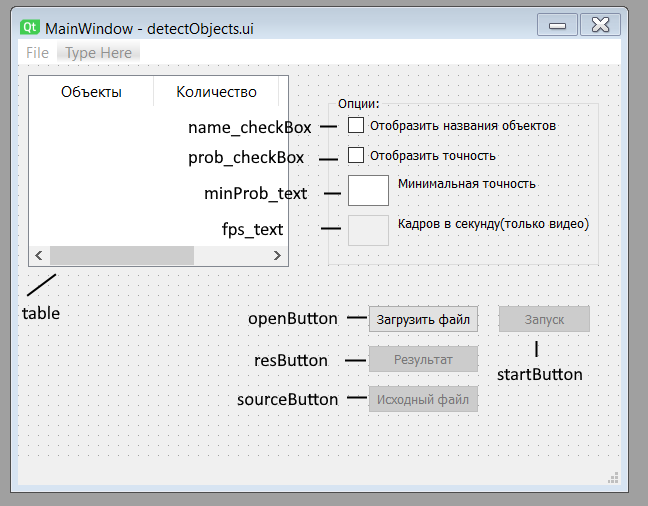
Отчет по лабораторной работе №16:

# СППР «РАСПОЗНАВАНИЕ И ОТСЛЕЖИВАНИЕ ОБЪЕКТОВ». Часть 2.

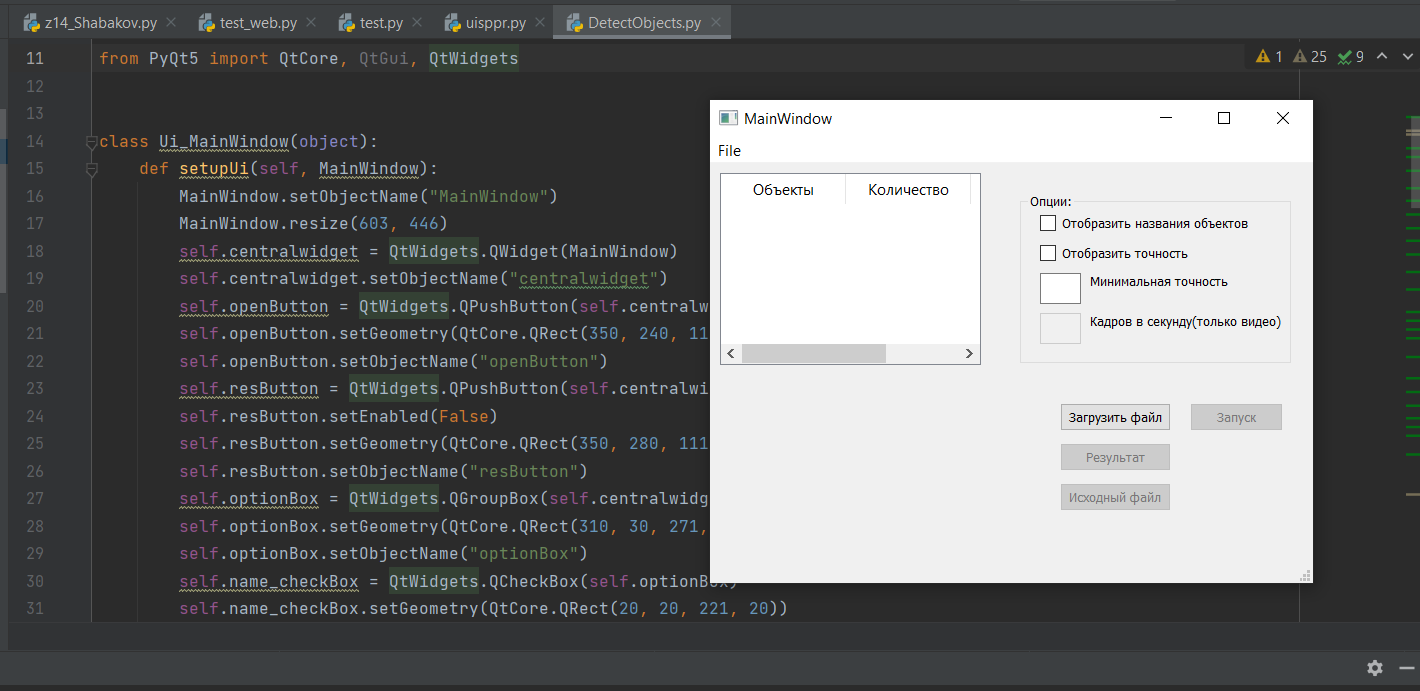
# Цель: Написать программу для создания СППР для распознавания объектов на картинке и отслеживания объектов на видео с заданной пользователем вероятностью.

# Интерфейс приложения:



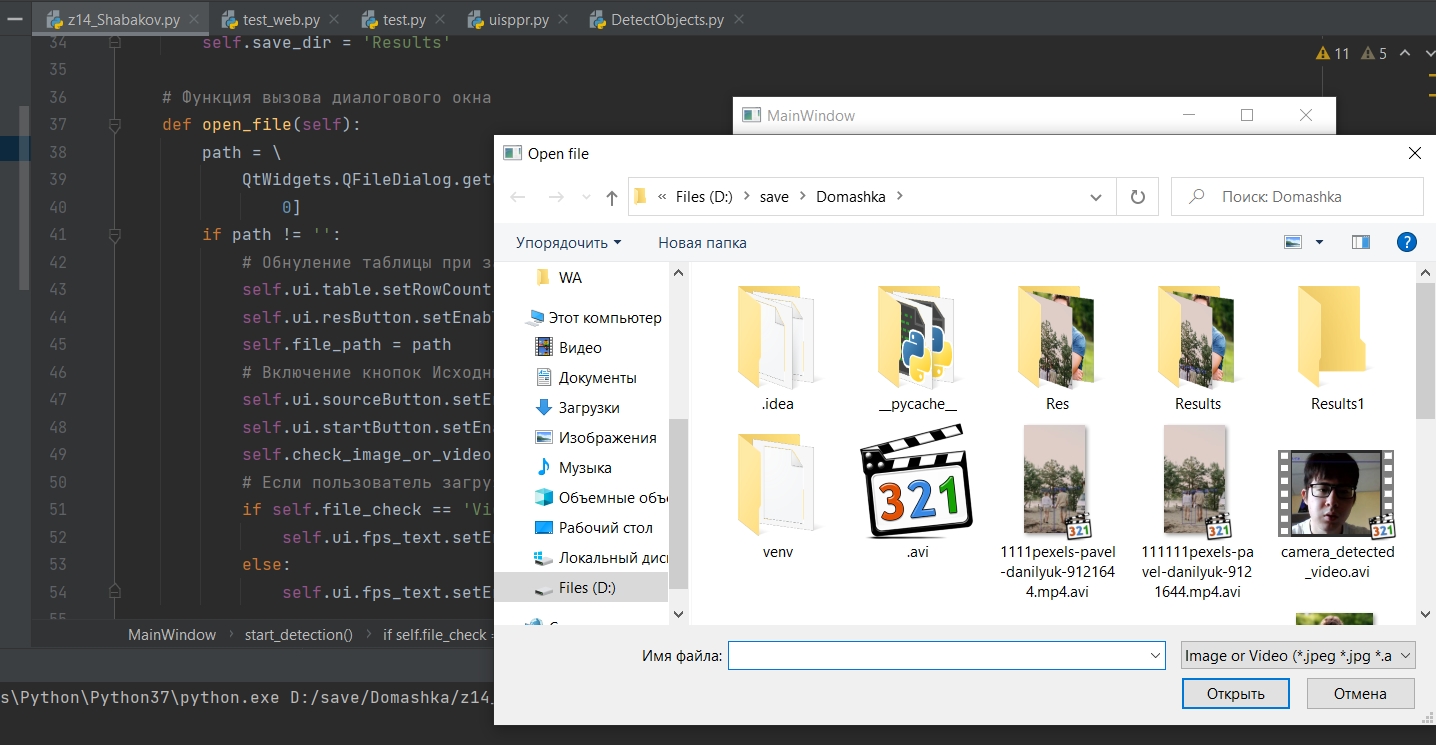
# Логика программы:

При запуске программы пользователь видит интерфейс приложения:



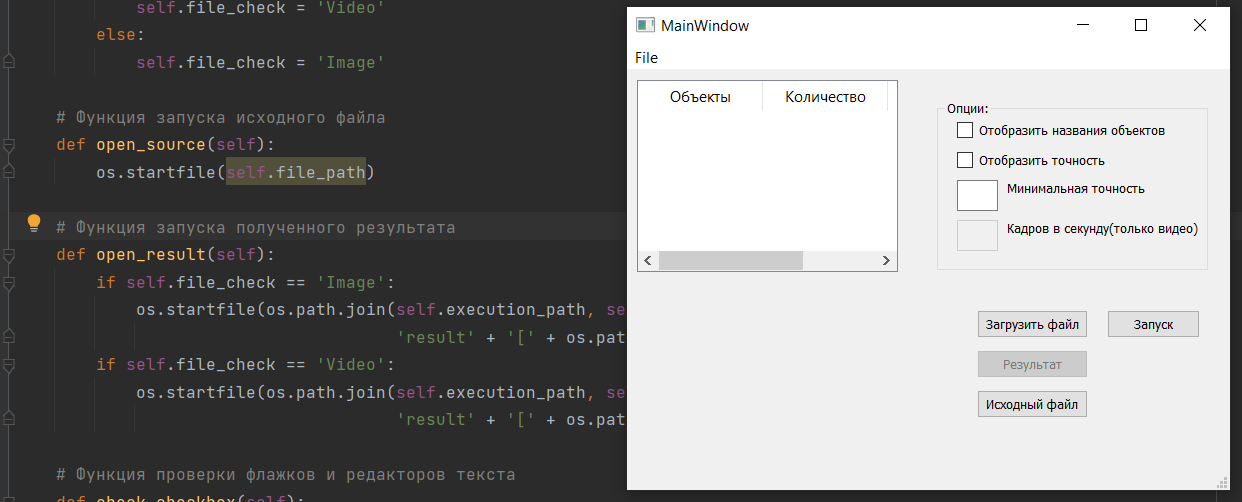
Интерфейс был выполнен в QtDesigner и конвертирован в расширение .py(код приложен, название файла: **DetectObjects.py).** **Команда конвертирования: pyuic5 DetectObjects.ui -o DetectObjects.py.**

Пользователь может загрузить файл, нажав кнопку “Загрузить файл”, после чего вызовется диалоговое окно:



Пользователю нужно выбрать исходный файл для обработки и нажать на кнопку “Открыть”.

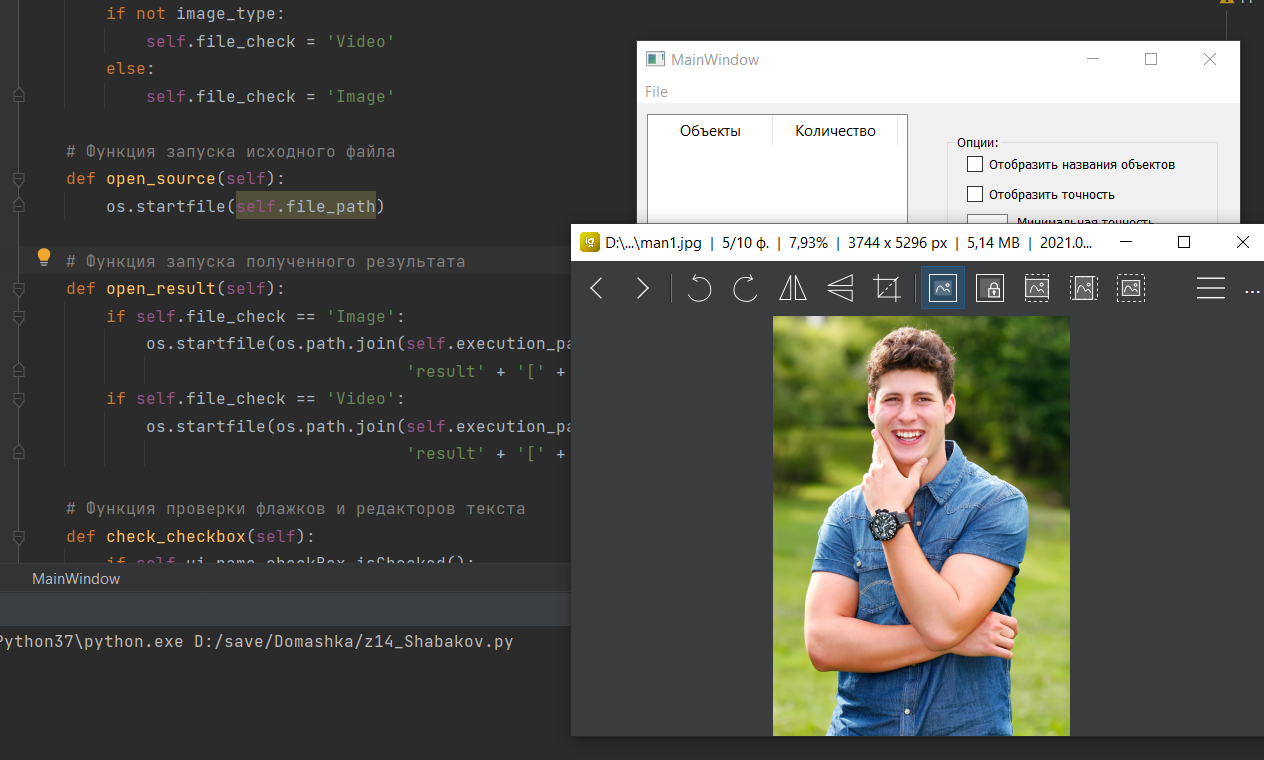
После чего станут доступны кнопки “Запуск” и “Исходный файл”:



**Фрагмент кода, отвечающий за обработку кнопки “Открыть”:**

def open\_file(self):  
 path = \  
 QtWidgets.QFileDialog.getOpenFileName(self, 'Open file', None, "Image or Video (\*.jpeg \*.jpg \*.avi \*.mp4)")[  
 0]  
 if path != '':  
 #Обнуление таблицы при загрузке нового файла  
 self.ui.table.setRowCount(0)  
 self.ui.resButton.setEnabled(False)  
 self.file\_path = path  
 # Включение кнопок Исходный файл и Запуск  
 self.ui.sourceButton.setEnabled(True)  
 self.ui.startButton.setEnabled(True)  
 self.check\_image\_or\_video()  
 # Если пользователь загрузил видео- включение редактора текста Количество кадров  
 if self.file\_check == 'Video':  
 self.ui.fps\_text.setEnabled(True)  
 else:  
 self.ui.fps\_text.setEnabled(False)

Нажмем на кнопку “Исходный файл”:

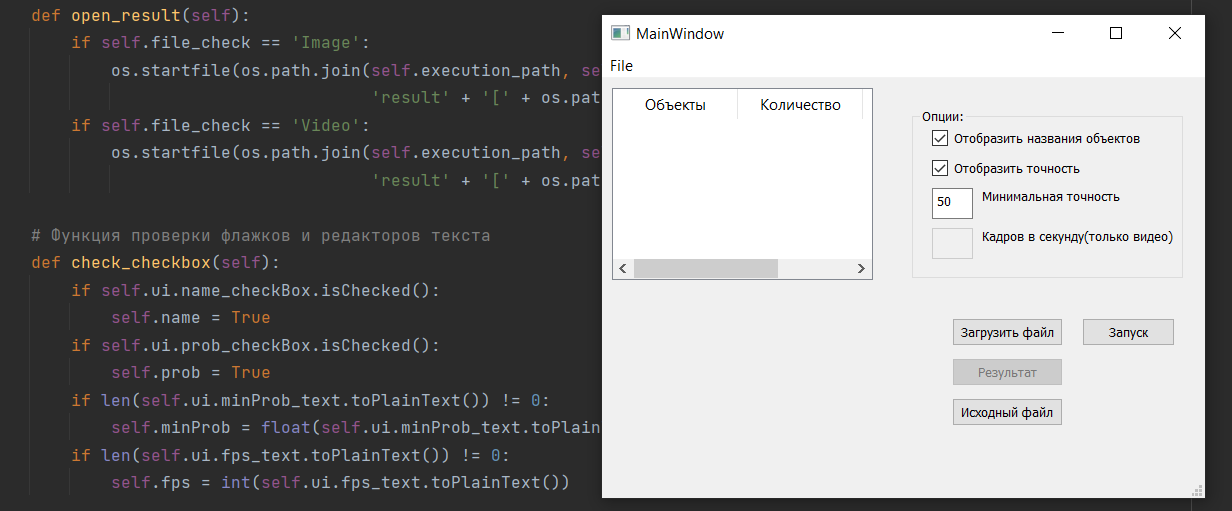


Как видим, открывается выбранное нами изображение для обработки.

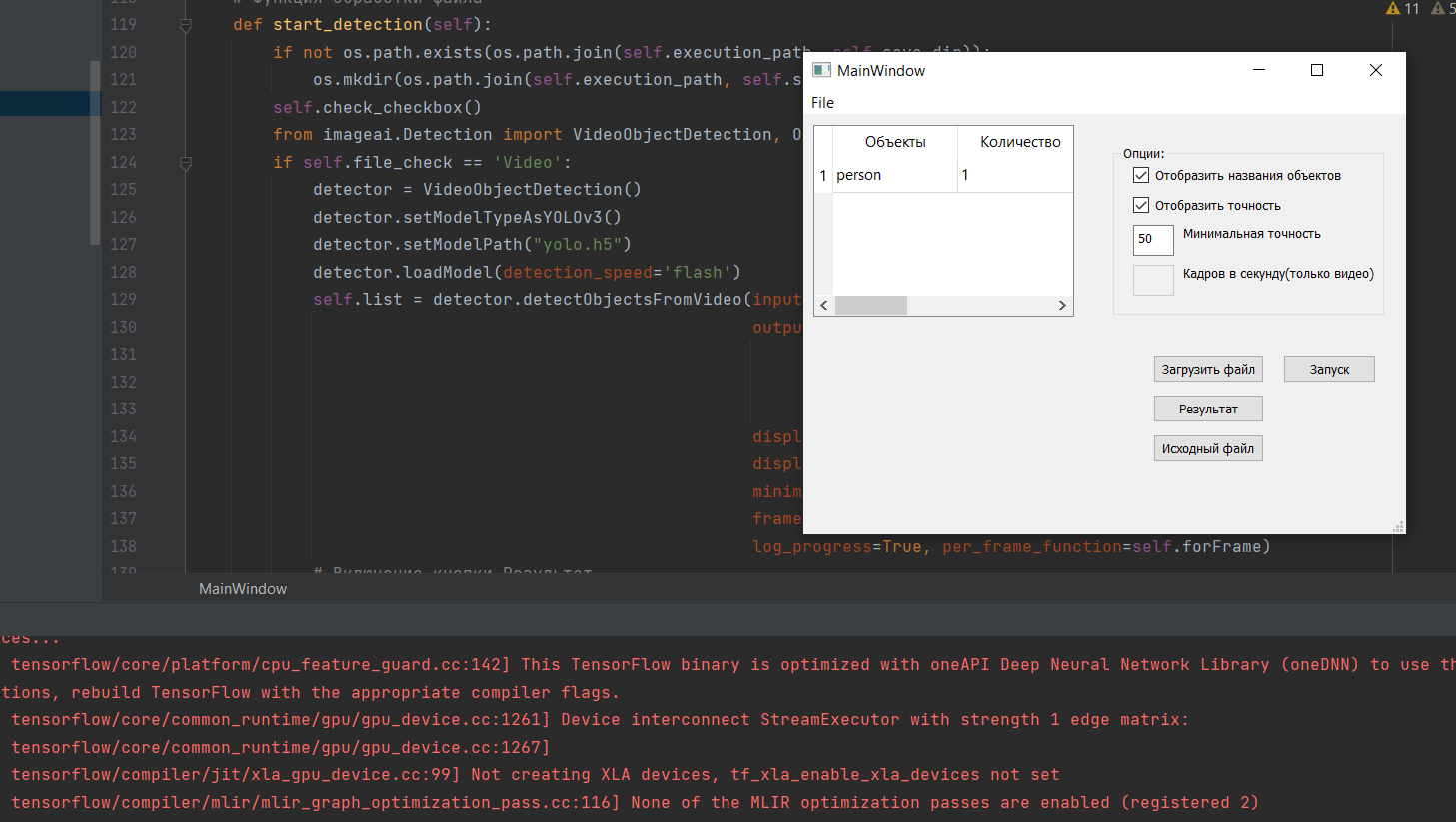
**Фрагмент кода, отвечающий за работу кнопки “Исходный”:**

# Функция запуска исходного файла  
def open\_source(self):  
 os.startfile(self.file\_path)

Теперь обработаем фотографию. В меню “Опции” выберем некоторые из пунктов, минимальная точность будет равна 50 процентам:



Запускаем:

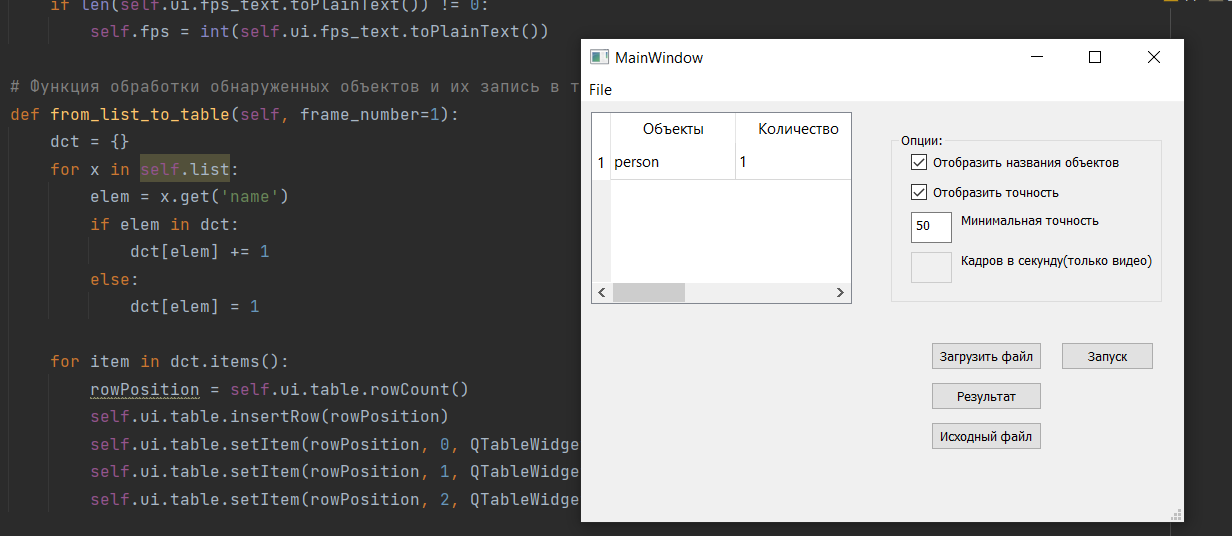


Изображение начало обрабатываться.

**Фрагмент кода, отвечающий за обработку изображения:**

if self.file\_check == 'Image':  
 detector = ObjectDetection()  
 detector.setModelTypeAsRetinaNet()  
 detector.setModelPath("resnet50\_coco\_best\_v2.1.0.h5")  
 detector.loadModel(detection\_speed='flash')  
 self.list = detector.detectObjectsFromImage(input\_image=self.file\_path,  
 output\_image\_path=os.path.join(self.execution\_path, self.save\_dir,  
 'result' + '[' + os.path.basename(  
 self.file\_path) +']'+'.jpg'),  
 minimum\_percentage\_probability=self.minProb,  
 display\_percentage\_probability=self.prob,  
 display\_object\_name=self.name)  
 # Включение кнопки Результат  
 self.ui.resButton.setEnabled(True)  
 self.from\_list\_to\_table()

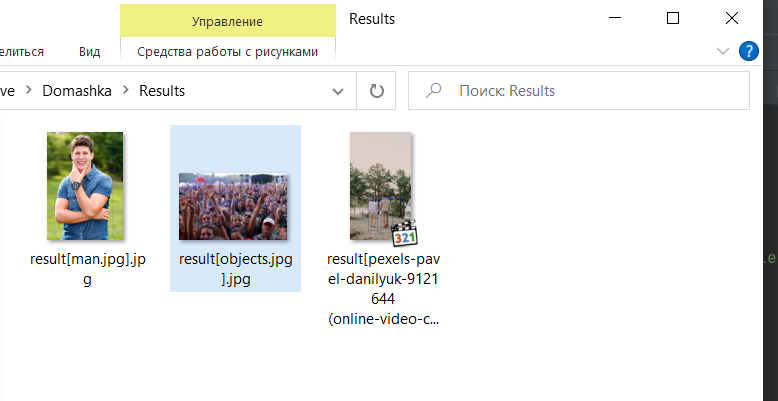
После окончания процесса все обнаруженные объекты и их количество записываются в таблицу:



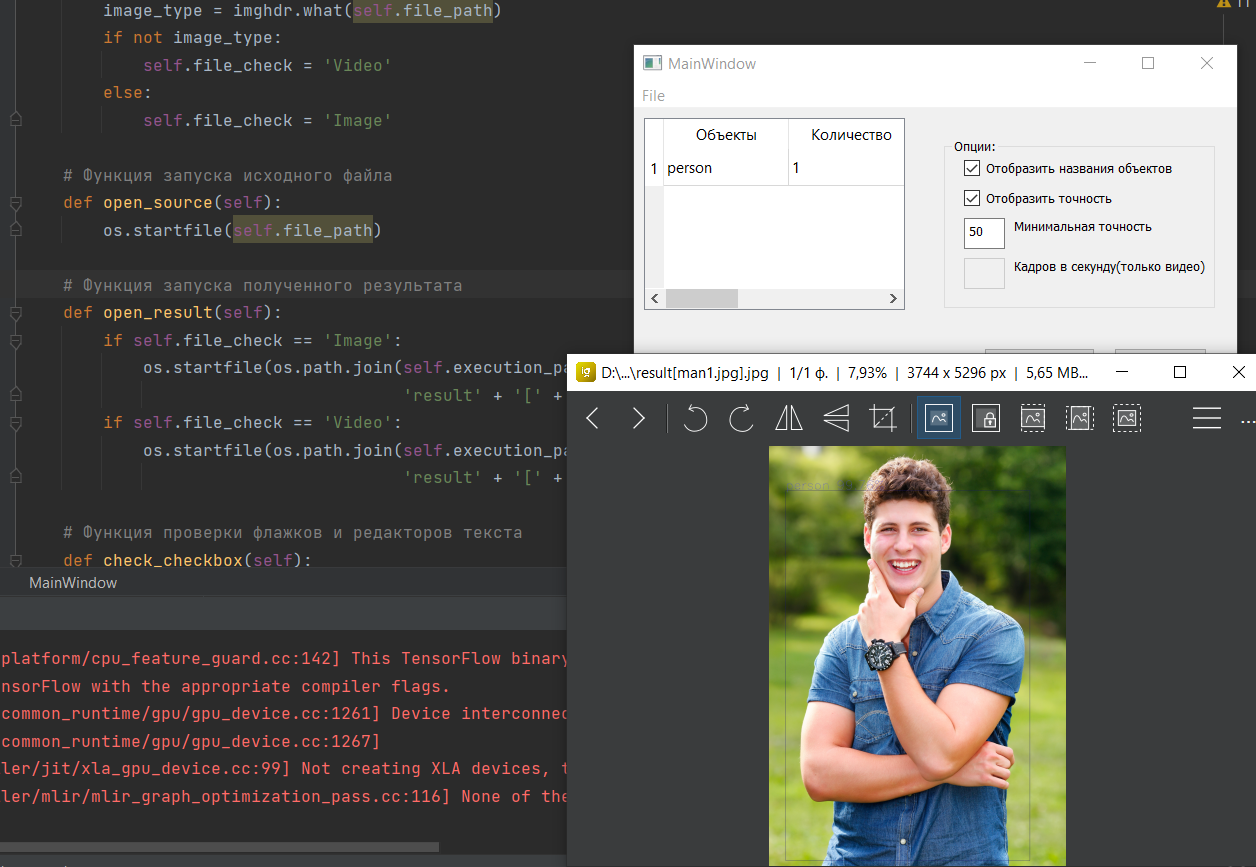
**Фрагмент кода, отвечающий за заполнение таблицы (для изображения):**

# Функция обработки обнаруженных объектов и их запись в таблицу(для фото)  
def from\_list\_to\_table(self):  
 dct = {}  
 for x in self.list:  
 elem = x.get('name')  
 if elem in dct:  
 dct[elem] += 1  
 else:  
 dct[elem] = 1  
  
 for item in dct.items():  
 rowPosition = self.ui.table.rowCount()  
 self.ui.table.insertRow(rowPosition)  
 self.ui.table.setItem(rowPosition, 0, QTableWidgetItem(item[0]))  
 self.ui.table.setItem(rowPosition, 1, QTableWidgetItem(str(item[1])))

Обработанный файл сохраняется в ту же директорию, в папку Results с шаблоном имени “result[название файла].jpg”



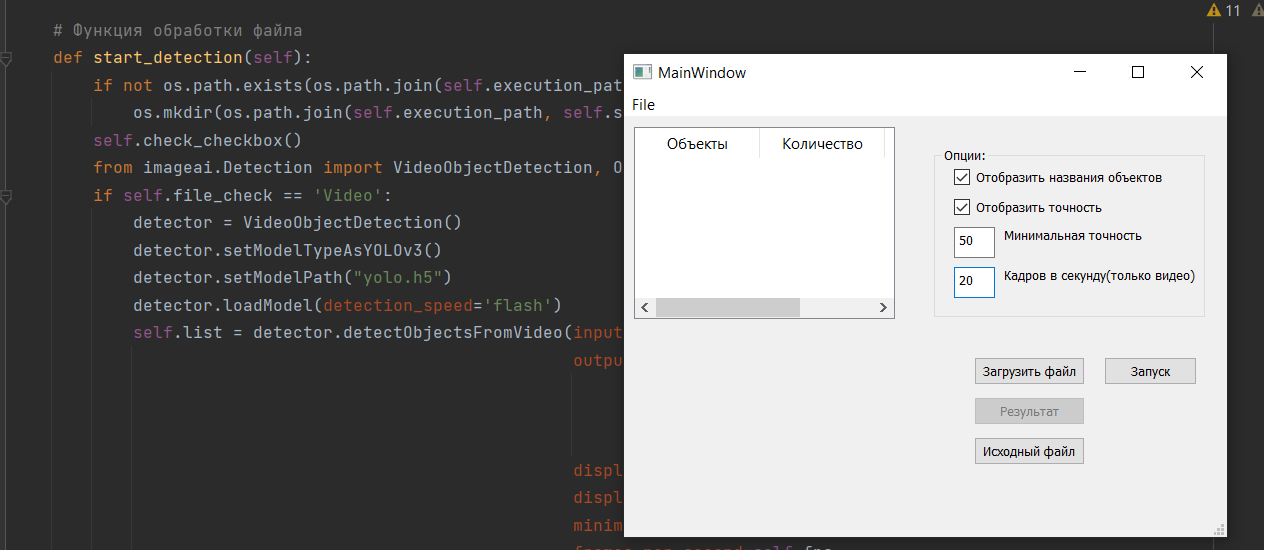
Запустим наше полученное фото из программы. Для этого нажимаем на кнопке “Результат”:



**Фрагмент кода, отвечающий за кнопку “Результат”:**

# Функция запуска полученного результата  
def open\_result(self):  
 if self.file\_check=='Image':  
 os.startfile(os.path.join(self.execution\_path, self.save\_dir,'result' + '[' + os.path.basename(self.file\_path) + ']'+'.jpg'))  
 if self.file\_check == 'Video':  
 os.startfile(os.path.join(self.execution\_path, self.save\_dir,'result' + '[' + os.path.basename(self.file\_path) + ']'+'.

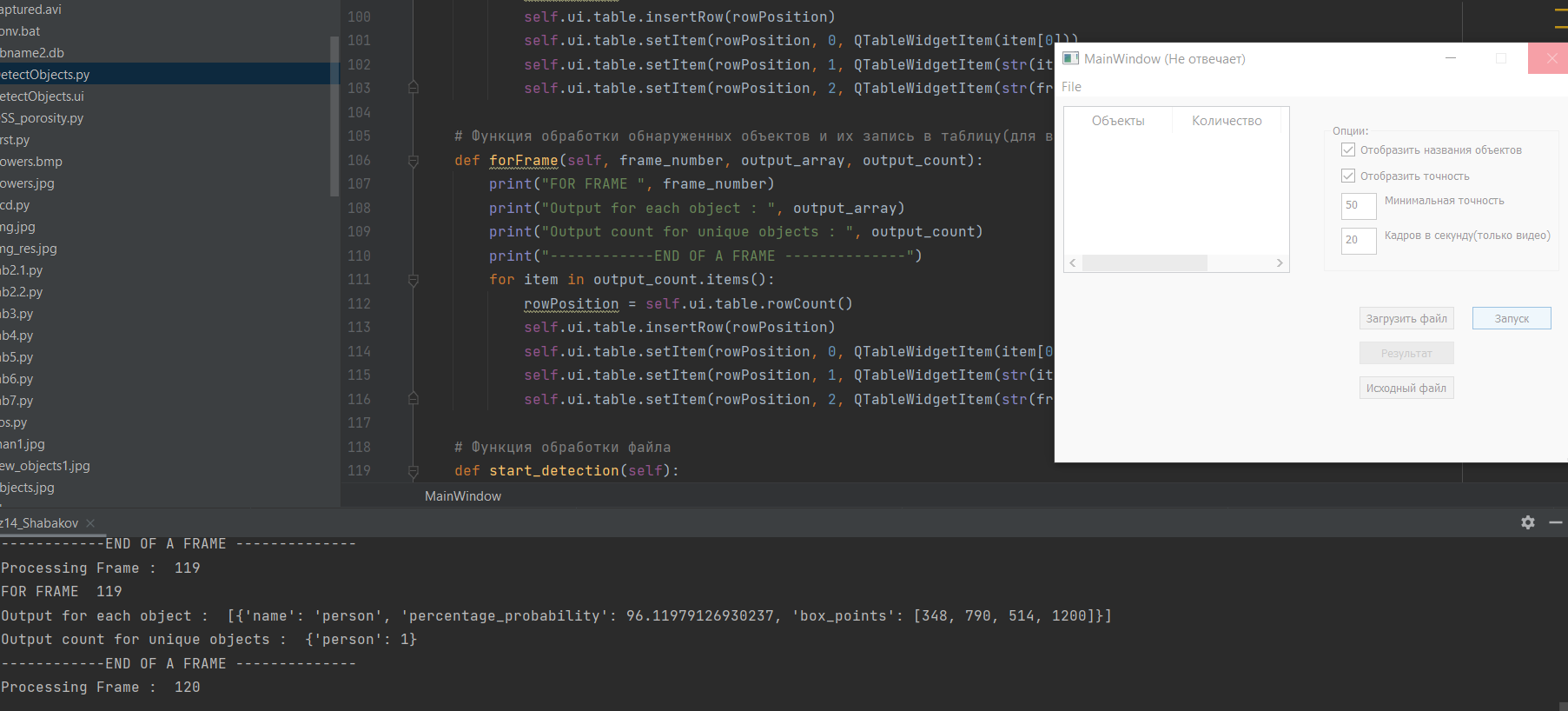
Для видео алгоритм идентичный, только открывается редактор текста “Кадров для видео”:



**Фрагмент кода, отвечающий за обработку видео:**

# Функция обработки файла  
def start\_detection(self):  
 if not os.path.exists(os.path.join(self.execution\_path,self.save\_dir)):  
 os.mkdir(os.path.join(self.execution\_path,self.save\_dir))  
 self.check\_checkbox()  
 from imageai.Detection import VideoObjectDetection, ObjectDetection  
 if self.file\_check == 'Video':  
 detector = VideoObjectDetection()  
 detector.setModelTypeAsYOLOv3()  
 detector.setModelPath("yolo.h5")  
 detector.loadModel(detection\_speed='flash')  
 self.list = detector.detectObjectsFromVideo(input\_file\_path=self.file\_path,  
 output\_file\_path=os.path.join(self.execution\_path, self.save\_dir,  
 'result' + '[' + os.path.basename(  
 self.file\_path) + ']'),  
 display\_object\_name=self.name,  
 display\_percentage\_probability=self.prob,  
 minimum\_percentage\_probability=self.minProb,  
 frames\_per\_second=self.fps,  
 log\_progress=True,per\_frame\_function=self.forFrame)  
 # Включение кнопки Результат  
 self.ui.resButton.setEnabled(True)

В консоли отображается количество обнаруженных объектов за кадр. Эти данные нужны для заполнения таблицы:

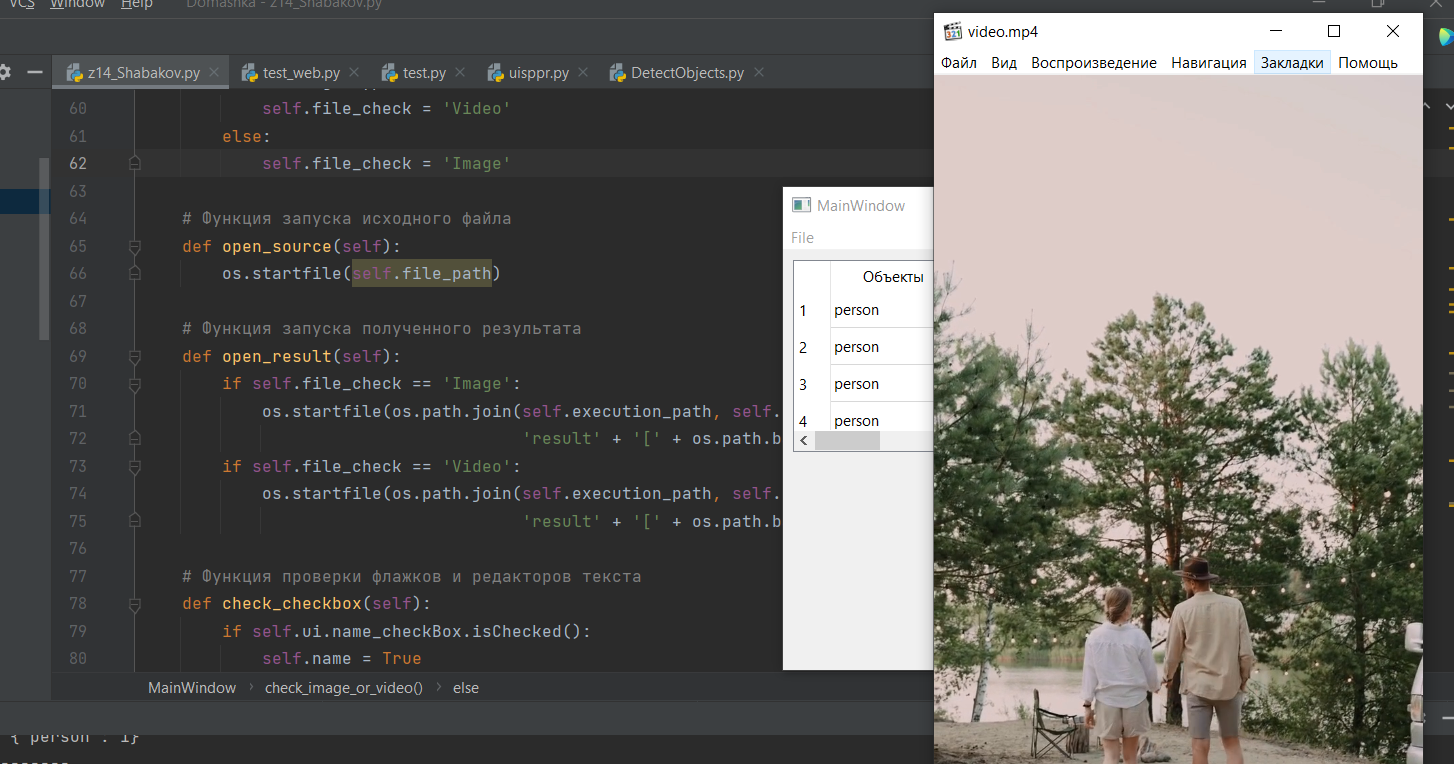


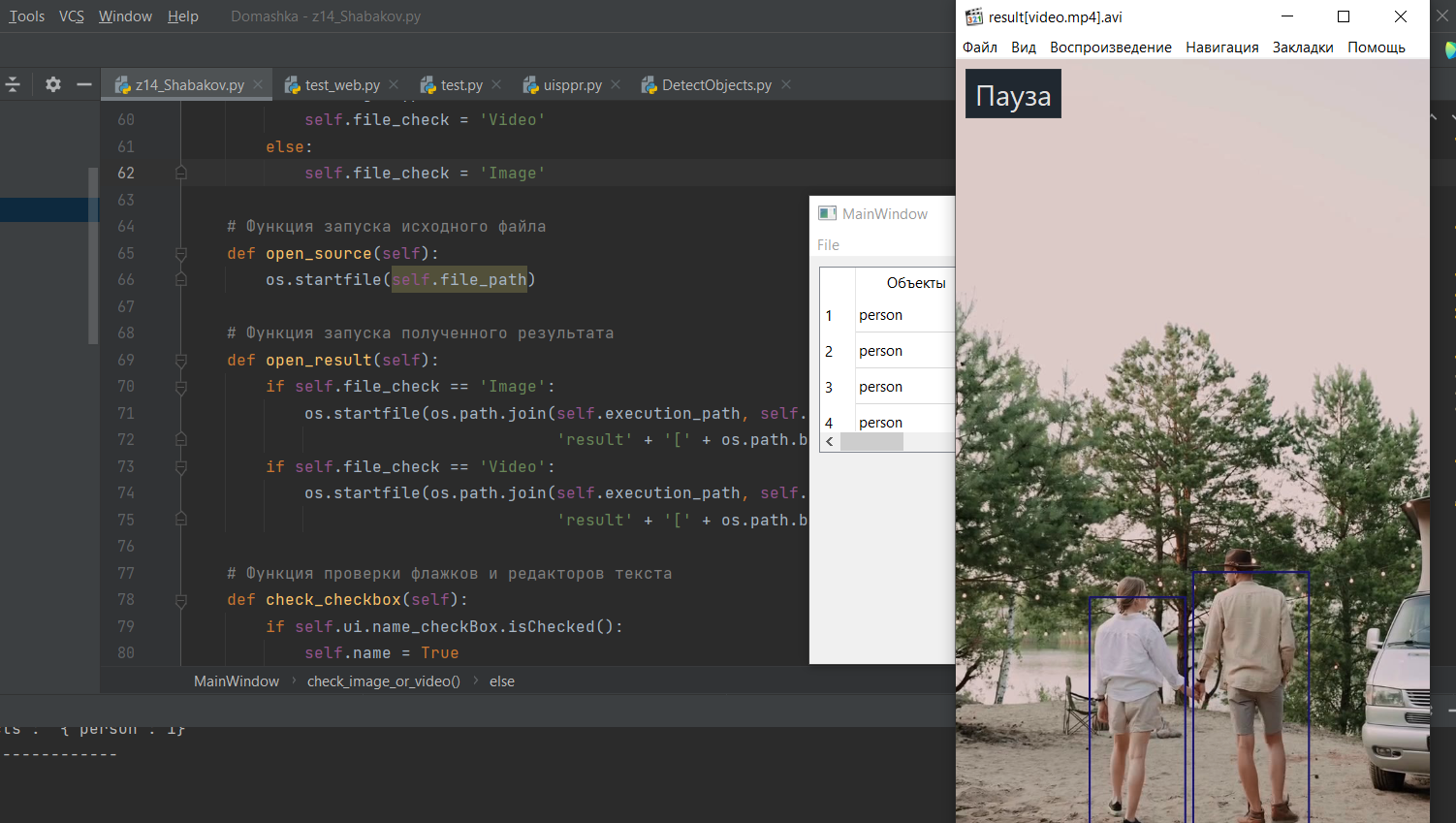
В таблице отображаются уникальные объекты за каждый кадр.

**Фрагмент кода, отвечающий за обработку полученных данных и занесения их в таблицу для видео:**

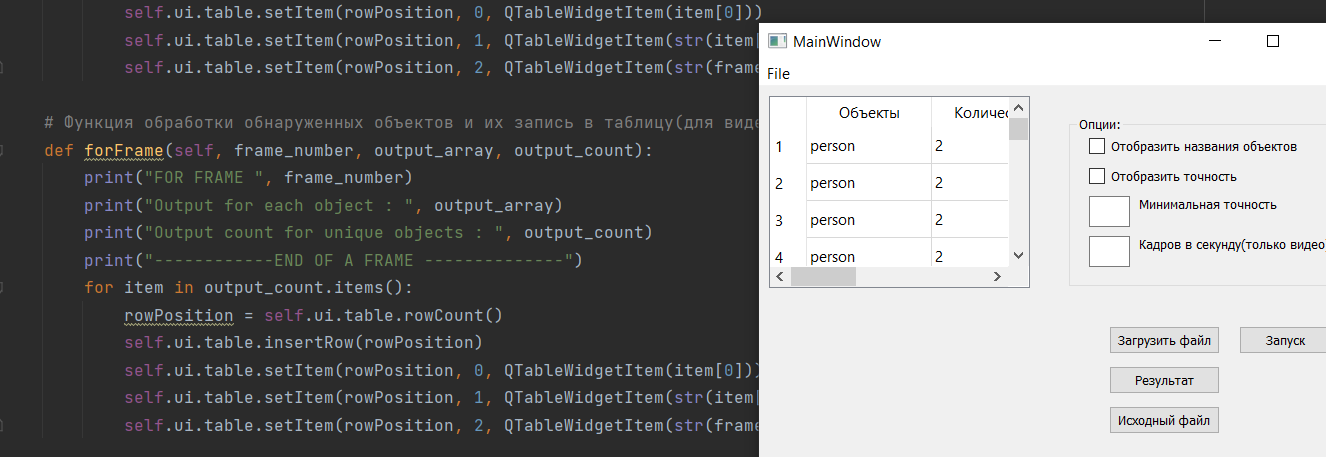
# Функция обработки обнаруженных объектов и их запись в таблицу(для видео)  
def forFrame(self,frame\_number, output\_array, output\_count):  
 print("FOR FRAME ", frame\_number)  
 print("Output for each object : ", output\_array)  
 print("Output count for unique objects : ", output\_count)  
 print("------------END OF A FRAME --------------")  
 for item in output\_count.items():  
 rowPosition = self.ui.table.rowCount()  
 self.ui.table.insertRow(rowPosition)  
 self.ui.table.setItem(rowPosition, 0, QTableWidgetItem(item[0]))  
 self.ui.table.setItem(rowPosition, 1, QTableWidgetItem(str(item[1])))  
 self.ui.table.setItem(rowPosition, 2, QTableWidgetItem(str(frame\_number)))

Запустим исходный файл и результат:





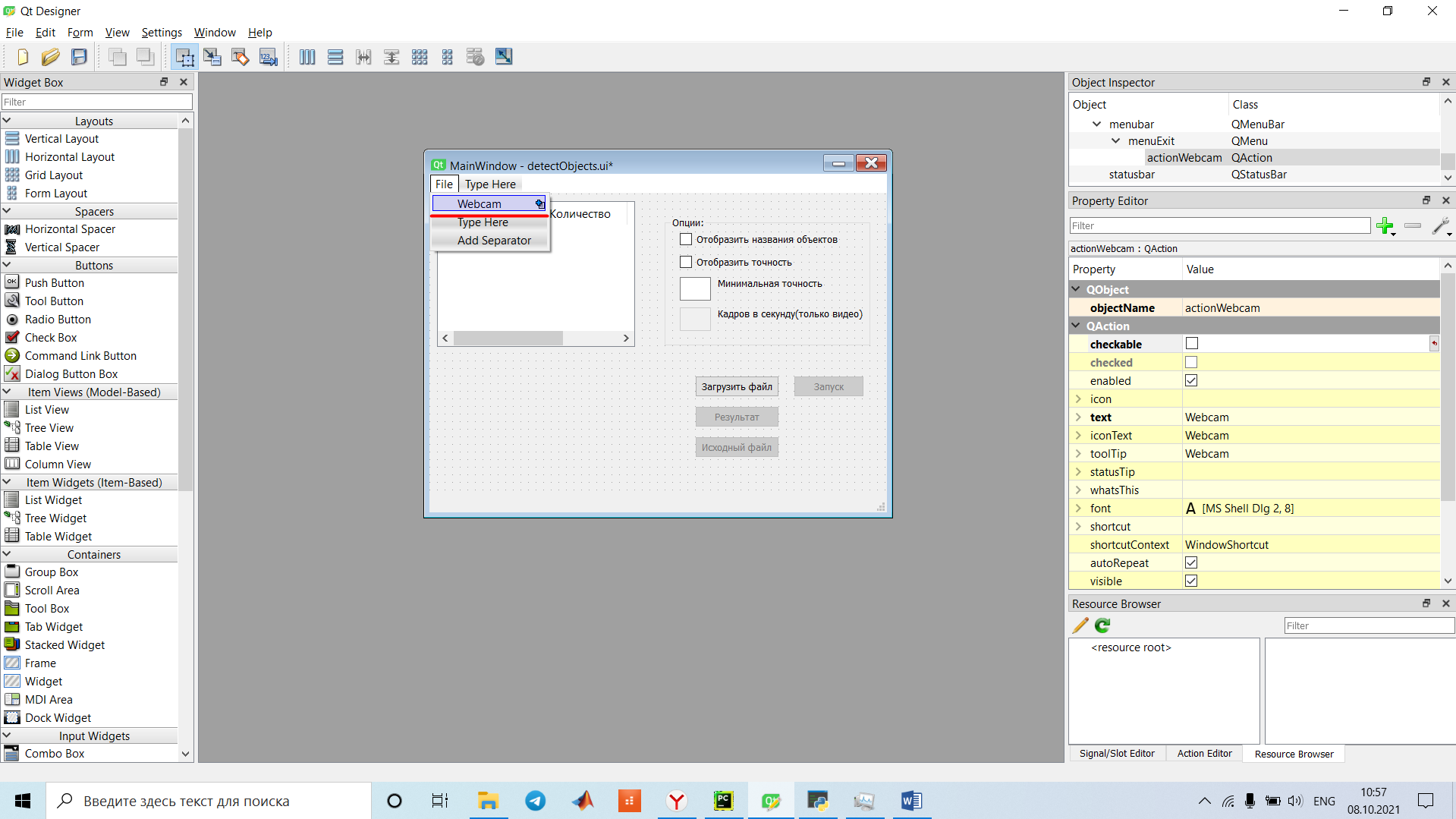
Проверим нашу таблицу:



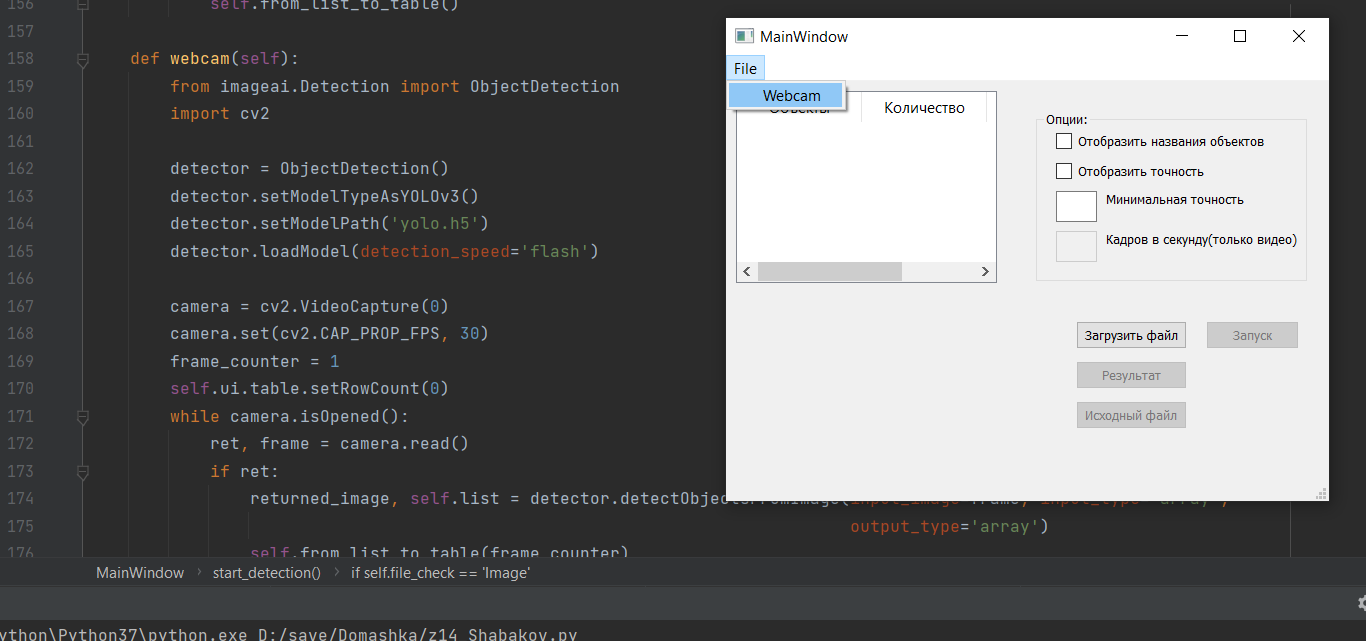
Для детального разбора функционала- см. **Код**. Для каждой функции написан комментарий

# Веб камера:

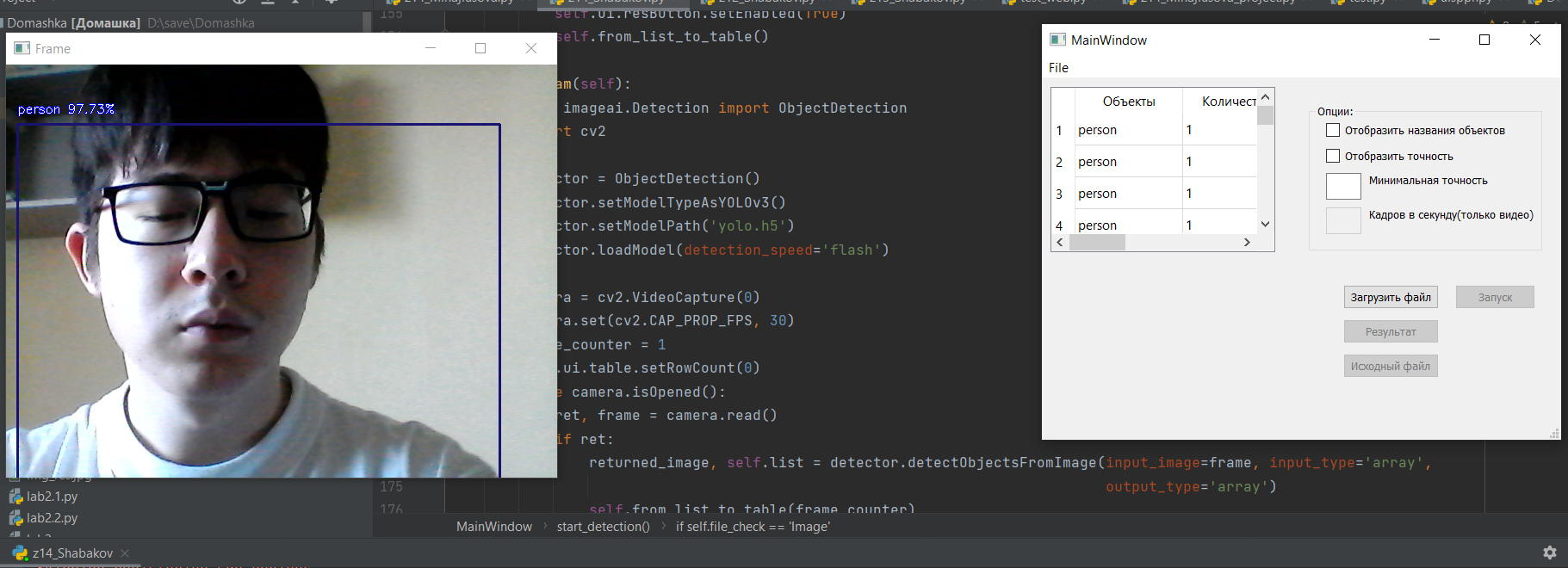
Внутри окна имеется меню, где включается распознавание объекта через веб-камеру(в реальном времени):



Как это выглядит в приложении:



Запуск:



Данные так же покадрово записываются в таблицу.

Фрагмент кода:

def webcam(self):  
 from imageai.Detection import ObjectDetection  
 import cv2  
  
 detector = ObjectDetection()  
 detector.setModelTypeAsYOLOv3()  
 detector.setModelPath('yolo.h5')  
 detector.loadModel(detection\_speed='flash')  
  
 camera = cv2.VideoCapture(0)  
 camera.set(cv2.CAP\_PROP\_FPS, 30)  
 frame\_counter = 1  
 self.ui.table.setRowCount(0)  
 while camera.isOpened():  
 ret, frame = camera.read()  
 if ret:  
 returned\_image, self.list = detector.detectObjectsFromImage(input\_image=frame, input\_type='array',  
 output\_type='array')  
 self.from\_list\_to\_table(frame\_counter)  
 frame\_counter += 1  
 cv2.imshow('Frame', returned\_image)  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
 camera.release()  
 cv2.destroyAllWindows()

# Код:

**Z14\_Shabakov.py**

import os  
import imghdr  
from PyQt5 import QtCore, QtWidgets  
import DetectObjects  
import sys  
from PyQt5.QtWidgets import \*  
  
  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self):  
 super(MainWindow, self).\_\_init\_\_()  
 self.ui = DetectObjects.Ui\_MainWindow()  
 self.ui.setupUi(self)  
 # Обработка кнопок  
 self.ui.openButton.clicked.connect(self.open\_file)  
 self.ui.sourceButton.clicked.connect(self.open\_source)  
 self.ui.startButton.clicked.connect(self.start\_detection)  
 self.ui.resButton.clicked.connect(self.open\_result)  
 self.ui.actionWebcam.triggered.connect(self.webcam)  
 # Путь к файлу  
 self.file\_path = 0  
 # Массив обнаруженных объектов  
 self.list = 0  
 # Передаваемые параметры  
 self.name = False  
 self.prob = False  
 self.minProb = 50  
 self.fps = 20  
 # Данная директория  
 self.execution\_path = os.getcwd()  
 # Проверка файл на расширение(изображение/видео)  
 self.file\_check = 0  
 self.save\_dir = 'Results'  
  
 # Функция вызова диалогового окна  
 def open\_file(self):  
 path = \  
 QtWidgets.QFileDialog.getOpenFileName(self, 'Open file', None, "Image or Video (\*.jpeg \*.jpg \*.avi \*.mp4)")[  
 0]  
 if path != '':  
 # Обнуление таблицы при загрузке нового файла  
 self.ui.table.setRowCount(0)  
 self.ui.resButton.setEnabled(False)  
 self.file\_path = path  
 # Включение кнопок Исходный файл и Запуск  
 self.ui.sourceButton.setEnabled(True)  
 self.ui.startButton.setEnabled(True)  
 self.check\_image\_or\_video()  
 # Если пользователь загрузил видео- включение редактора текста Количество кадров  
 if self.file\_check == 'Video':  
 self.ui.fps\_text.setEnabled(True)  
 else:  
 self.ui.fps\_text.setEnabled(False)  
  
 # Функция проверки типа файла  
 def check\_image\_or\_video(self):  
 image\_type = imghdr.what(self.file\_path)  
 if not image\_type:  
 self.file\_check = 'Video'  
 else:  
 self.file\_check = 'Image'  
  
 # Функция запуска исходного файла  
 def open\_source(self):  
 os.startfile(self.file\_path)  
  
 # Функция запуска полученного результата  
 def open\_result(self):  
 if self.file\_check == 'Image':  
 os.startfile(os.path.join(self.execution\_path, self.save\_dir,  
 'result' + '[' + os.path.basename(self.file\_path) + ']' + '.jpg'))  
 if self.file\_check == 'Video':  
 os.startfile(os.path.join(self.execution\_path, self.save\_dir,  
 'result' + '[' + os.path.basename(self.file\_path) + ']' + '.avi'))  
  
 # Функция проверки флажков и редакторов текста  
 def check\_checkbox(self):  
 if self.ui.name\_checkBox.isChecked():  
 self.name = True  
 if self.ui.prob\_checkBox.isChecked():  
 self.prob = True  
 if len(self.ui.minProb\_text.toPlainText()) != 0:  
 self.minProb = float(self.ui.minProb\_text.toPlainText())  
 if len(self.ui.fps\_text.toPlainText()) != 0:  
 self.fps = int(self.ui.fps\_text.toPlainText())  
  
 # Функция обработки обнаруженных объектов и их запись в таблицу(для фото)  
 def from\_list\_to\_table(self, frame\_number=1):  
 dct = {}  
 for x in self.list:  
 elem = x.get('name')  
 if elem in dct:  
 dct[elem] += 1  
 else:  
 dct[elem] = 1  
  
 for item in dct.items():  
 rowPosition = self.ui.table.rowCount()  
 self.ui.table.insertRow(rowPosition)  
 self.ui.table.setItem(rowPosition, 0, QTableWidgetItem(item[0]))  
 self.ui.table.setItem(rowPosition, 1, QTableWidgetItem(str(item[1])))  
 self.ui.table.setItem(rowPosition, 2, QTableWidgetItem(str(frame\_number)))  
  
 # Функция обработки обнаруженных объектов и их запись в таблицу(для видео)  
 def forFrame(self, frame\_number, output\_array, output\_count):  
 print("FOR FRAME ", frame\_number)  
 print("Output for each object : ", output\_array)  
 print("Output count for unique objects : ", output\_count)  
 print("------------END OF A FRAME --------------")  
 for item in output\_count.items():  
 rowPosition = self.ui.table.rowCount()  
 self.ui.table.insertRow(rowPosition)  
 self.ui.table.setItem(rowPosition, 0, QTableWidgetItem(item[0]))  
 self.ui.table.setItem(rowPosition, 1, QTableWidgetItem(str(item[1])))  
 self.ui.table.setItem(rowPosition, 2, QTableWidgetItem(str(frame\_number)))  
  
 # Функция обработки файла  
 def start\_detection(self):  
 if not os.path.exists(os.path.join(self.execution\_path, self.save\_dir)):  
 os.mkdir(os.path.join(self.execution\_path, self.save\_dir))  
 self.check\_checkbox()  
 from imageai.Detection import VideoObjectDetection, ObjectDetection  
 if self.file\_check == 'Video':  
 detector = VideoObjectDetection()  
 detector.setModelTypeAsYOLOv3()  
 detector.setModelPath("yolo.h5")  
 detector.loadModel(detection\_speed='flash')  
 self.list = detector.detectObjectsFromVideo(input\_file\_path=self.file\_path,  
 output\_file\_path=os.path.join(self.execution\_path,  
 self.save\_dir,  
 'result' + '[' + os.path.basename(  
 self.file\_path) + ']'),  
 display\_object\_name=self.name,  
 display\_percentage\_probability=self.prob,  
 minimum\_percentage\_probability=self.minProb,  
 frames\_per\_second=self.fps,  
 log\_progress=True, per\_frame\_function=self.forFrame)  
 # Включение кнопки Результат  
 self.ui.resButton.setEnabled(True)  
  
 if self.file\_check == 'Image':  
 detector = ObjectDetection()  
 detector.setModelTypeAsRetinaNet()  
 detector.setModelPath("resnet50\_coco\_best\_v2.1.0.h5")  
 detector.loadModel(detection\_speed='flash')  
 self.list = detector.detectObjectsFromImage(input\_image=self.file\_path,  
 output\_image\_path=os.path.join(self.execution\_path,  
 self.save\_dir,  
 'result' + '[' + os.path.basename(  
 self.file\_path) + ']' + '.jpg'),  
 minimum\_percentage\_probability=self.minProb,  
 display\_percentage\_probability=self.prob,  
 display\_object\_name=self.name)  
 # Включение кнопки Результат  
 self.ui.resButton.setEnabled(True)  
 self.from\_list\_to\_table()  
  
 def webcam(self):  
 from imageai.Detection import ObjectDetection  
 import cv2  
  
 detector = ObjectDetection()  
 detector.setModelTypeAsYOLOv3()  
 detector.setModelPath('yolo.h5')  
 detector.loadModel(detection\_speed='flash')  
  
 camera = cv2.VideoCapture(0)  
 camera.set(cv2.CAP\_PROP\_FPS, 30)  
 frame\_counter = 1  
 self.ui.table.setRowCount(0)  
 while camera.isOpened():  
 ret, frame = camera.read()  
 if ret:  
 returned\_image, self.list = detector.detectObjectsFromImage(input\_image=frame, input\_type='array',  
 output\_type='array')  
 self.from\_list\_to\_table(frame\_counter)  
 frame\_counter += 1  
 cv2.imshow('Frame', returned\_image)  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
 camera.release()  
 cv2.destroyAllWindows()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 form = MainWindow()  
 form.show()  
 app.exec()

**DetectObjects.py**

# -\*- coding: utf-8 -\*-  
  
# Form implementation generated from reading ui file 'detectObjects.ui'  
#  
# Created by: PyQt5 UI code generator 5.15.4  
#  
# WARNING: Any manual changes made to this file will be lost when pyuic5 is  
# run again. Do not edit this file unless you know what you are doing.  
  
  
from PyQt5 import QtCore, QtGui, QtWidgets  
  
  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 MainWindow.setObjectName("MainWindow")  
 MainWindow.resize(603, 446)  
 self.centralwidget = QtWidgets.QWidget(MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.openButton = QtWidgets.QPushButton(self.centralwidget)  
 self.openButton.setGeometry(QtCore.QRect(350, 240, 111, 28))  
 self.openButton.setObjectName("openButton")  
 self.resButton = QtWidgets.QPushButton(self.centralwidget)  
 self.resButton.setEnabled(False)  
 self.resButton.setGeometry(QtCore.QRect(350, 280, 111, 28))  
 self.resButton.setObjectName("resButton")  
 self.optionBox = QtWidgets.QGroupBox(self.centralwidget)  
 self.optionBox.setGeometry(QtCore.QRect(310, 30, 271, 171))  
 self.optionBox.setObjectName("optionBox")  
 self.name\_checkBox = QtWidgets.QCheckBox(self.optionBox)  
 self.name\_checkBox.setGeometry(QtCore.QRect(20, 20, 221, 20))  
 self.name\_checkBox.setObjectName("name\_checkBox")  
 self.prob\_checkBox = QtWidgets.QCheckBox(self.optionBox)  
 self.prob\_checkBox.setGeometry(QtCore.QRect(20, 50, 241, 20))  
 self.prob\_checkBox.setObjectName("prob\_checkBox")  
 self.minProb\_text = QtWidgets.QTextEdit(self.optionBox)  
 self.minProb\_text.setGeometry(QtCore.QRect(20, 80, 41, 31))  
 self.minProb\_text.setObjectName("minProb\_text")  
 self.label = QtWidgets.QLabel(self.optionBox)  
 self.label.setGeometry(QtCore.QRect(70, 80, 151, 16))  
 self.label.setObjectName("label")  
 self.fps\_text = QtWidgets.QTextEdit(self.optionBox)  
 self.fps\_text.setEnabled(False)  
 self.fps\_text.setGeometry(QtCore.QRect(20, 120, 41, 31))  
 self.fps\_text.setObjectName("fps\_text")  
 self.label\_2 = QtWidgets.QLabel(self.optionBox)  
 self.label\_2.setEnabled(True)  
 self.label\_2.setGeometry(QtCore.QRect(70, 120, 201, 16))  
 self.label\_2.setObjectName("label\_2")  
 self.table = QtWidgets.QTableWidget(self.centralwidget)  
 self.table.setGeometry(QtCore.QRect(10, 10, 261, 192))  
 self.table.setObjectName("table")  
 self.table.setColumnCount(3)  
 self.table.setRowCount(0)  
 item = QtWidgets.QTableWidgetItem()  
 self.table.setHorizontalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.table.setHorizontalHeaderItem(1, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.table.setHorizontalHeaderItem(2, item)  
 self.sourceButton = QtWidgets.QPushButton(self.centralwidget)  
 self.sourceButton.setEnabled(False)  
 self.sourceButton.setGeometry(QtCore.QRect(350, 320, 111, 28))  
 self.sourceButton.setObjectName("sourceButton")  
 self.startButton = QtWidgets.QPushButton(self.centralwidget)  
 self.startButton.setEnabled(False)  
 self.startButton.setGeometry(QtCore.QRect(480, 240, 93, 28))  
 self.startButton.setObjectName("startButton")  
 MainWindow.setCentralWidget(self.centralwidget)  
 self.menubar = QtWidgets.QMenuBar(MainWindow)  
 self.menubar.setGeometry(QtCore.QRect(0, 0, 603, 26))  
 self.menubar.setObjectName("menubar")  
 self.menuExit = QtWidgets.QMenu(self.menubar)  
 self.menuExit.setObjectName("menuExit")  
 MainWindow.setMenuBar(self.menubar)  
 self.statusbar = QtWidgets.QStatusBar(MainWindow)  
 self.statusbar.setObjectName("statusbar")  
 MainWindow.setStatusBar(self.statusbar)  
 self.actionOpen = QtWidgets.QAction(MainWindow)  
 self.actionOpen.setObjectName("actionOpen")  
 self.actionExit = QtWidgets.QAction(MainWindow)  
 self.actionExit.setObjectName("actionExit")  
 self.actionWebcam = QtWidgets.QAction(MainWindow)  
 self.actionWebcam.setObjectName("actionWebcam")  
 self.menuExit.addAction(self.actionWebcam)  
 self.menubar.addAction(self.menuExit.menuAction())  
  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))  
 self.openButton.setText(\_translate("MainWindow", "Загрузить файл"))  
 self.resButton.setText(\_translate("MainWindow", "Результат"))  
 self.optionBox.setTitle(\_translate("MainWindow", "Опции:"))  
 self.name\_checkBox.setText(\_translate("MainWindow", "Отобразить названия объектов"))  
 self.prob\_checkBox.setText(\_translate("MainWindow", "Отобразить точность"))  
 self.label.setText(\_translate("MainWindow", "Минимальная точность"))  
 self.label\_2.setText(\_translate("MainWindow", "Кадров в секунду(только видео)"))  
 item = self.table.horizontalHeaderItem(0)  
 item.setText(\_translate("MainWindow", "Объекты"))  
 item = self.table.horizontalHeaderItem(1)  
 item.setText(\_translate("MainWindow", "Количество"))  
 item = self.table.horizontalHeaderItem(2)  
 item.setText(\_translate("MainWindow", "Номер кадра"))  
 self.sourceButton.setText(\_translate("MainWindow", "Исходный файл"))  
 self.startButton.setText(\_translate("MainWindow", "Запуск"))  
 self.menuExit.setTitle(\_translate("MainWindow", "File"))  
 self.actionOpen.setText(\_translate("MainWindow", "Open"))  
 self.actionExit.setText(\_translate("MainWindow", "Exit"))  
 self.actionWebcam.setText(\_translate("MainWindow", "Webcam"))