第一种：函数直接创建

from PySide2.QtWidgets import QApplication, QMainWindow, QPushButton, QPlainTextEdit, QMessageBox  
  
def handleCalc():  
 info = textEdit.toPlainText()  
 # 薪资20000 以上 和 以下 的人员名单  
 salary\_above\_20k = '' #创建变量  
 salary\_below\_20k = '' #创建变量  
 for line in info.splitlines(): #换行符（分割线）拆分  
 if not line.strip(): # 去掉列表中的空字符串内容  
 continue  
 parts = line.split(' ') #通过‘ ’ 拆分  
 # 去掉列表中的空字符串内容  
 parts = [p for p in parts if p] #创建列表  
 name,salary,age = parts #变量绑定  
 if int(salary) >= 20000:  
 salary\_above\_20k += name + '\n'  
 else:  
 salary\_below\_20k += name + '\n'  
  
 QMessageBox.about(window,  
 '统计结果',  
 f'''薪资20000 以上的有：\n{salary\_above\_20k}  
 \n薪资20000 以下的有：\n{salary\_below\_20k}'''  
 ) #在window窗口创建提示窗口  
  
  
app = QApplication([]) #整个图形界面程序的底层管理功能  
  
window = QMainWindow() #创建主窗口  
window.resize(500, 400) #窗口大小  
window.move(300, 310) #屏幕的位置  
window.setWindowTitle('薪资统计') #窗口标题  
  
textEdit = QPlainTextEdit(window) #在window窗口内创建文本  
textEdit.setPlaceholderText("请输入薪资表") #文本提示  
textEdit.move(10,25) #在window窗口内的相对位置  
textEdit.resize(300,350) #文本框大小  
  
button = QPushButton('统计', window) #在window窗口内创建'统计'控件  
button.move(380,80) #在window窗口内的相对位置  
button.clicked.connect(handleCalc) #按钮被点击让 handleCalc函数 来处理  
  
window.show() #显示窗口  
  
app.exec\_() #app死循环，用来一直显示

from PySide2.QtWidgets import QApplication, QMainWindow, QPushButton, QPlainTextEdit,QMessageBox  
  
class Stats(object):  
 def \_\_init\_\_(self):  
 self.window = QMainWindow()  
 self.window.resize(500, 400)  
 self.window.move(300, 300)  
 self.window.setWindowTitle('薪资统计')  
  
 self.textEdit = QPlainTextEdit(self.window)  
 self.textEdit.setPlaceholderText("请输入薪资表")  
 self.textEdit.move(10, 25)  
 self.textEdit.resize(300, 350)  
  
 self.button = QPushButton('统计', self.window)  
 self.button.move(380, 80)  
 self.button.clicked.connect(self.handleCalc)  
  
  
 def handleCalc(self):  
 info = self.textEdit.toPlainText()  
  
 # 薪资20000 以上 和 以下 的人员名单  
 salary\_above\_20k = ''  
 salary\_below\_20k = ''  
 for line in info.splitlines():  
 if not line.strip():  
 continue  
 parts = line.split(' ')  
 # 去掉列表中的空字符串内容  
 parts = [p for p in parts if p]  
 name,salary,age = parts  
 if int(salary) >= 20000:  
 salary\_above\_20k += name + '\n'  
 else:  
 salary\_below\_20k += name + '\n'  
  
 QMessageBox.about(self.window,  
 '统计结果',  
 f'''薪资20000 以上的有：\n{salary\_above\_20k}  
 \n薪资20000 以下的有：\n{salary\_below\_20k}'''  
 )  
  
app = QApplication([])  
stats = Stats()  
stats.window.show()  
app.exec\_()

第二种：调用UI生成的py文件

# cheshi1.py  
  
# -\*- coding: utf-8 -\*-  
  
# Form implementation generated from reading ui file 'ceshi1.ui',  
# licensing of 'ceshi1.ui' applies.  
#  
# Created: Mon Dec 9 07:54:57 2019  
# by: pyside2-uic running on PySide2 5.13.0  
#  
# WARNING! All changes made in this file will be lost!  
  
from PySide2 import QtCore, QtGui, QtWidgets  
  
class Ui\_stats\_2(object):  
 def setupUi(self, stats\_2):  
 stats\_2.setObjectName("stats\_2")  
 stats\_2.resize(324, 290)  
 self.stats = QtWidgets.QWidget(stats\_2)  
 self.stats.setObjectName("stats")  
 self.textEdit = QtWidgets.QPlainTextEdit(self.stats)  
 self.textEdit.setGeometry(QtCore.QRect(10, 10, 181, 241))  
 self.textEdit.setObjectName("textEdit")  
 self.button = QtWidgets.QPushButton(self.stats)  
 self.button.setGeometry(QtCore.QRect(230, 10, 75, 23))  
 self.button.setObjectName("button")  
 stats\_2.setCentralWidget(self.stats)  
 self.menubar = QtWidgets.QMenuBar(stats\_2)  
 self.menubar.setGeometry(QtCore.QRect(0, 0, 324, 23))  
 self.menubar.setObjectName("menubar")  
 self.aaa = QtWidgets.QMenu(self.menubar)  
 self.aaa.setObjectName("aaa")  
 self.bbb = QtWidgets.QMenu(self.menubar)  
 self.bbb.setObjectName("bbb")  
 self.ccc = QtWidgets.QMenu(self.menubar)  
 self.ccc.setObjectName("ccc")  
 stats\_2.setMenuBar(self.menubar)  
 self.a1 = QtWidgets.QAction(stats\_2)  
 self.a1.setObjectName("a1")  
 self.a2 = QtWidgets.QAction(stats\_2)  
 self.a2.setObjectName("a2")  
 self.b1 = QtWidgets.QAction(stats\_2)  
 self.b1.setObjectName("b1")  
 self.b2 = QtWidgets.QAction(stats\_2)  
 self.b2.setObjectName("b2")  
 self.c1 = QtWidgets.QAction(stats\_2)  
 self.c1.setObjectName("c1")  
 self.c2 = QtWidgets.QAction(stats\_2)  
 self.c2.setObjectName("c2")  
 self.aaa.addAction(self.a1)  
 self.aaa.addAction(self.a2)  
 self.bbb.addAction(self.b1)  
 self.bbb.addAction(self.b2)  
 self.ccc.addAction(self.c1)  
 self.ccc.addAction(self.c2)  
 self.menubar.addAction(self.aaa.menuAction())  
 self.menubar.addAction(self.bbb.menuAction())  
 self.menubar.addAction(self.ccc.menuAction())  
  
 self.retranslateUi(stats\_2)  
 QtCore.QMetaObject.connectSlotsByName(stats\_2)  
  
 def retranslateUi(self, stats\_2):  
 stats\_2.setWindowTitle(QtWidgets.QApplication.translate("stats\_2", "测试程式", None, -1))  
 self.textEdit.setPlainText(QtWidgets.QApplication.translate("stats\_2", "请输入文本内容", None, -1))  
 self.button.setText(QtWidgets.QApplication.translate("stats\_2", "统计", None, -1))  
 self.aaa.setTitle(QtWidgets.QApplication.translate("stats\_2", "aaa", None, -1))  
 self.bbb.setTitle(QtWidgets.QApplication.translate("stats\_2", "bbb", None, -1))  
 self.ccc.setTitle(QtWidgets.QApplication.translate("stats\_2", "ccc", None, -1))  
 self.a1.setText(QtWidgets.QApplication.translate("stats\_2", "a1", None, -1))  
 self.a2.setText(QtWidgets.QApplication.translate("stats\_2", "a2", None, -1))  
 self.b1.setText(QtWidgets.QApplication.translate("stats\_2", "b1", None, -1))  
 self.b2.setText(QtWidgets.QApplication.translate("stats\_2", "b2", None, -1))  
 self.c1.setText(QtWidgets.QApplication.translate("stats\_2", "c1", None, -1))  
 self.c2.setText(QtWidgets.QApplication.translate("stats\_2", "c1", None, -1))

# -\*- coding: utf-8 -\*-  
import sys, os  
from PySide2.QtWidgets import QMainWindow, QFileDialog, QApplication  
import ceshi1 as design  
# 创建APP类，并继承UI的类  
class WoHeYunApp(QMainWindow, design.Ui\_stats\_2):  
 # 初始化构造函数  
 def \_\_init\_\_(self):  
 # 继承： super指父类（子类，实例）.构造函数  
 super(WoHeYunApp, self).\_\_init\_\_()  
 self.setupUi(self)  
 # 把UI中的控件连接函数功能（事件）  
 self.button.clicked.connect(self.browse\_folder)  
  
 # 功能函数  
 def browse\_folder(self):  
 # 清除列表内容  
 self.listWidget.clear()  
 # 创建文件夹选择对话框  
 directory = QFileDialog.getExistingDirectory(self, "Pick a folder")  
 # 如果有范围的选择路径则搜索该路径下所有存在的文件，并将其名字加载都列表中  
 if directory:  
 for file\_name in os.listdir(directory):  
 self.listWidget.addItem(file\_name)  
  
  
# 创建实例  
def main():  
 # 创建新的实例应用  
 app = QApplication(sys.argv)  
 # 我们将表单设置为WoHeYunApp  
 widgets = WoHeYunApp()  
 # 显示我们的表单  
 widgets.show()  
 # 退出程序  
 app.exec\_()  
  
  
# 如果我们直接运行文件而不是导入它，则执行  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

第三种：直接调用UI界面文件

class Stats:  
  
 def \_\_init\_\_(self):  
 # 从文件中加载UI定义  
 qfile\_stats = QFile("ce\_shi.ui")  
 qfile\_stats.open(QFile.ReadOnly)  
 qfile\_stats.close()  
  
 # 从 UI 定义中动态 创建一个相应的窗口对象  
 # 注意：里面的控件对象也成为窗口对象的属性了  
 # 比如 self.ui.button , self.ui.textEdit  
 self.ui = QUiLoader().load(qfile\_stats)  
  
 #self.ui.button.clicked.connect(self.handleCalc)  
  
 def handleCalc(self):  
 info = self.ui.textEdit.toPlainText()  
  
 salary\_above\_20k = ''  
 salary\_below\_20k = ''  
 for line in info.splitlines():  
 if not line.strip():  
 continue  
 parts = line.split(' ')  
  
 parts = [p for p in parts if p]  
 name,salary,age = parts  
 if int(salary) >= 20000:  
 salary\_above\_20k += name + '\n'  
 else:  
 salary\_below\_20k += name + '\n'  
  
 QMessageBox.about(self.ui,  
 '统计结果',  
 f'''薪资20000 以上的有：\n{salary\_above\_20k}  
 \n薪资20000 以下的有：\n{salary\_below\_20k}'''  
 )  
  
app = QApplication([])  
stat = Stats()  
stat.ui.show()  
app.exec\_()