我需要的文件：

Dlg\_xxxx.ui

Dlg\_xxxx.py：ui文件编译的.py文件

Func\_xxxx.py：函数类文件

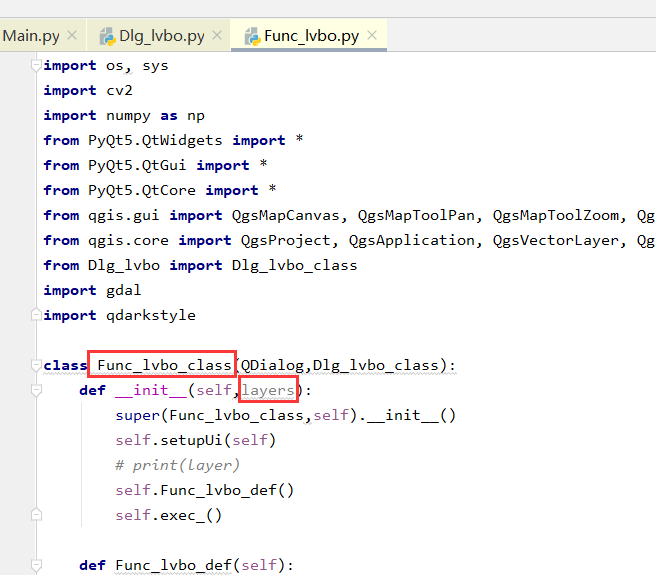
（xxxx自定义）

**1. 设计UI**

设计好UI后，编译成.py文件。

**2. 编写函数文件Func\_xxxx.py**

将所有你要传出的变量与功能封装到一个类里面，取名为**Func\_xxxx\_class**，同时在init函数里传入栅格layerset类型变量，利用这个变量写你们的程序。注意，需要判断layerset的数组个数，如果只有1个值，应该是多波段的影像；如果有多个值，应该是各单波段的影像。



**3. 返回信息**

我需要处理结果，以QgsRasterLayer的格式返回，变量取名为**result**，这样我可以直接加到地图里。

同时需要传递信号，以判断是否运行成功，与是否需要将结果加载到图层中。我需要利用这一变量进行判断。

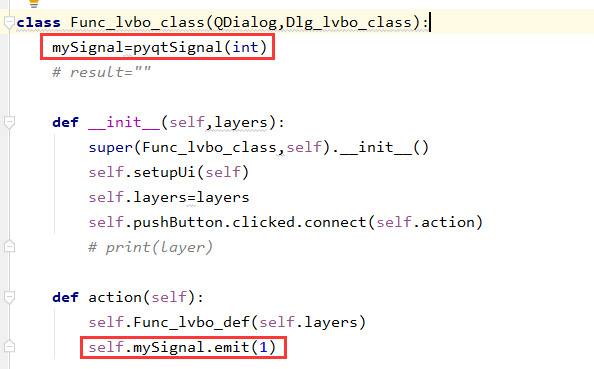
该信号代替了success与loadmap变量的作用，其中信号发送

-1表示未运行成功；

0表示运行成功但不加载结果；

1表示运行成功且加载结果。

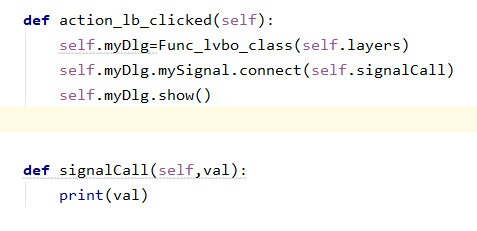
信号命名为**mySignal**，如下图所示



如果有其他需要返回的信息，与我协商。

**4. 注意事项**

在上述的功能函数类**Func\_xxxx\_class**中，不要进行窗口的显示（show()或exec\_()），我将在主函数进行窗口的显示。下图为我在主函数中对对话框的使用方法。



**5. 统一方法**

需要对输入的layerset进行处理，将其转化为单波段的图层数组。处理函数写在类的初始化函数里。（见附录一）

为自己的对话框加上下拉框，给每个指定波段选择波段。（见附录三）

在执行功能时，需要进行判断：

1）波段是否为空

2）所有选择的波段的宽高是否一致

6. 文件夹Image有示例图片，可以尝试使用；文件夹processImage可以用来存放中间计算结果，最好处理完后清空

**附录**

**一、传入的layerset转化成numpy代码**

# 状态变量、信息变量初始化  
        self.success = False  
        self.errorMessage = &apos;&apos;  
        # 对图层进行处理，处理成cv的numpy数组  
        self.bandName = []  
        self.bands = []  
        self.projections = []  
        self.geoTransform = []  
        for i in range(0, len(layers)):  
            tempCount = layers[i].bandCount()  
            tempName = layers[i].name()  
            tempDs = gdal.Open(layers[i].source())  
            for j in range(1, tempCount + 1):  
                self.bandName.append(tempName + &apos;@&apos; + str(j))  
                tempband = np.array(tempDs.GetRasterBand(j).ReadAsArray())  
                self.bands.append(tempband)  
                self.projections.append(tempDs.GetProjection())  
                self.geoTransform.append(tempDs.GetGeoTransform())

**二、gdal保存成tiff文件的函数**

  def create\_tiff(self, fullPath, projection, geoTransform, data):  
        driver = gdal.GetDriverByName("GTiff")  
        if &apos;int8&apos; in data.dtype.name:  
            datatype = gdal.GDT\_Byte  
        elif &apos;int16&apos; in data.dtype.name:  
            datatype = gdal.GDT\_UInt16  
        else:  
            datatype = gdal.GDT\_Float32  
        outDs = driver.Create(fullPath, data.shape[1], data.shape[0], data.shape[2], datatype)  
        outDs.SetGeoTransform(geoTransform)  
        outDs.SetProjection(projection)  
        outDs.GetRasterBand(1).WriteArray(data[:, :, 2])  
        outDs.GetRasterBand(2).WriteArray(data[:, :, 1])  
        outDs.GetRasterBand(3).WriteArray(data[:, :, 0])  
        del outDs

**三、在选项中加载波段**

 # 在选项中加载波段  
        self.comboBox\_r.clear()  
        self.comboBox\_g.clear()  
        self.comboBox\_b.clear()  
        for i in range(0, len(self.bandName)):  
            self.comboBox\_r.addItem(self.bandName[i], i)  
            self.comboBox\_g.addItem(self.bandName[i], i)  
            self.comboBox\_b.addItem(self.bandName[i], i)

**四、合并波段**

img = cv2.merge([self.bands[index\_b], self.bands[index\_g], self.bands[index\_r]])