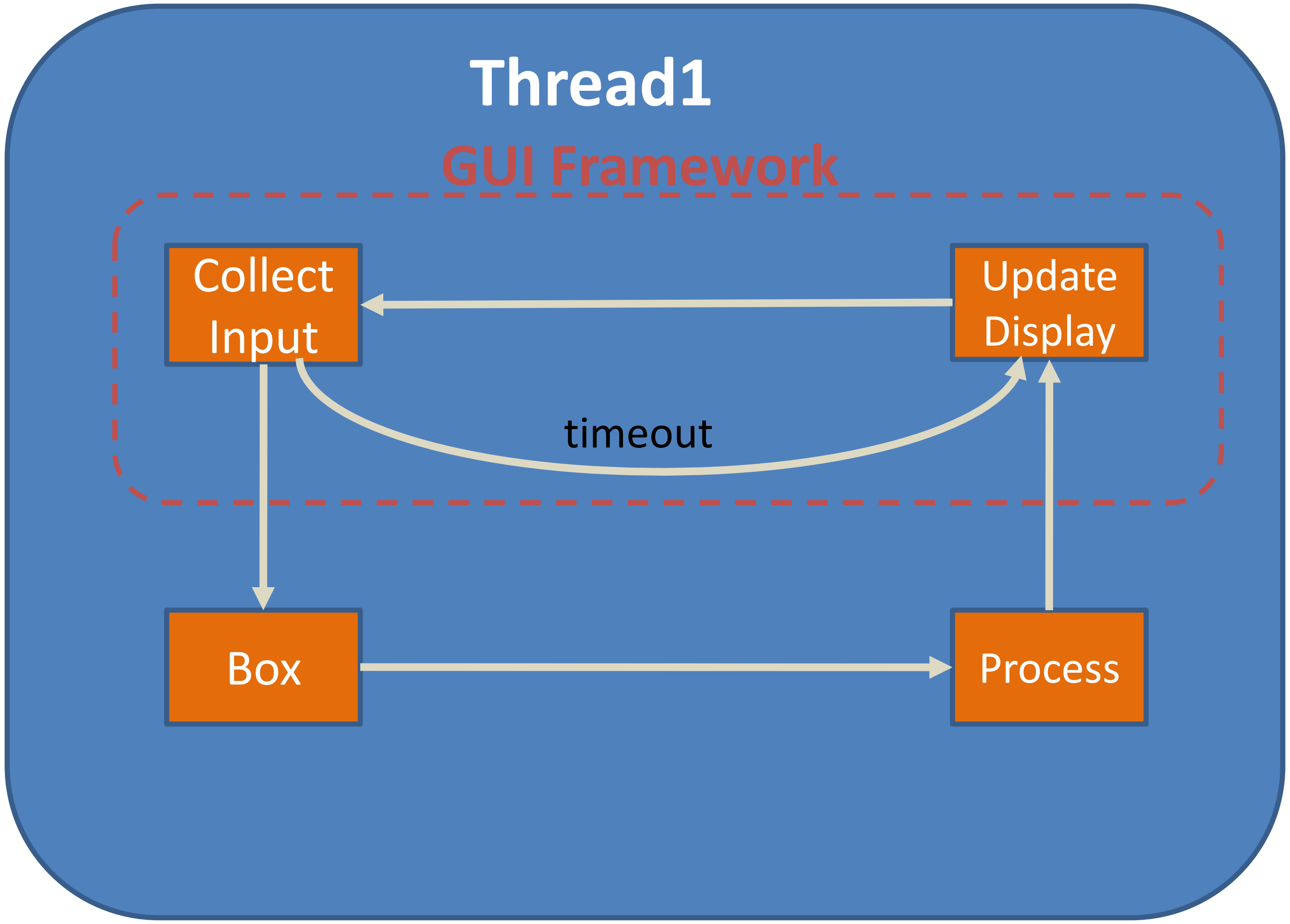


```
In [2]: box = Box('/Users/jlazear/pty1', timeout=None)

In [3]: box.identify()
Out[3]: 'I am the all powerful Box.\r\n'

In [4]: box.do_something()
Out[4]: 'No.\r\n'

In [5]: 
```

Always in the computer now…



Default structure of GUIs



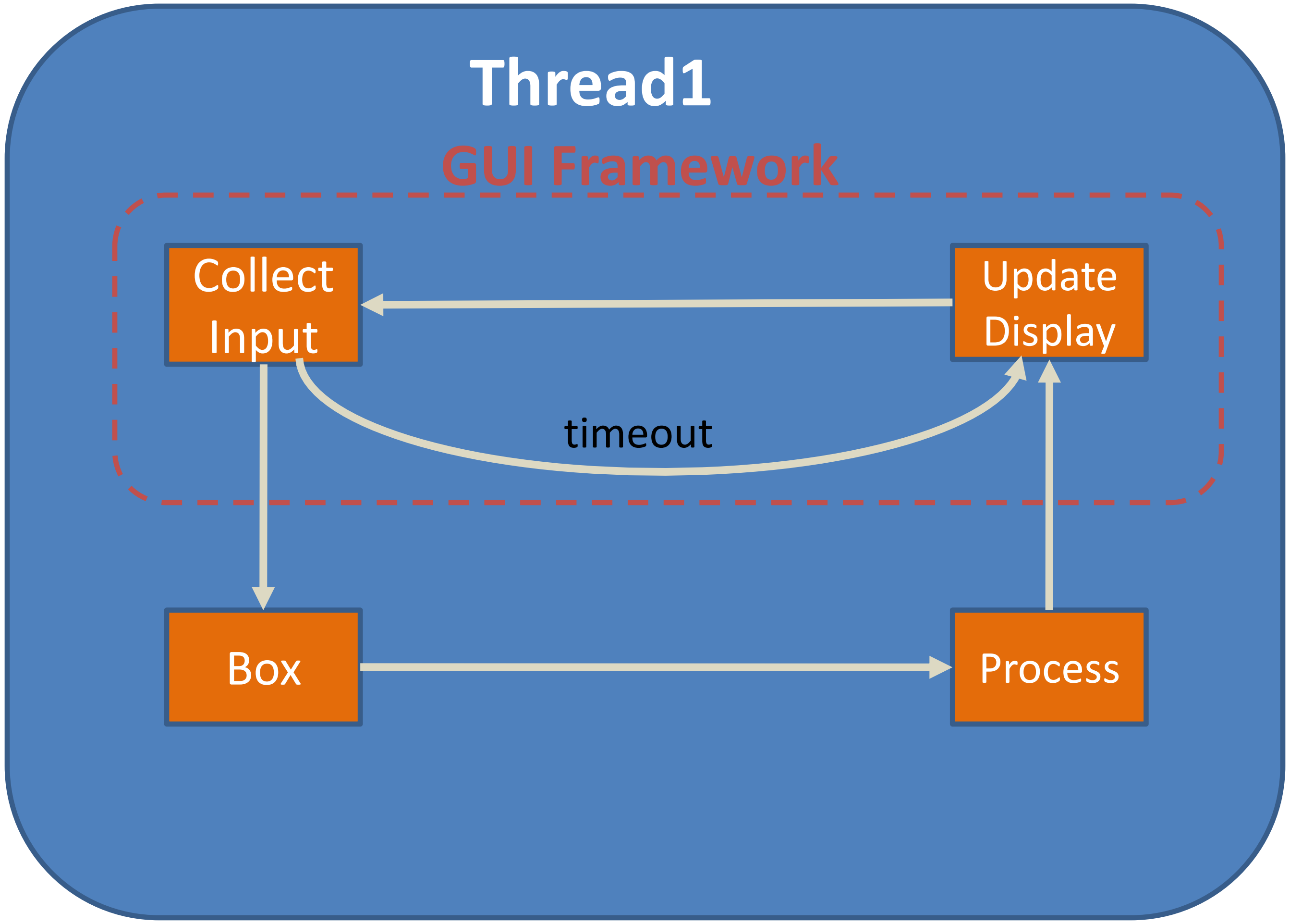
GUI Frameworks

- **PyQT/PySide (QT)**
  - https://wiki.python.org/moin/PyQt
- **wxPython/Project Phoenix (wxWidgets)**
  - http://www.wxpython.org/
  - http://wiki.wxpython.org/ProjectPhoenix
- **Tkinter (Tk/Tcl)**
  - https://wiki.python.org/moin/TkInter
- Others…
  - Kivy, PyGame, Traits/TraitsUI, …
  - https://wiki.python.org/moin/GuiProgramming

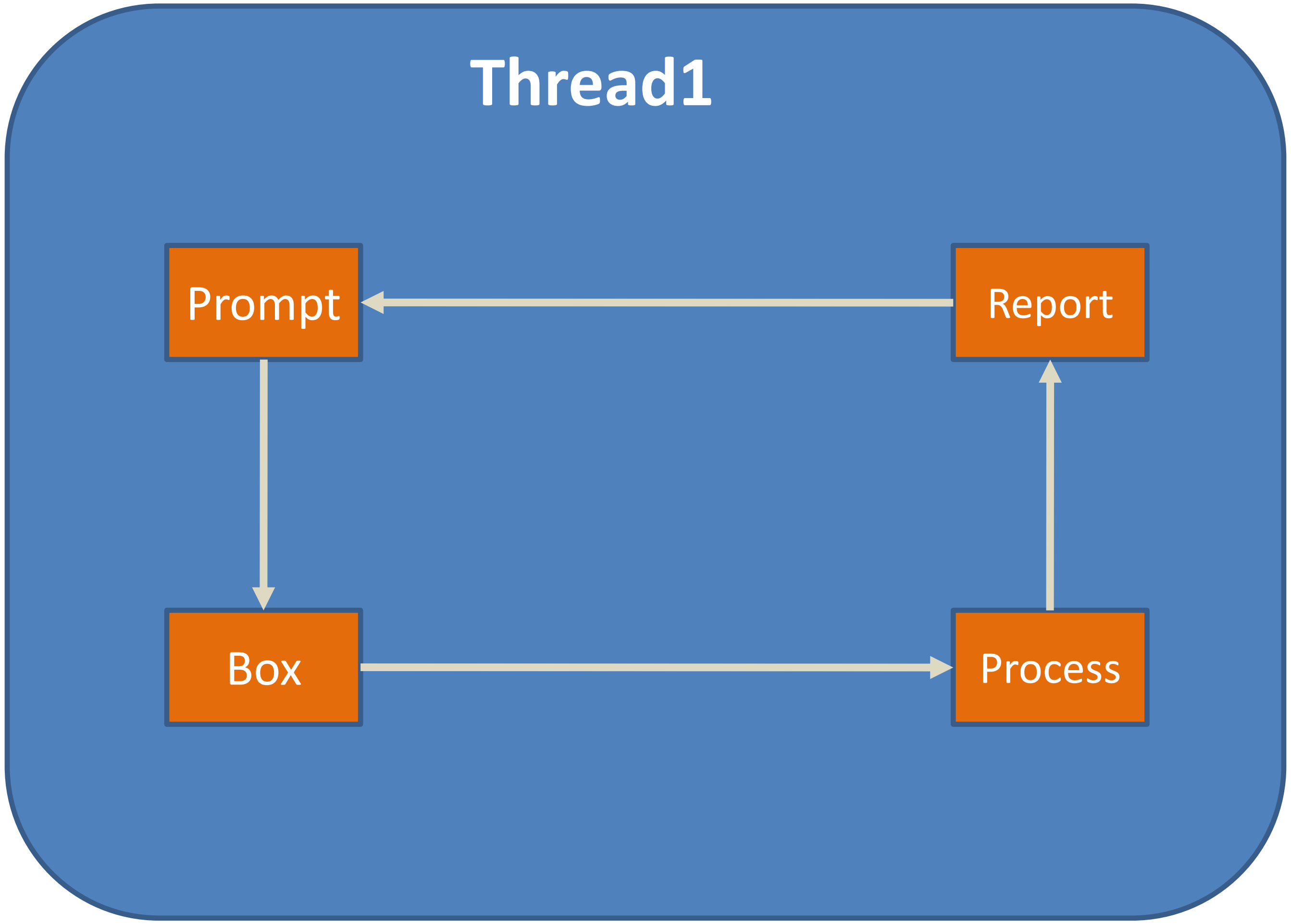
Almost entirely preference-based.

All use the same design patterns and structures.

Good enough for basic lab use!

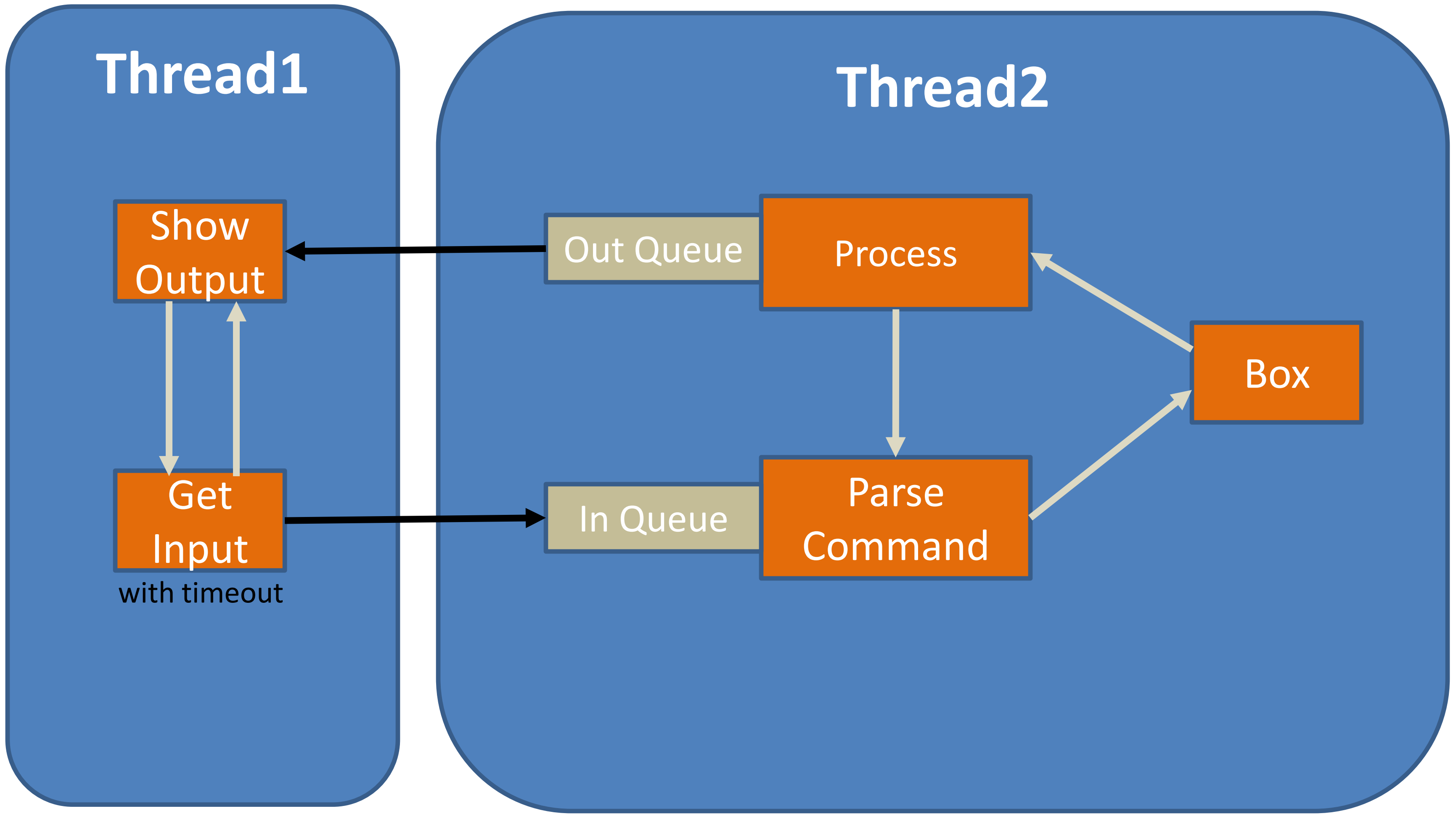


But we can do better!



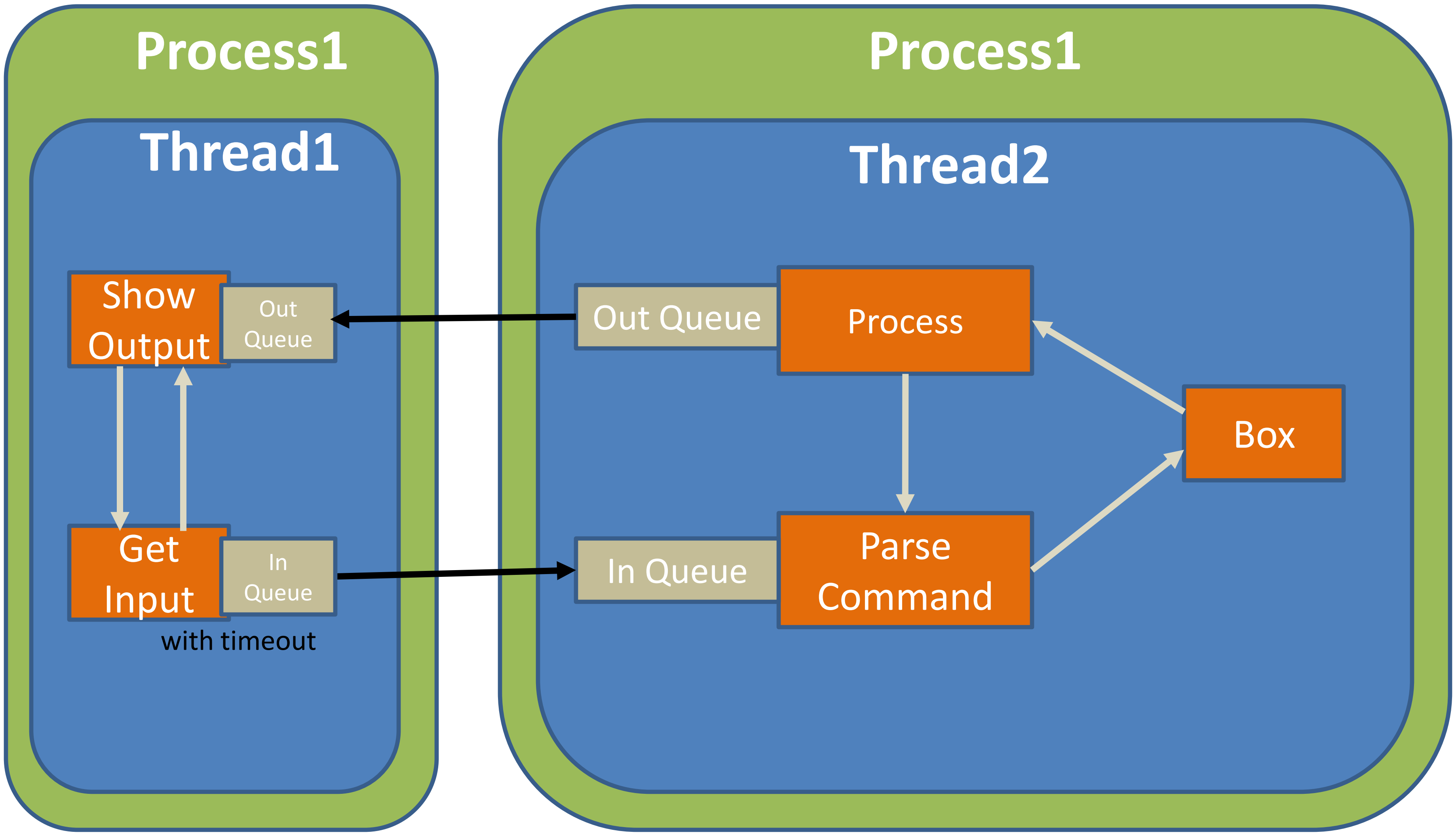
PROBLEM: Not very responsive…

PROBLEM: Streaming data = overflow HW buffer



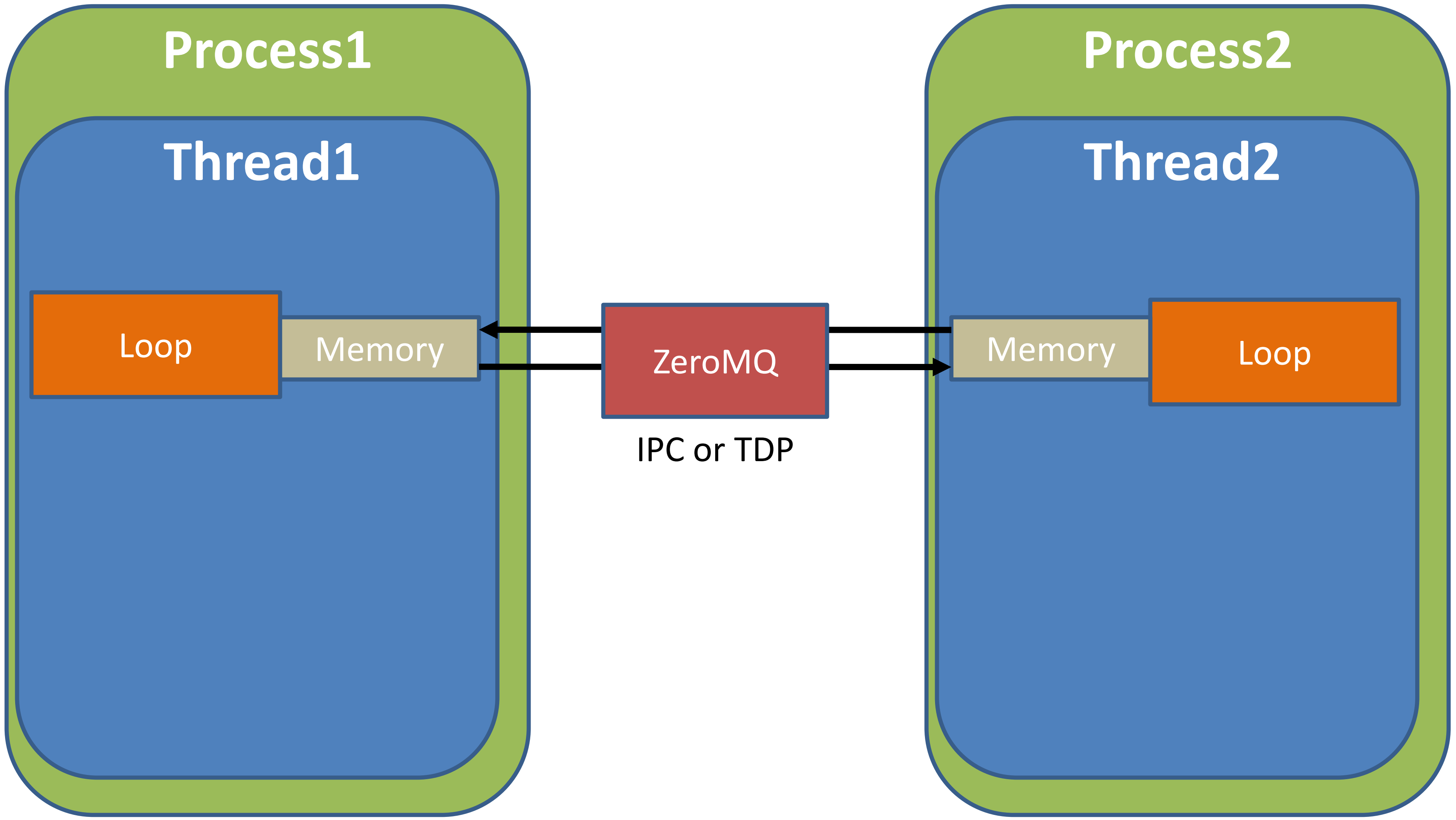
PROBLEM: Any thread crashes = zombie invasion

PROBLEM: Read/write collisions in queues

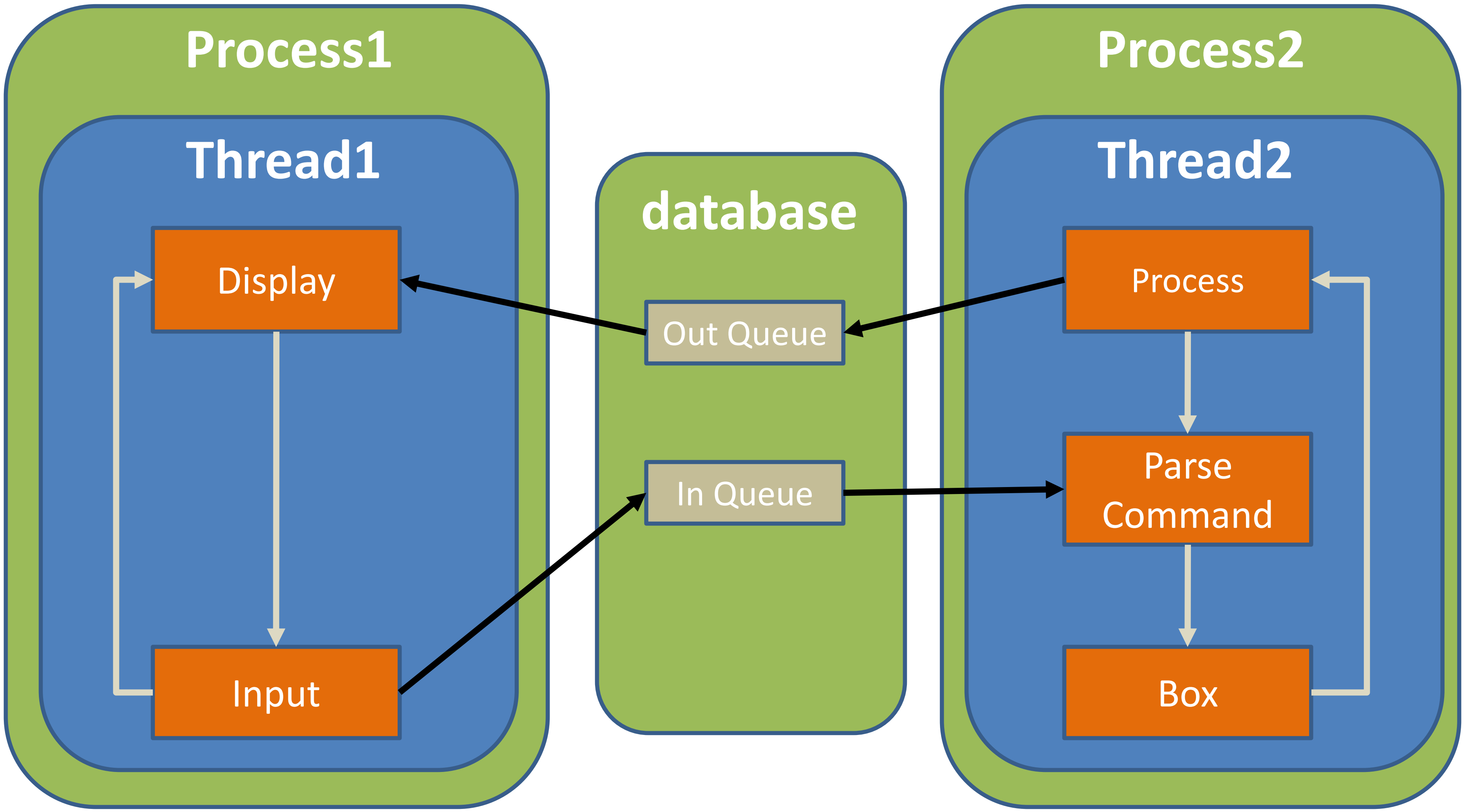


PROBLEM: Passing messages b/w processes is hard

PROBLEM: Double the memory!



PROBLEM: Build your own protocol…
PROBLEM: Double the memory!



DOWNSIDE: More complexity

BONUS: N-M connections – Network for Free – Robust

Could do yet more!

But we'll stop there for now.

BE WARNED:

- Lots of details were skipped!
- Still missing many components for robustness!