class Event(object):

def \_init\_(self):

self.callbacks = []

def notify(self, \*args, \*\*kwargs):

for callback in self.callbacks:

callback(\*args, \*\*kwargs)

def register(self, callback):

self.callbacks.append(callback)

return callback

class SomeData(object):

def \_init\_(self, foo):

self.changed = Event()

self.\_foo = foo

@property

def foo(self):

return self.\_foo

@foo.setter

def foo(self, value):

self.\_foo = value

self.changed.notify(self, 'foo', value)

class SomeGUI(object):

def redraw(self, obj, key, newvalue):

print('redrawing %s with value %s' % (self, newvalue))

if \_name\_ == '\_main\_':

my\_data = SomeData(42)

# Registering the feww function with the use of decorator syntax

@my\_data.changed.register

def print\_it(obj, key, value):

print('Key %s changed to %s' % (key, value))

# Registering the SomeGUI element

my\_gui = SomeGUI()

my\_data.changed.register(my\_gui.redraw)

my\_data.foo = 10