from pydispatch import Dispatcher

class MyEmitter(Dispatcher):

# Events are defined in classes and subclasses with the '\_events\_' attribute

\_events\_ = ['on\_state', 'new\_data']

def do\_some\_stuff(self, data=None):

# do stuff that makes new data

self.get\_some\_data()

# Then emit the change with optional positional and keyword arguments

self.emit('new\_data', data=data)

def get\_some\_data(self):

pass

# An observer - could inherit from Dispatcher or any other class

def on\_emitter\_state():

print('emitter state changed')

def on\_new\_data(\*\*kwargs):

data = kwargs.get('data')

print('I got data: {}'.format(data))

class MyListener(object):

pass

emitter = MyEmitter()

listener = MyListener()

emitter.bind(on\_state=on\_emitter\_state)

emitter.bind(new\_data=on\_new\_data)

emitter.do\_some\_stuff()

# >>> I got data: ...

emitter.emit('on\_state')

# >>> emitter state changed