КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ИНСТИТУТ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра анализа данных и исследования операций

Е.П.Шустова

ВВЕДЕНИЕ В АНАЛИЗ ИЗОБРАЖЕНИЙ НА PYTHON

П Р А К Т И К У М

ЭЛЕКТРОННЫЙ ОБРАЗОВАТЕЛЬНЫЙ РЕСУРС

Казань 2020

УДК 51-37 519.816
ББК 16.333
Ш97

Принято на заседании учебно-методической комиссии ИВМиИТ
протокол № 8 от 22 июня 2020 года

Рецензенты:
кандидат физико-математических наук,
доцент кафедры высшей математики и математического моделирования КФУ, **О.А.Широкова**,

кандидат физико-математических наук,
доцент кафедры программной инженерии Высшей школы информационных технологий и интеллектуальных систем КФУ **И.Н. Голицына**

Введение в анализ изображений на Python. Практикум. Электронный образовательный ресурс / Е.П.Шустова. – Казань: Казан. ун-т, 2020.–88 с.

Электронный образовательный ресурс удовлетворяет требованиям новых государственных образовательных стандартов высшего профессионального образования. Оно предназначено для проведения лабораторный занятий со студентами направления «Прикладная математика и информатика» по дисциплине: «Введение в анализ изображений».

Лабораторные работы составлены таким образом, что, выполнив соответствующие задания, студент овладеет навыками обработки изображений в Python; получения информации с изображения, необходимой для его анализа и принятия решения; а также приобретет опыт создания систем поддержки принятия решений в прикладных областях, где требуется анализ изображений.

ВВЕДЕНИЕ

Электронный образовательный ресурс удовлетворяет требованиям новых государственных образовательных стандартов высшего профессионального образования. Оно предназначено для проведения лабораторных занятий со студентами направления «Прикладная математика и информатика» по дисциплине: «Введение в анализ изображений».

Лабораторные работы составлены таким образом, что, выполнив соответствующие задания, студент овладеет навыками обработки изображений в Python; получения информации с изображения, необходимой для его анализа и принятия решения; а также приобретет опыт создания систем поддержки принятия решений в прикладных областях, где требуется анализ изображений.

ЛАБОРАТОРНАЯ РАБОТА 1.
УСТАНОВКА PYTHON И PYCHAM

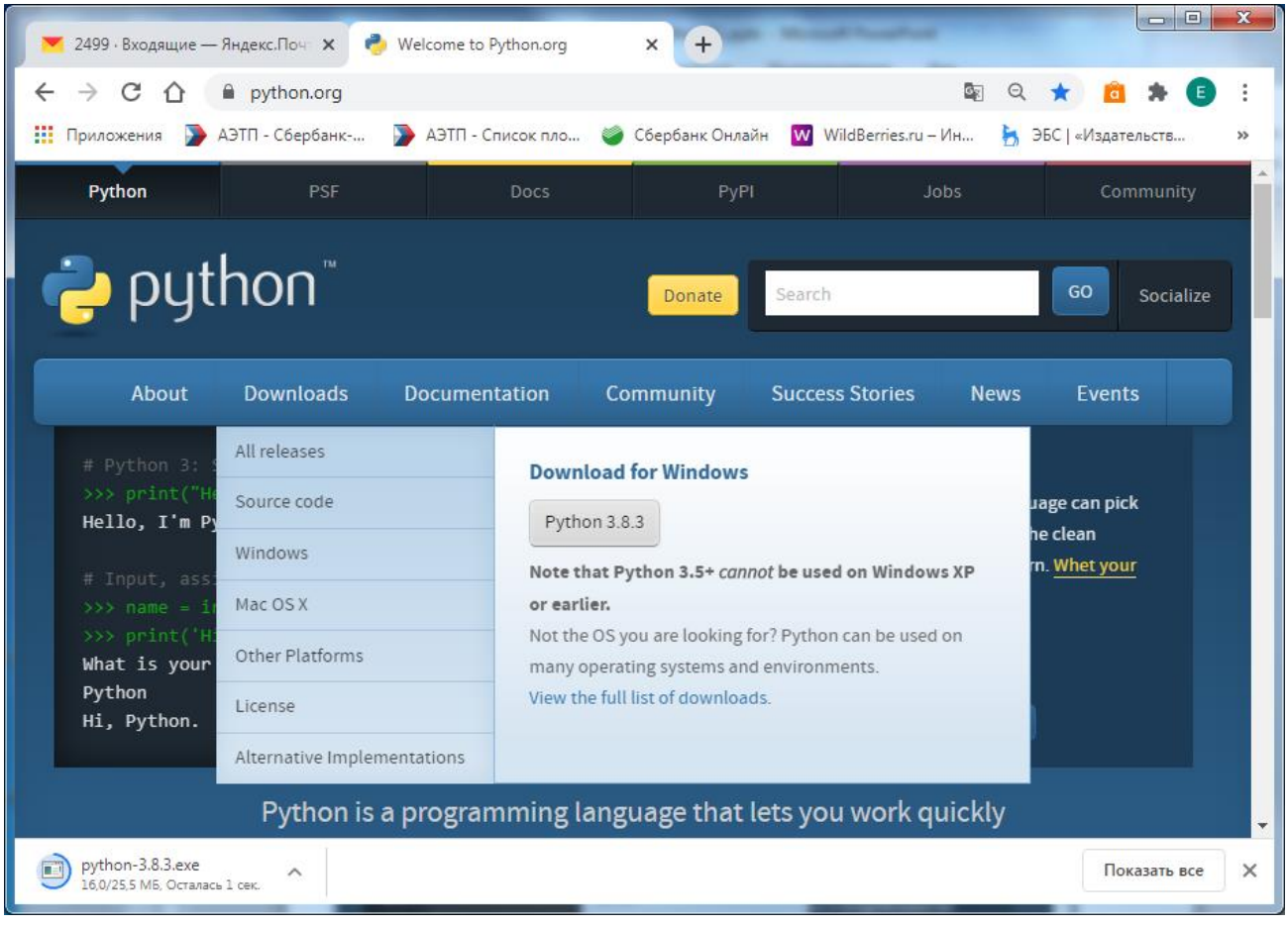
Цель работы. Научиться устанавливать **Python** и **PyCharm,** создавать новый проект, новый документ в проекте и устанавливать внешние библиотеки в данный проект в **PyCharm.**

Подготовка к выполнению заданий.

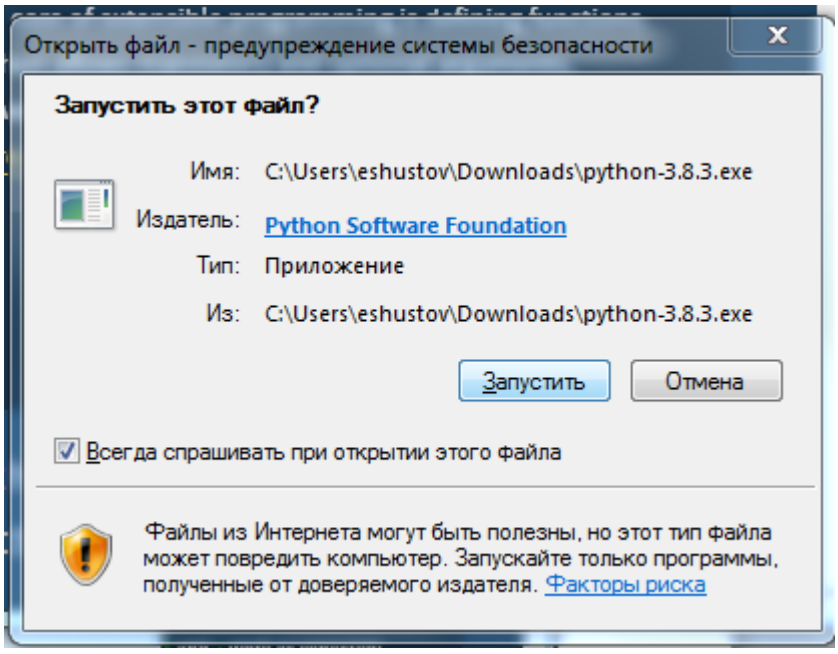
На компьютере должен быть выход в интернет.

Задание 1. Устанавка Python .

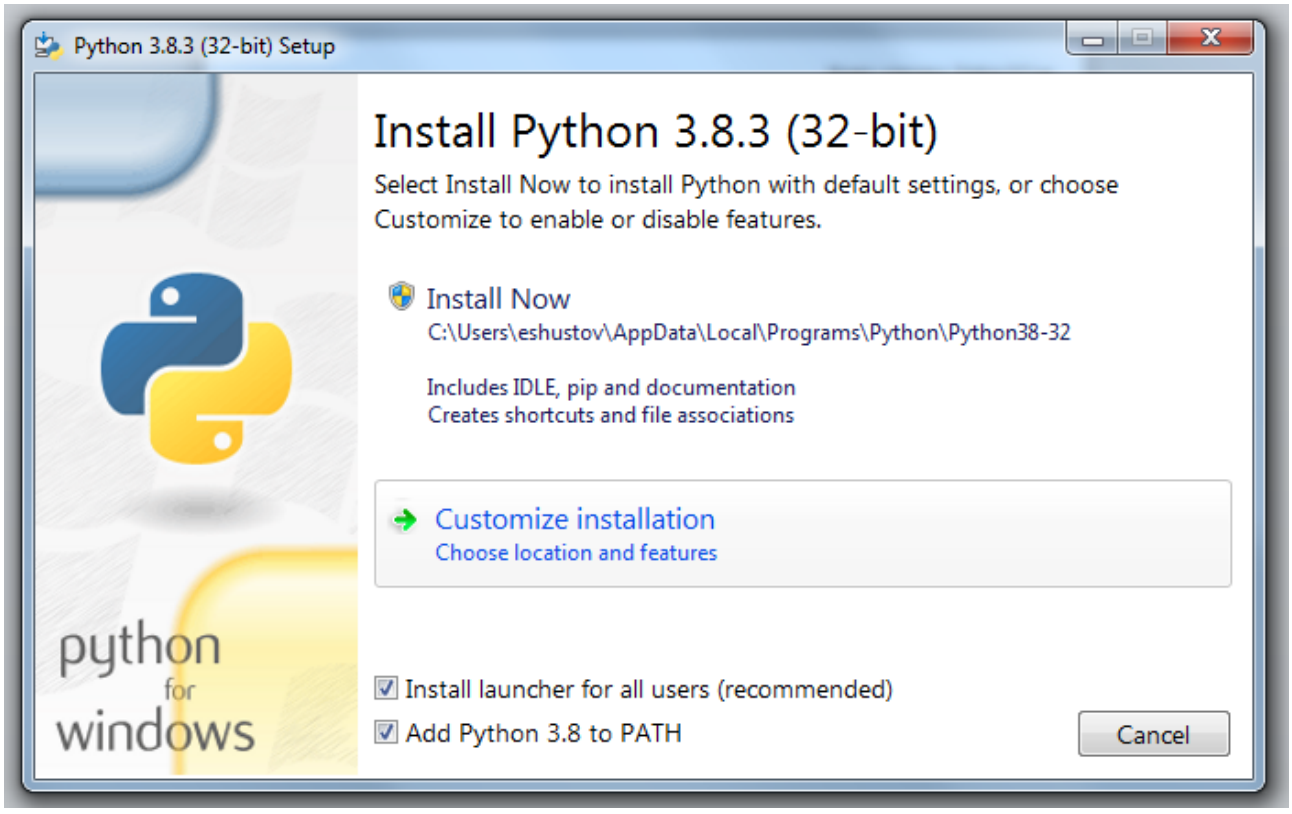
Для создания программы Вам необходимо установить **Python**. Это Вы можете сделать, воспользовавшись официальным сайтом. Здесь Вы выбираете в меню **Download** и скачиваете последнюю версию **Python** (https://www.python.org/):



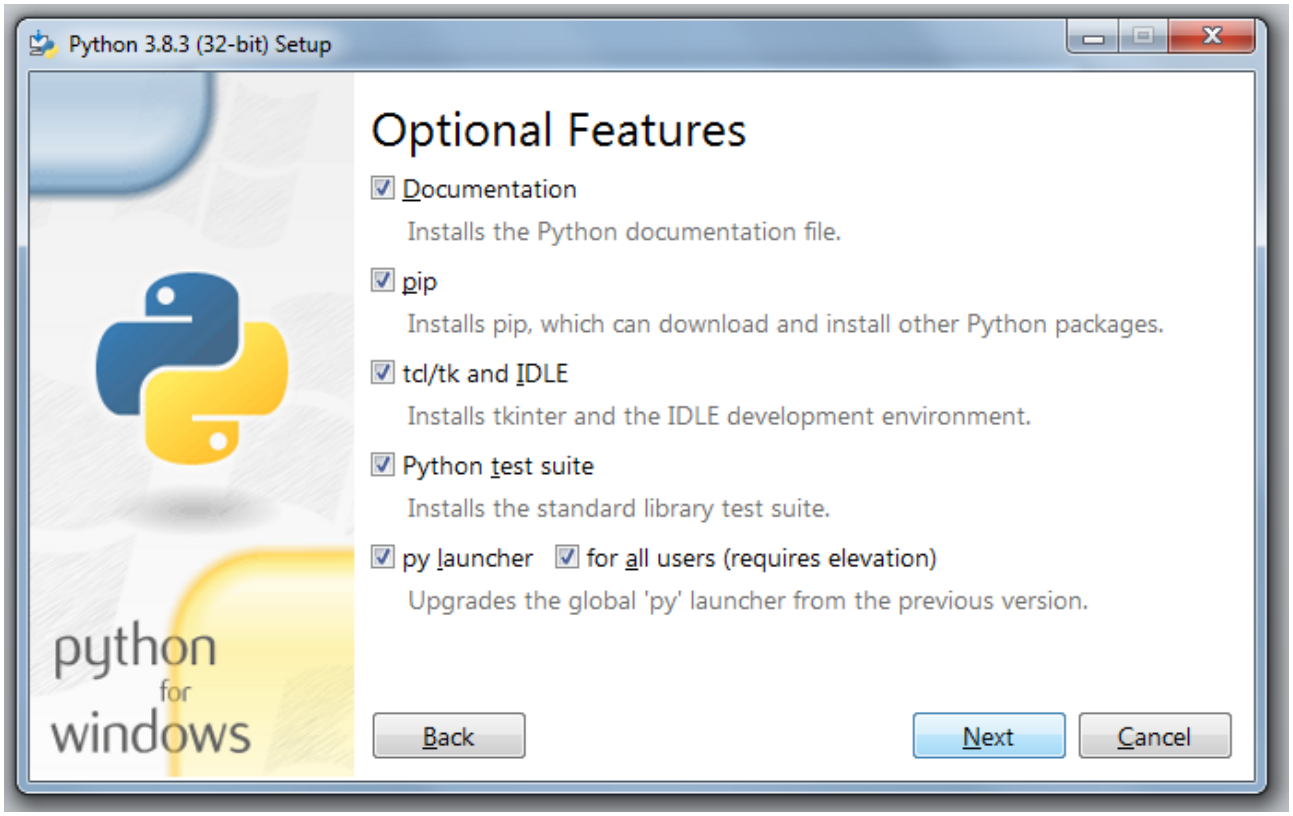
После чего запустите скаченный файл на установку, кликнув на кнопку «Запустить»:



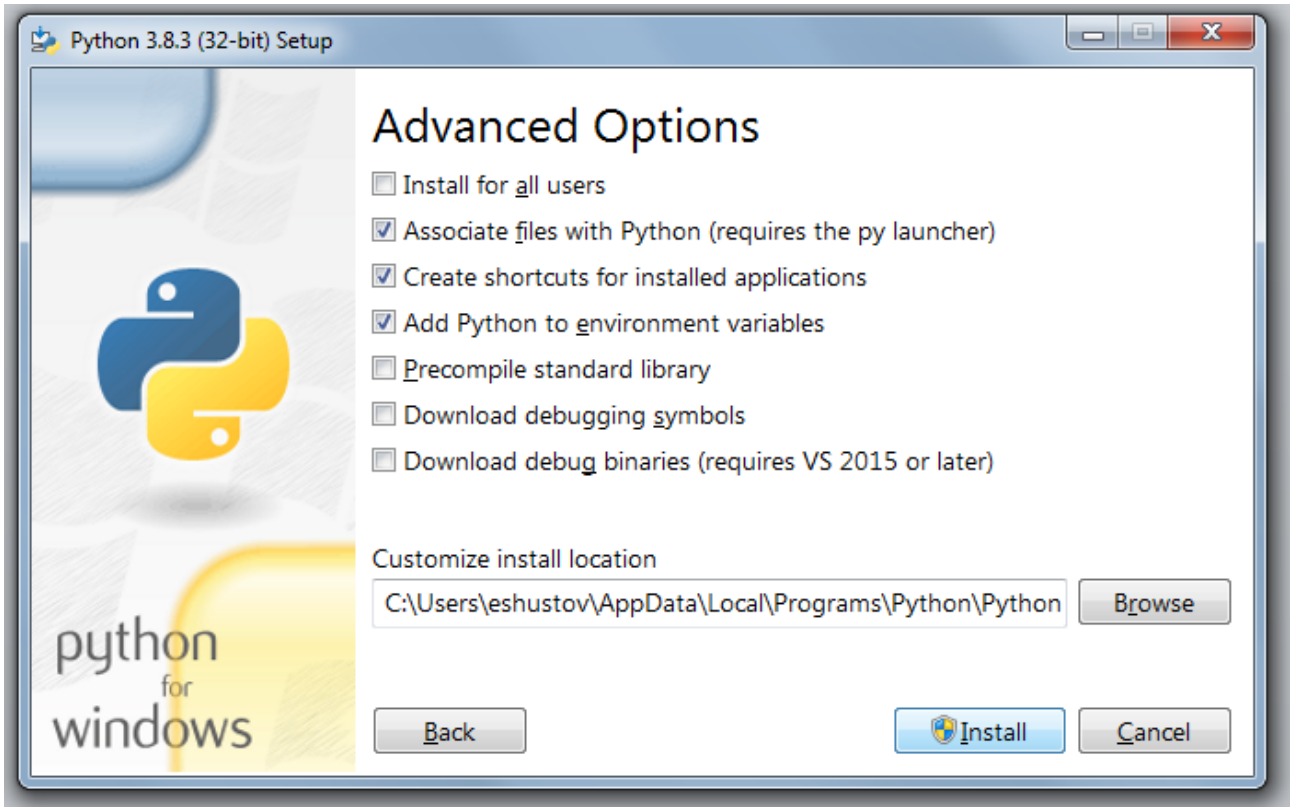
Ставим галочку рядом с “Add Python 3.8 to PATH” и выбираем ручную установку:

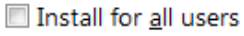


Появится форма:



Кликаем «Next». Появится форма:

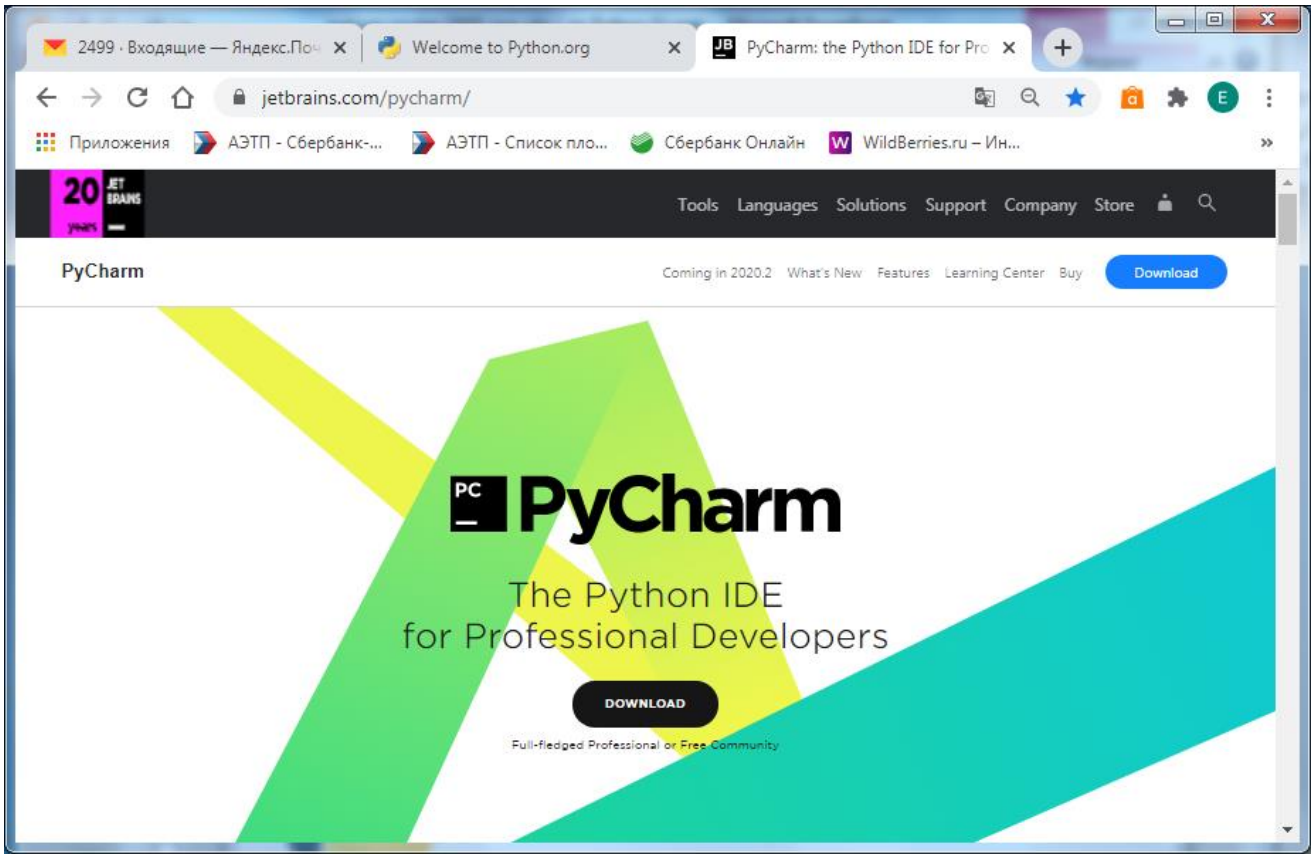


Здесь можем указать инсталяцию для всех пользователей 

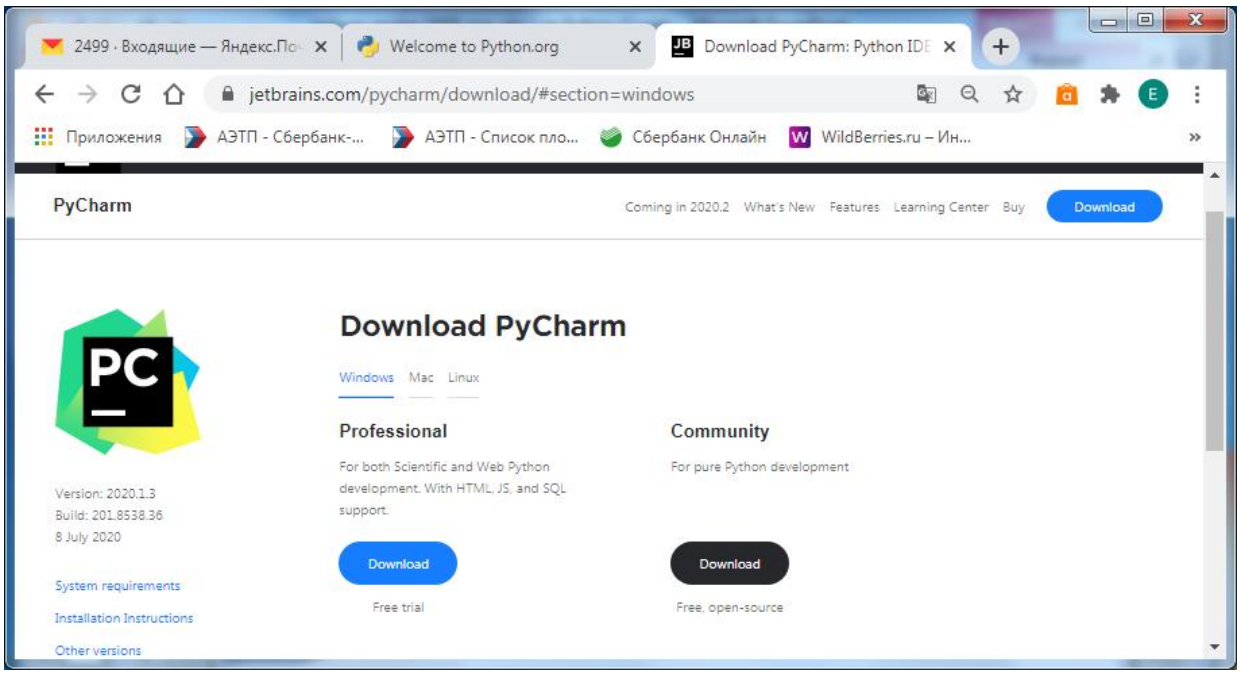
и через окно обзора указать путь, куда установить Python и кликнуть «Insnall». Но мы сейчас согласимся с предложенными системой опциями. Поэтому просто кликнем «Insnall». Запустится установка.

Задание 2. Установка `PyCharm`.

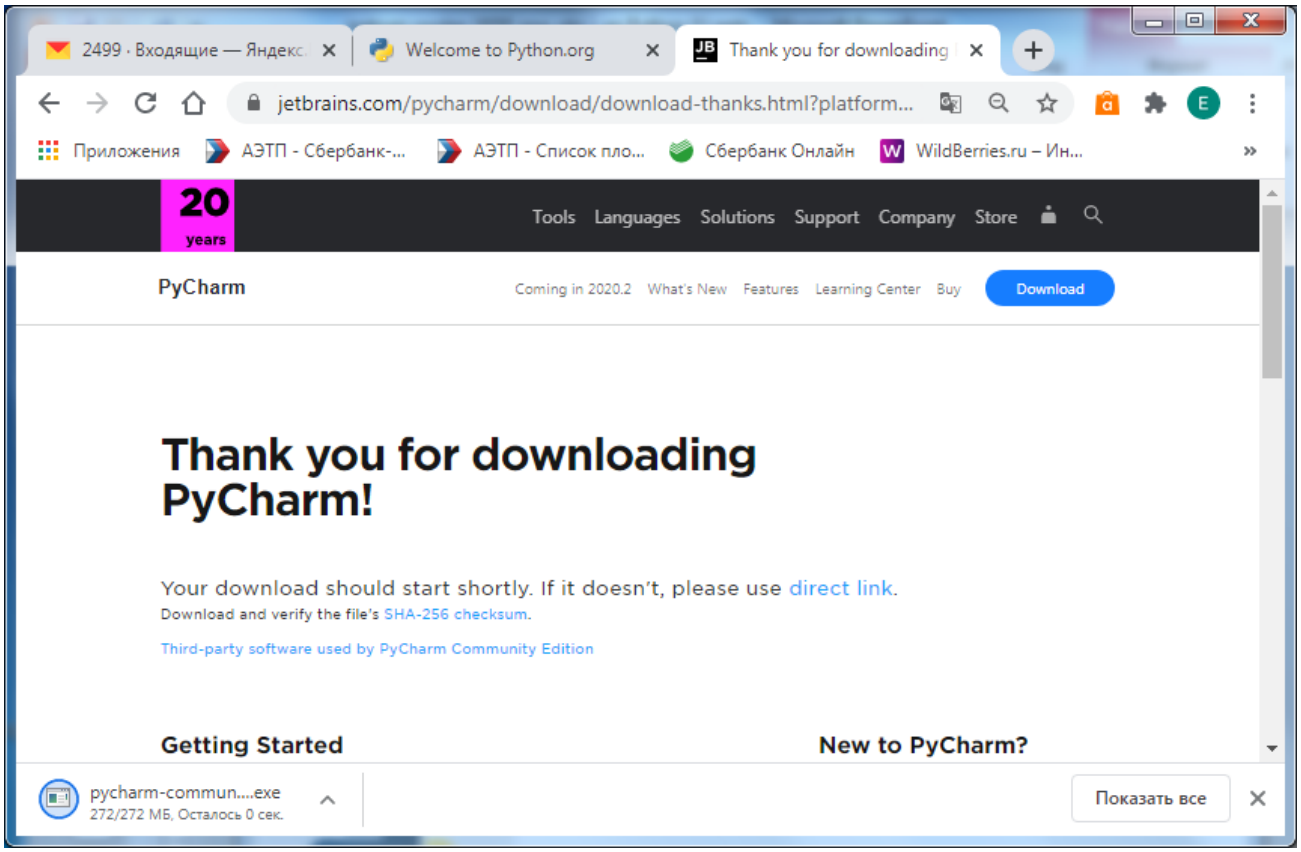
Вам потребуется интегрированная среда разработки для языка программирования Python. Мы будем использовать **`PyCharm`.** Скачайте установочный файл с официального сайта: https://www.jetbrains.com/pycharm/ и запустите его на установку.



Выберите версию Community:

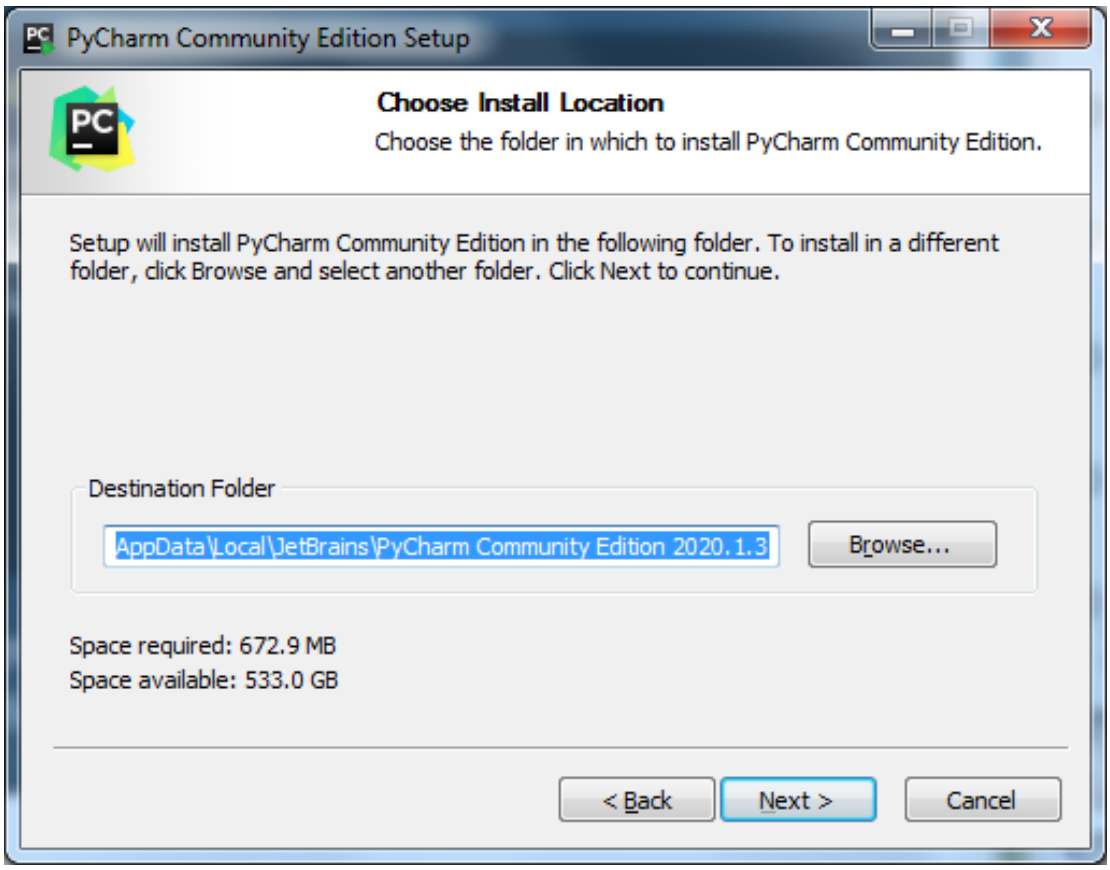


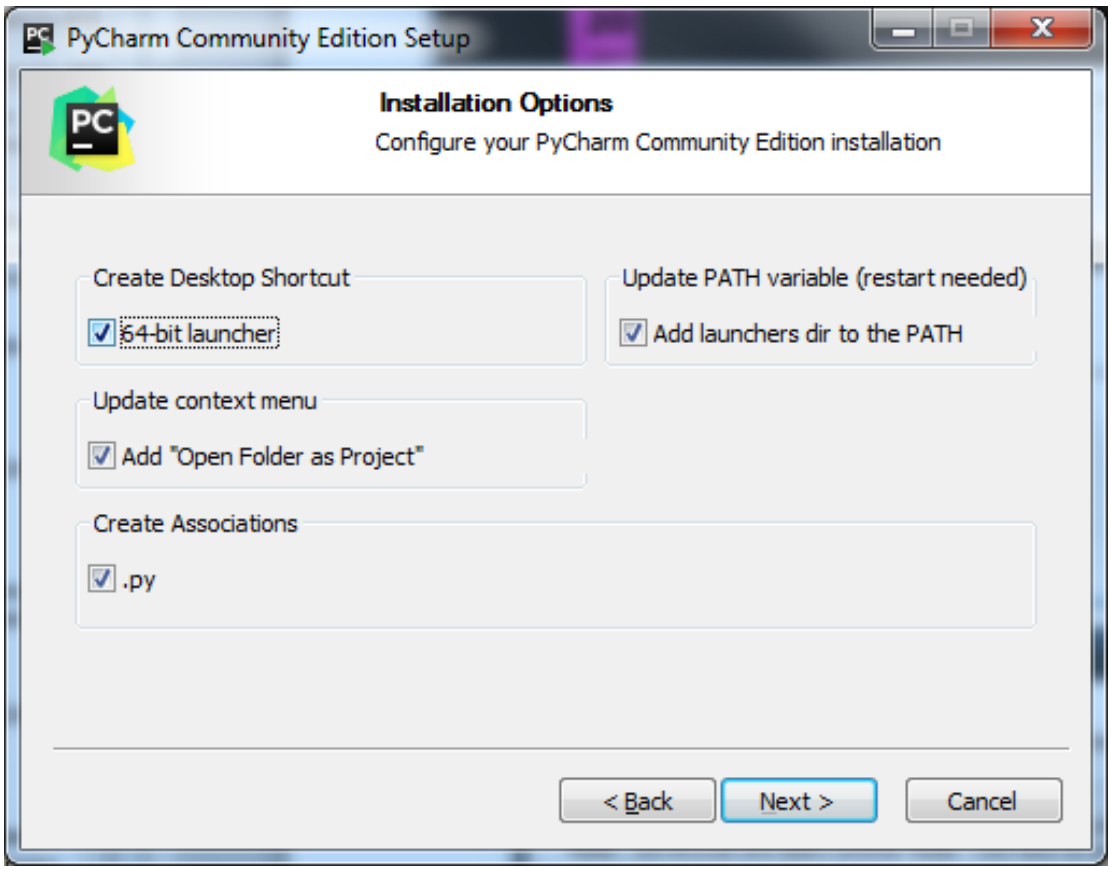
И запустите скаченный установочный файл:

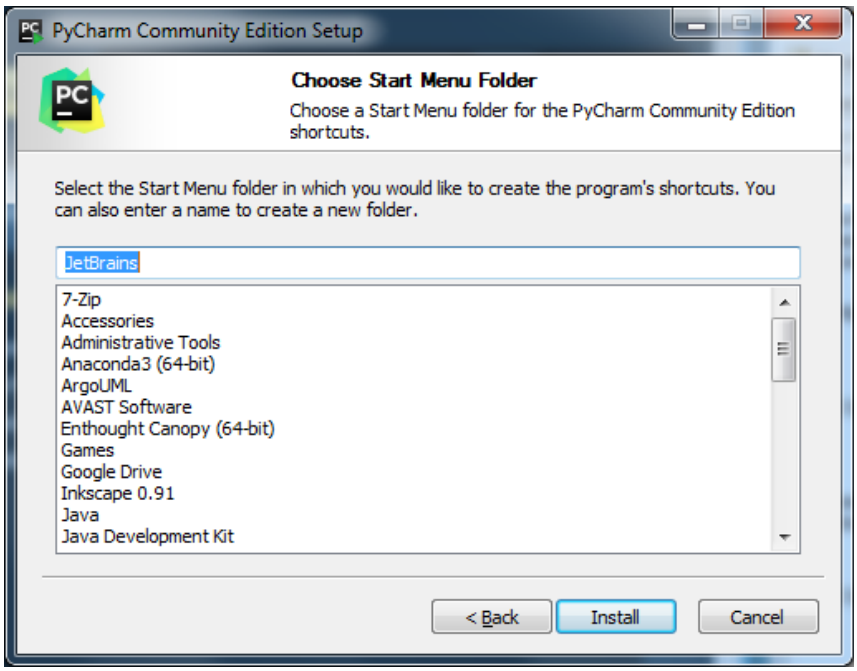


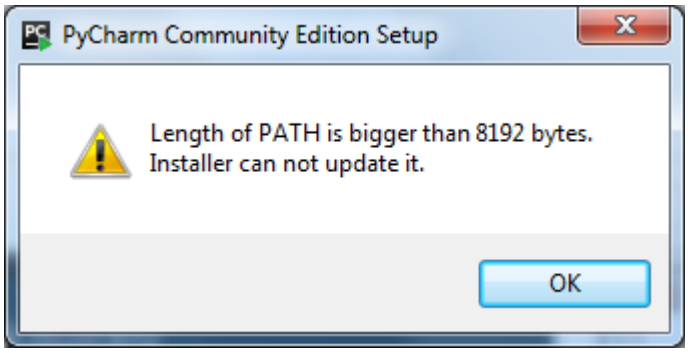
Голубым цветом везде ниже веделены на формах действия, которые Вам надо сделать на этой форме.

Далее выполните последовательно действия, указанные на скриншотах, расположенных ниже:

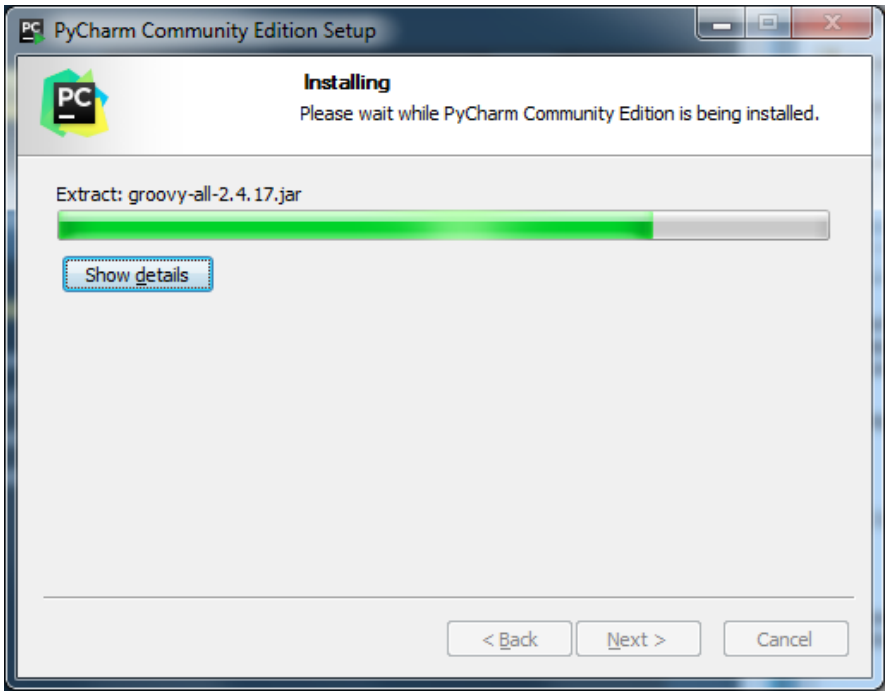

PyCharm Community Edition Setup
Choose Install Location
Choose the folder in which to install PyCharm Community Edition.
Setup will install PyCharm Community Edition in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
AppData\Local\JetBrains\PyCharm Community Edition 2020.1.3
Browse...
Space required: 672.9 MB
Space available: 533.0 GB
< Back
Next >
Cancel


PyCharm Community Edition Setup
Installation Options
Configure your PyCharm Community Edition installation
Create Desktop Shortcut
64-bit launcher
Update PATH variable (restart needed)
Add launchers dir to the PATH
Update context menu
Add "Open Folder as Project"
Create Associations
.py
< Back
Next >
Cancel

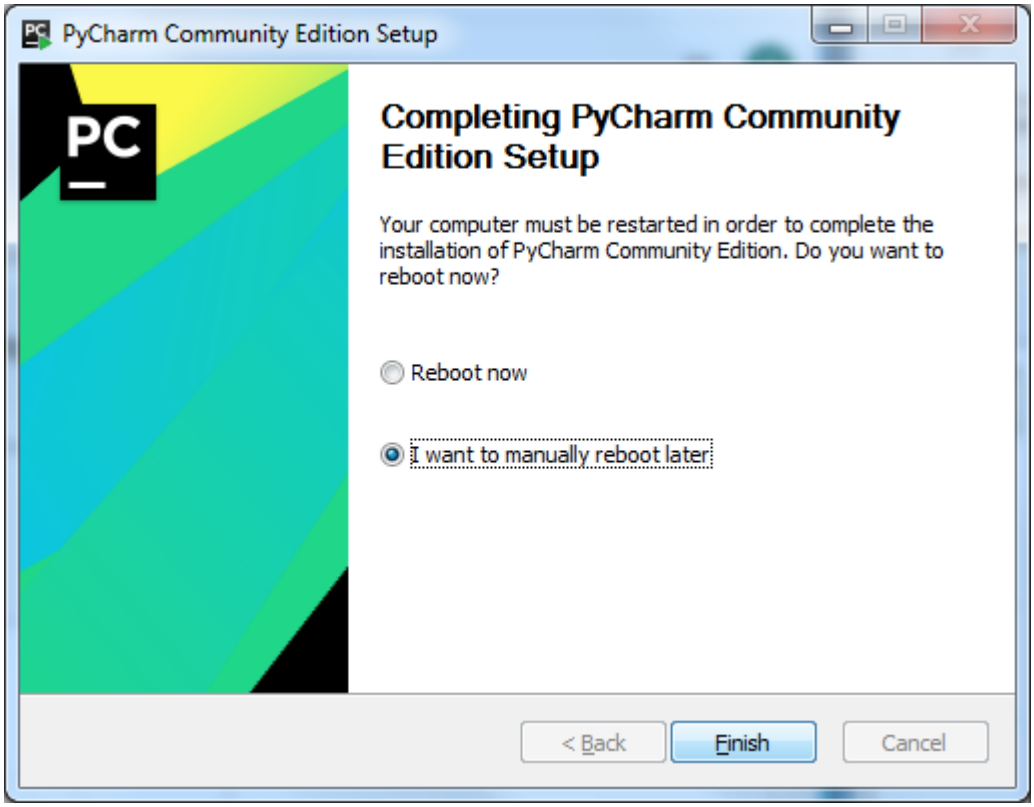




Форма, приведенная ниже. На ней не кликаем ни на какие кнопки. Подожем пока пройдет процесс :

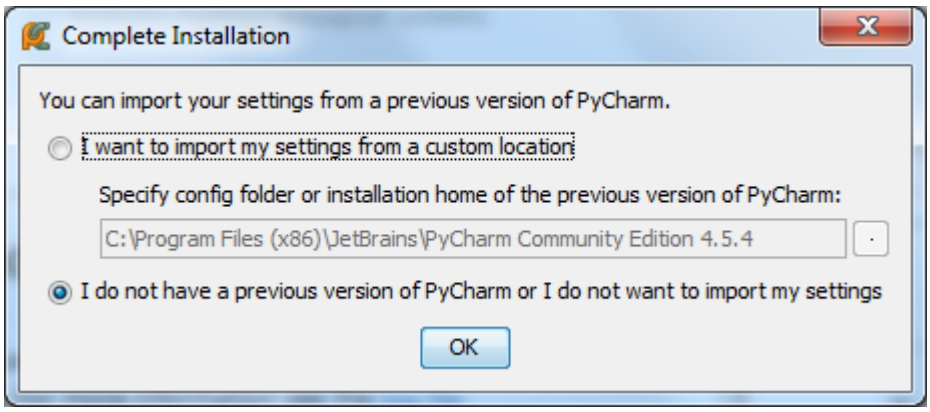


Дождемся появления приведенной ниже формы. На появившейся форме выберем перезагрузку вручную и кликнем «Finish»:



Теперь можно пользоваться PyCharm. Через проводник запускаем его:

Если у Вас на компьютере уже был ранее установлен PyCharm, то Вам первом запуске выдастся сообщение:



Выбираем «У меня нет предыдущей версии PyCharm или я не хочу импортировать мои настройки»:

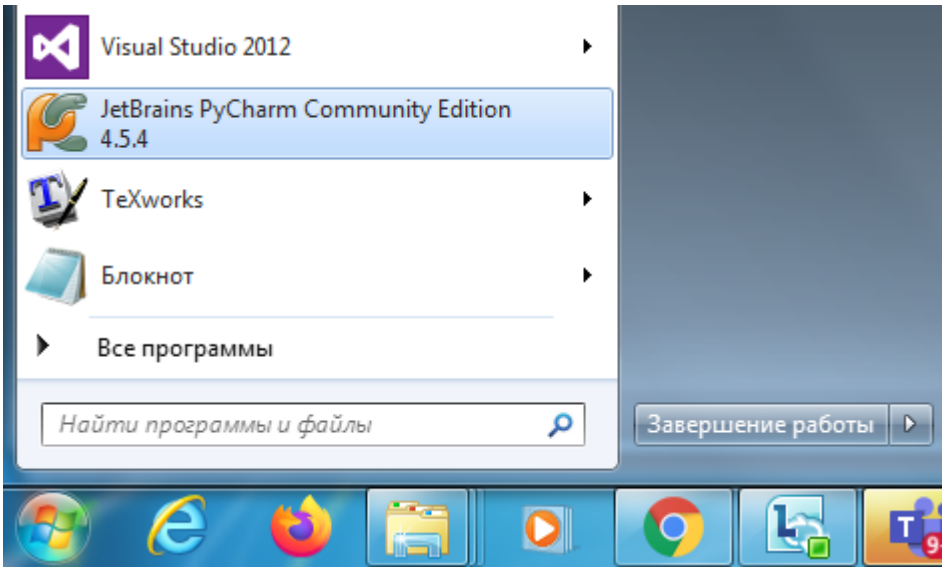
и кликаем «Ок». Появится форма главного окна дляработы в PyCharm.

В **PyCharm** уже есть пакетный менеджер **pip**. Благодаря пакетному менеджеру **pip** мы сможем устанавливать различные внешние библиотеки в нашу среду разработки.

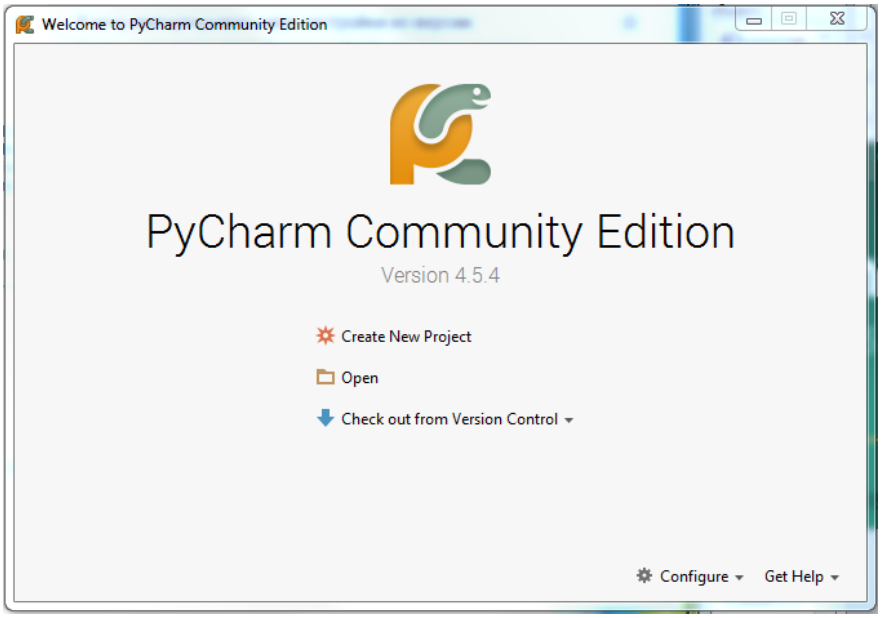
Задание 3. Создание нового проекта.

После того как Вы выполните задания 1 и 2 запустите **PyCharm** создайте новый проект. Для этого выполните следующее:

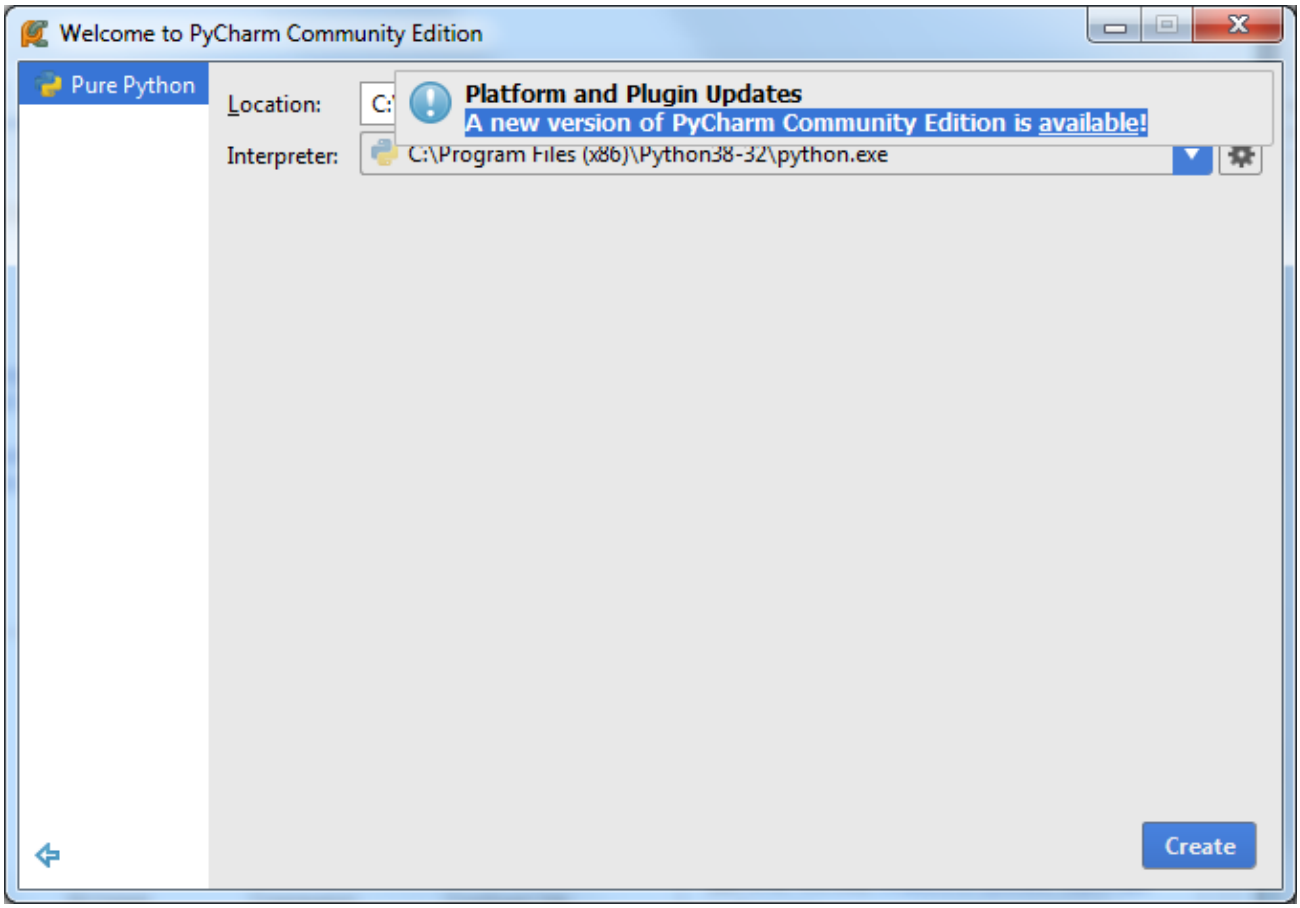
1. Через проводник запустите **PyCharm** :



2. Появится окно:

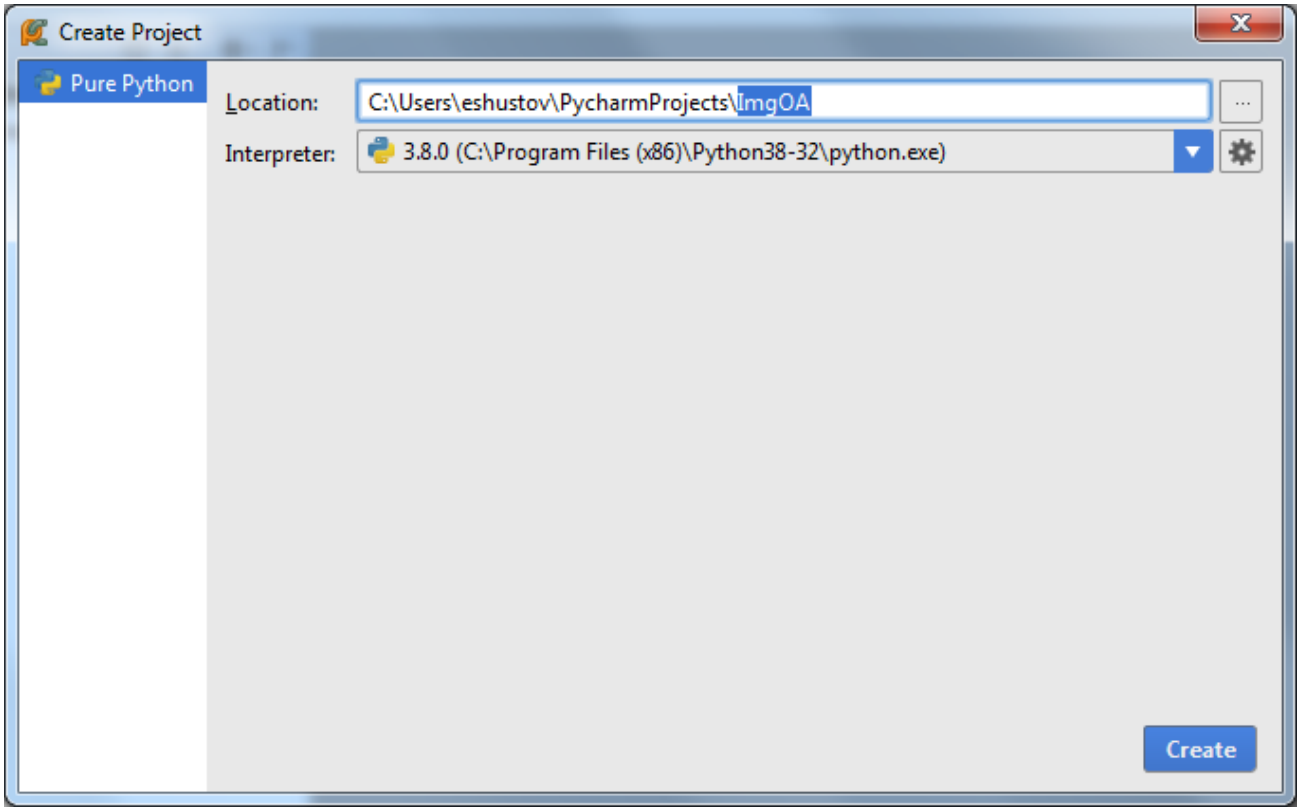


2. В правой области этого окна выберете **`Create New Project`** (создать новый проект). Появится форма с указанием локального пути плагина и интерпретатора:

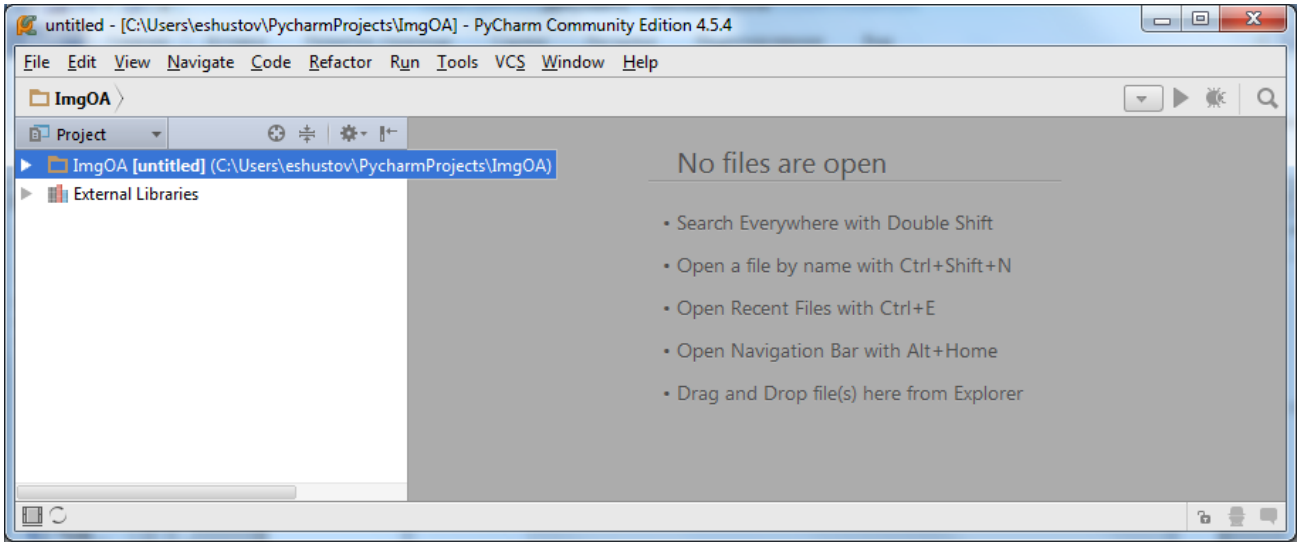


Соглашаемся и кликаем «Create»:

3. Назовите свой новый проект, например **`ImgOA:`**



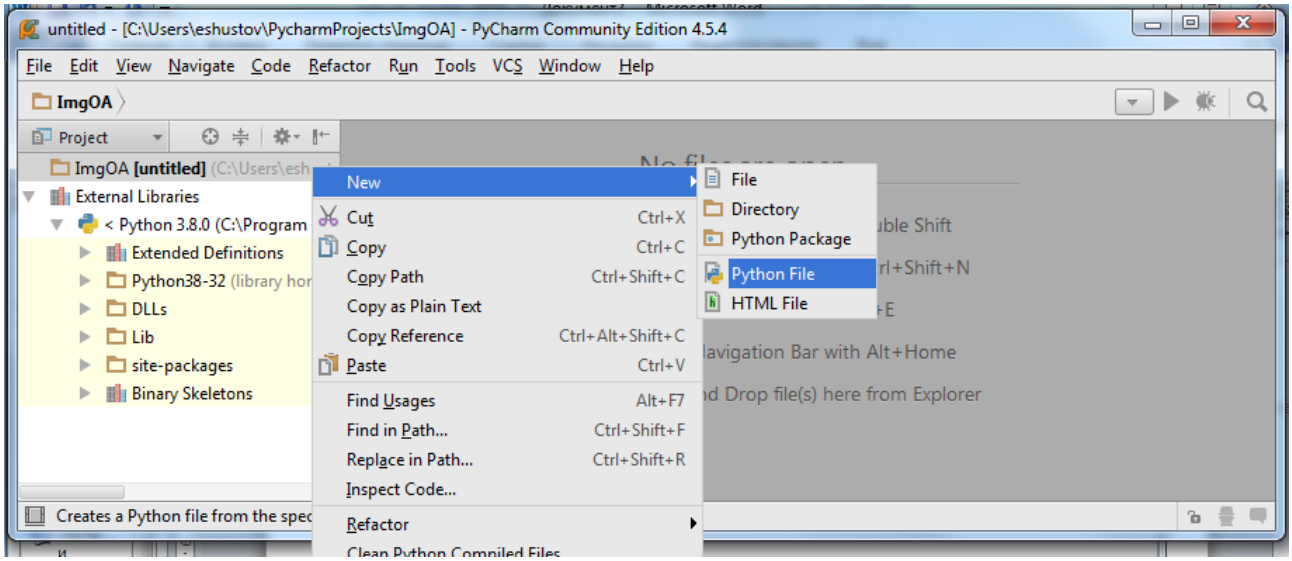
и нажмите **`Create`** (см. нижний правый угол), на приведенном выше скриншоте. Видим, что наш первый проект создался (см. на выделенное синим в левом поле на скриншоте, размещенном ниже):



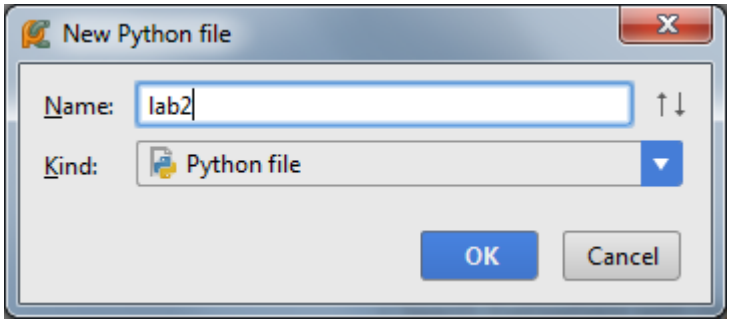
А чуть ниже видим внешние библиотеки этого проекта.

Задание 4. Создание нового документа в проекте

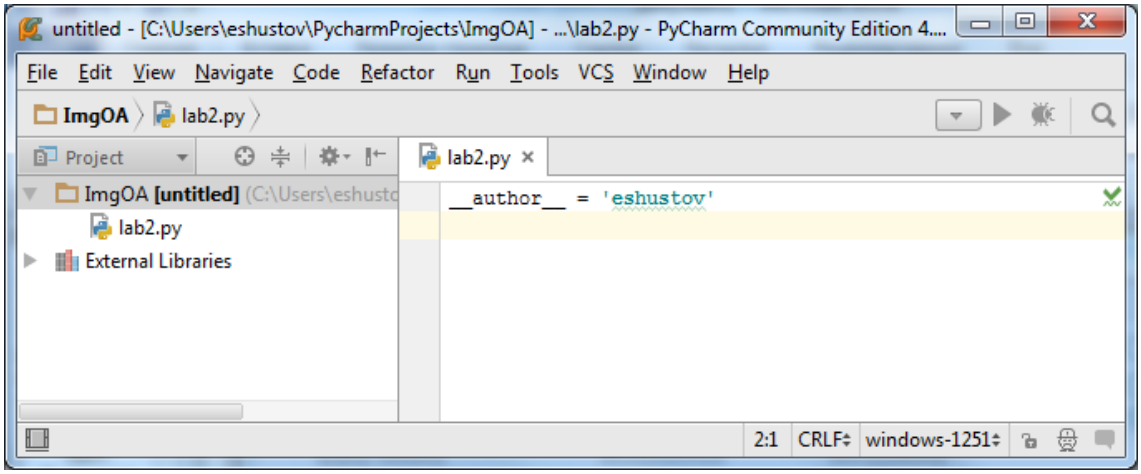
После создания проекта с именем **ImgOA** (см. предыдущий скрин) щелкните на значок , расположенный слева от **ImgOA.** Вы увидите, что на данном этапе в этом проекте нет файлов. Сздадим в этом проекте python-файл. Для этого выполним действия, указанные на скрине:



Откроется подменю, в нем вводим имя нашего python-файла:



Файл создался:

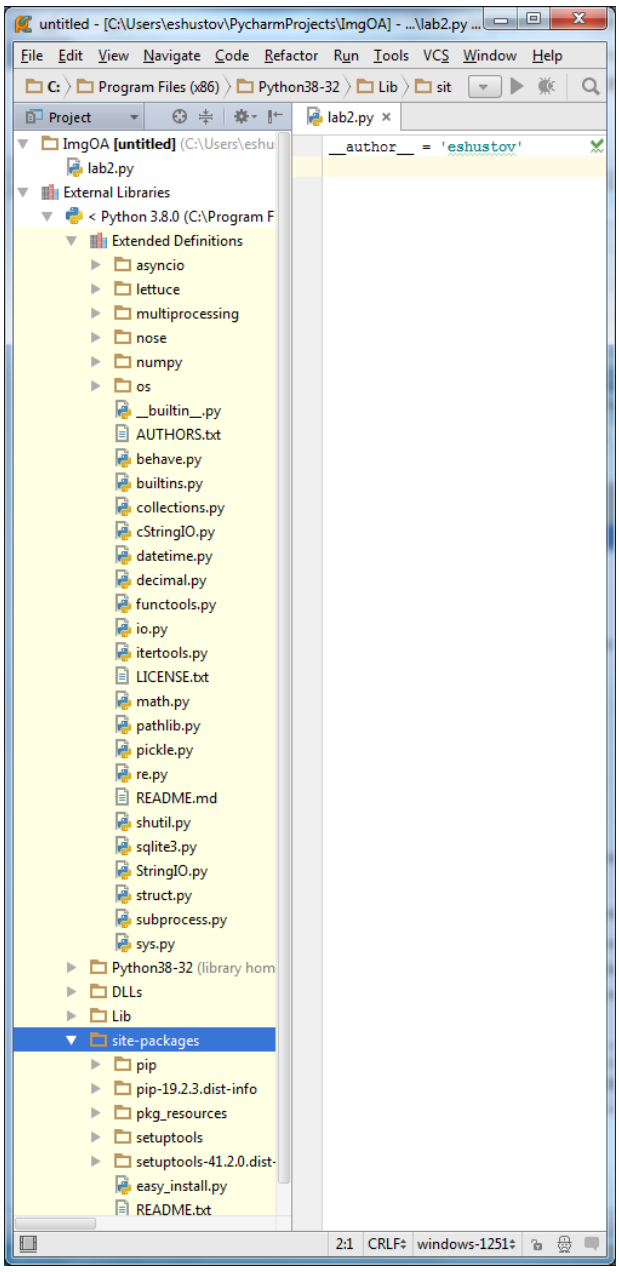


Белое место справа в окне, размещенном на рисунке выше, это место, где набираем код нашей программы.

Итак, можем приступать к созданию нашей программы. Если для решения поставленной задачи нужны какие-то библиотеки, то сначала надо выполнить установку всех необходимых библиотек в проект.

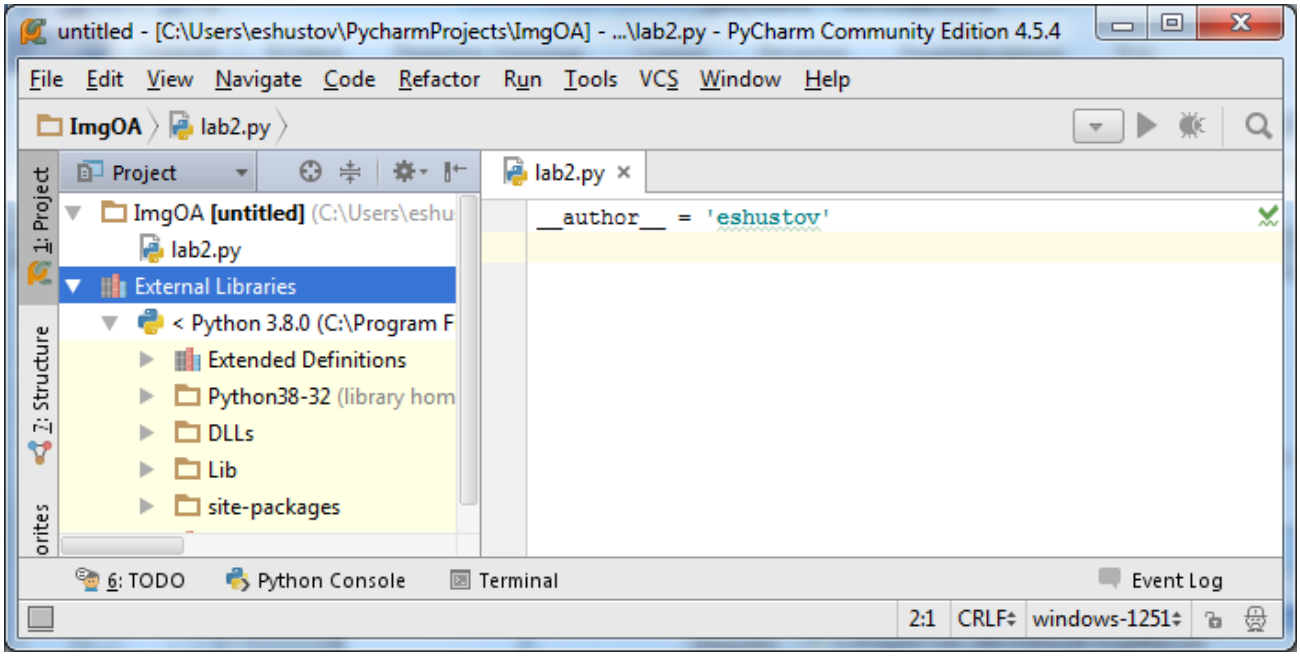
Задание 5. Установка внешних библиотек в данный проект в `PyCharm`

Сначала ознакомьтесь с теми библиотеками, которые в Вашем проекте уже присутствуют. Для этого разверните список библиотек Вашего проекта:

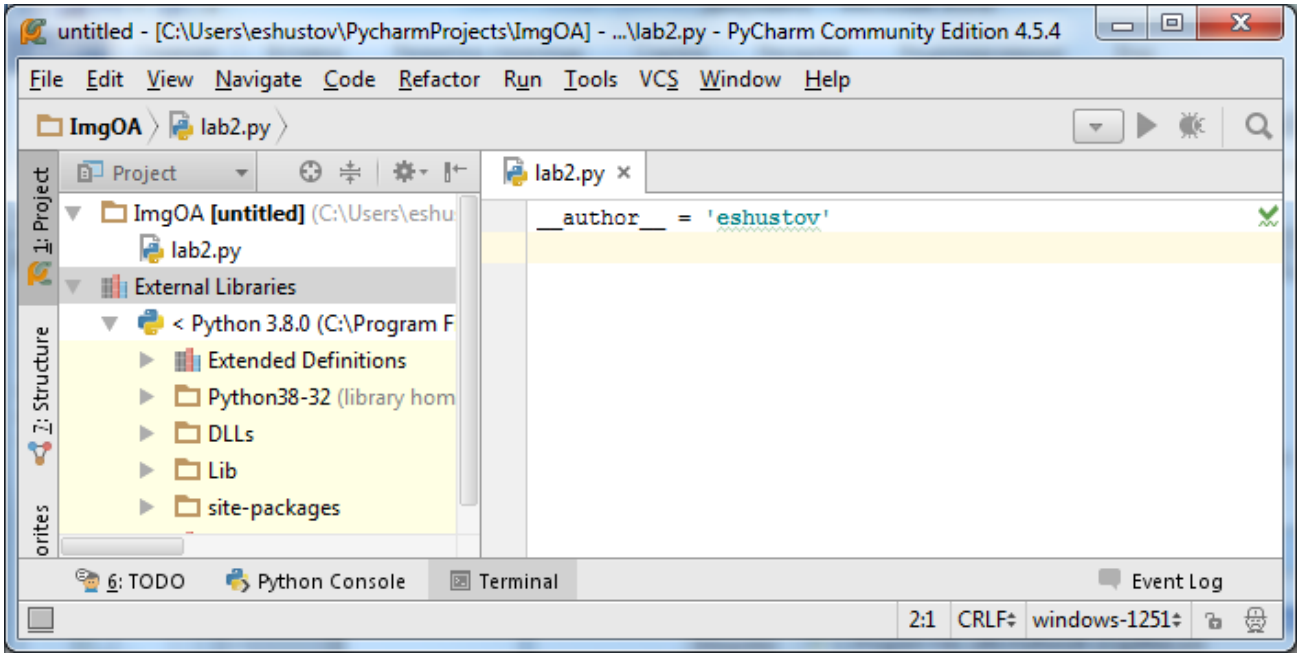


Если для Вашего проекта нужной библиотеки нет в этом списке, то необходимо установить ее в проект. Поэтому в этом задании установите внешнюю библиотеку в свой проект. Сделайте это следующим образом.

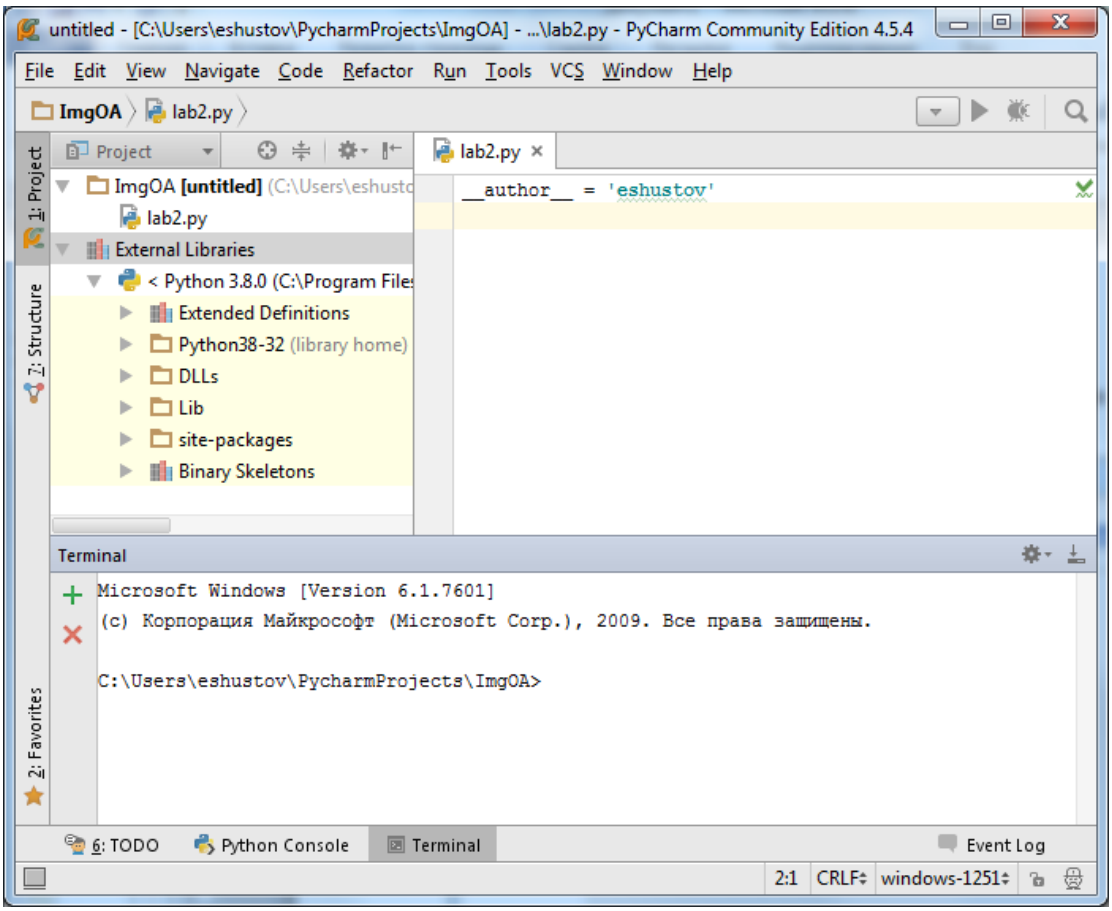
Если Вы кликните на значок  в левом нижнем углу, то в нижней области меню появится меню:



Кликните в меню внизу в Terminal:



Снизу в этом окне откроется окно терминала:



В **`Terminal`** набрать одну из команд для установки нужной внешней библиотеки (см., например, Приложение 2) и нажать клавишу **`Enter`**. Начнется загрузка этой библиотеки.

Если библиотека была уже установлена ранее, то она просто собирается в этот пакет. Она заново не скачивается, а просто добавляется в данный пакет.

Если библиотека не была ранее установлена, то она скачается, установится и добавится в данный проект.

Теперь Вы ее можете посмотреть в папочке **`Lib`**.

Папка **`lib`** – папка, которая отвечает за дополнительные библиотеки в вашем проекте.

Литература к работе

1. Официальный сайт Python, https://www.python.org
2. Официальный сайт PyCharm, https://www.jetbrains.com/pycharm

ЛАБОРАТОРНАЯ РАБОТА 2.
ЗАГРУЗКА ИЗОБРАЖЕНИЯ И ПОЛУЧЕНИЕ ИНФОРМАЦИИ О НЕМ.

Цель работы: ознакомление с основными методами загрузки изображений и получение общей информации о нем с использованием внешней библиотеки PIL.

Подготовка к выполнению заданий.

1. На компьютере должен быть выход в интернет.
2. Поместите в текущую папку набор цветных изображений.
3. Поместите не в текущую папку набор других цветных изображений.
4. Установите внешнюю библиотеку PIL (Pillow).

Задание 1. Загрузка изображения.

1. Подключите модуль Image из библиотеки PIL. Для этого напишите и запустите на выполнение команду:

   ```
   from PIL import Image
   ```

2. Откройте файл, расположенный в текущем рабочем каталоге. Для этого напишите и запустите на выполнение команду:

   ```
   img = Image.open("img1.jpg")
   # выводим изображение на экран:
   img.show()
   ```

3. Откройте этот же файл, расположенный в текущем рабочем каталоге, в бинарном режиме. Для этого напишите и запустите на выполнение команду:

   ```
   img = Image.open("Tulips.jpg")
   img = img.convert("L")
   ```

   а в более ранних, чем Release 7.2.0.dev0 Pillow выполняем команду

   ```
   img = Image.open("img1.jpg", "rb")
   # выводим изображение на экран:
   img.show()
   ```

4. Вместо указания пути к файлу можно передать файловый объект, открытый в бинарном режиме. Для этого напишите и запустите на выполнение команды:

```
# Открываем файл в бинарном режиме
f = open("img2.gif", "rb")
# Передаем объект файла
img = Image.open(f)
# выводим переданное изображение на экран:
img.show()
f.close() # Закрываем файл
```

5. Загрузка изображения из строки. Модуль StringIO .

Сначала откройте файл в бинарном режиме:

```
f = open("img2.gif", "rb")
# Сохраните изображение в переменной:
i = f.read()
f.close() # Закрываем файл
# Подключаем модуль StringIO:
import StringIO
# Передаем объект:
img = Image.open(StringIO.StringIO(i))
# выводим изображение на экран:
img.show()
```

Задание 2. Получение информации об изображении

1. Получите полную информацию об открытом изображении. Для этого выполните следующее:

```
>>> img = Image.open("foto.jpg")
>>> img.size, img.format, img.mode
((800, 600), 'JPEG', 'RGB')
>>> img.info
{'jfif': 258, 'jfif_unit': 0, 'adobe': 100, 'progression': 1,
'jfif_version': (1, 2), 'adobe_transform': 100,
'jfif_density': (100, 100)}
```

2. Получите информацию о цветовой модели изображения.

```
>>> img = Image.open("foto.jpg")
>>> img.mode
'RGB'
```

3. Получите информацию о размере изображения.

img.size

В консоль выведется размер повернутого на 90 градусов изображения:

(600, 800)

4. Получите координаты прямоугольной области, в которую вписывается всё изображение **img .**

img.getbbox()

(0, 0, 800, 600)

Литература к работе

1. Python Imaging Library (Fork), https://pypi.org/project/Pillow/
2. Прохоренок, Н. А. Python 3. Самое необходимое: Пособие / Прохоренок Н.А., Дронов В.А. - СПб:БХВ-Петербург, 2016. - 464 с. ISBN 978-5-9775-3631-8. - Текст : электронный. - URL: https://codernet.ru/books/python/python_3_samoe_neobxodimoe_proxorenok/
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.
4. Alex Clark Pillow (PIL Fork) Documentation, Release 7.2.0.dev0, Apr 21, 2020, https://readthedocs.org/projects/pillow/downloads/pdf/latest/

ЛАБОРАТОРНАЯ РАБОТА 3.
ЦВЕТОВЫЕ МОДЕЛИ.
СОЗДАНИЕ И СОХРАНЕНИЕ ИЗОБРАЖЕНИЯ.

Цель работы: ознакомление со способами создания и сохранения изображения, а так же с цветовыми моделями в Python с использованием внешней библиотеки PIL.

Подготовка к выполнению заданий.

1. На компьютере должен быть выход в интернет.
2. Поместите в текущую папку набор цветных изображений.
3. Поместите не в текущую папку набор других цветных изображений.
4. Установите внешнюю библиотеку PIL.

Задание 1. Получение и изменение цвета пиксела.

1. Получите и измените цвет пиксела методом **`load()`**.

   Сначала получите информацию о цветовой модели открытого изображения:

```
>>> img = Image.open("foto.jpg")
>>> img.mode
'RGB'
```

   Теперь получите цвет пиксела:

```
obj = img.load()
obj[25, 45]  # Получаем цвет пикселя
(122, 86, 62)
# Задаем цвет пикселя (красный)
obj[25, 45] = (255, 0, 0)
img.show()  # Просматриваем изображение (цвет пикселя будет изменен на obj[25, 45], заданный выше.
```

2. Получите и измените цвет пиксела методами **gel()** и **putpixel().**

```
# Получаем цвет пикселя
img.getpixel((25, 45))
(255, 0, 0)
# Изменяем цвет пикселя
img.putpixel((25, 45), (0, 0, 255))
# Получаем цвет пикселя
img.getpixel((25, 45))
(0, 0, 255)
img.show() # Просматриваем изображение
```

Задание 2. Переход от одной цветовой модели изображения к другой

1. Преобразуйте изображение из режима **RGB в RGBA, используя split() и merge .**

split() – возвращает каналы изображения в виде кортежа.

merge(<Режим>, <Каналы>) – собирает изображение из каналов (операция, обратная **split).**

```
>>> img = Image.open("foto.jpg")
>>> img.mode
'RGB'
>>> R, G, B = img.split()
>>> mask = Image.new("L", img.size, 128)
>>> img2 = Image.merge("RGBA", (R, G, B, mask) )
>>> img2.mode
'RGBA'
>>> img2.show()
```

2. Преобразуйте изображение из режима **RGB в RGBA, используя convert().**

```
img = Image.open("foto.jpg")
img.mode
'RGB'
img2 = img.convert("RGBA")
img2.mode
'RGBA'
img2.show()
```

#прочитать изображение и конвертировать

его в градациях серого:

```
pil_im = Image.open('empire.jpg').convert('L')
```

3. Преобразуйте изображение **RGB** в формат **P** , указав смешивание цветов и адаптивную палитру в 128 цветов.

```
>>> img = Image.open("foto.jpg")
>>> img.mode
'RGB'
>>> img2 = img.convert("P", None,
Image.FLOYDSTEINBERG, Image.ADAPTIVE, 128)
>>> img2.mode
'P'
```

Задание 3. Сохранение изображения.

Сохраните изображение в текущую папку.

```
# Сохранение изображения в текущую папку в формате JPEG:
img.save("tmp.jpg")
# в формате BMP:
img.save("tmp.bmp", "BMP")
f = open("tmp2.bmp", "wb")
# Передаем файловый объект
img.save(f, "BMP")
f.close()
```

Задание 4. Создание нового изображения.

1. Создайте черный квадрат.

```
img = Image.new("RGB", (100, 100))
img.show() # Черный квадрат
```

2. Создайте красный квадрат.

```
img = Image.new("RGB", (100, 100), (255, 0, 0))
img.show() # Красный квадрат
```

3. Создайте Зеленый квадрат.

```
img = Image.new("RGB", (100, 100), "green")
img.show() # Зеленый квадрат
```

4. Создайте красный квадрат.

```
img = Image.new("RGB", (100, 100), "#f00")
img.show() # тоже красный квадрат
```

5. Создайте белый квадрат.

```
img = Image.new("RGB", (100, 100), "white")
img.show() # Белый квадрат
```

6. Создайте серый квадрат.

```
img = Image.new("RGB", (320, 240), "silver")
img.show() # Серый квадрат
```

7. Создайте цветной прямоугольник.

```
img = Image.new("RGB", (320, 240), "rgb(205,
100,200)")
img.show() # цветной прямоугольник
```

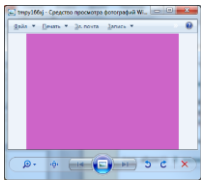
8. Создайте цветной прямоугольник (Каналы в процентах).

```
img = Image.new("RGB", (320, 240), "rgb(10%,
100%,40%)")
img.show() # цветной прямоугольник. Каналы в процентах
```

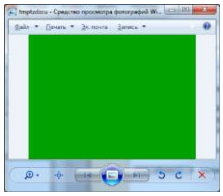
9. Создайте цветной прямоугольник заданного цвета. Затем поменяйте этот

цвет.

```
img = Image.new("RGB", (640, 480), "rgb(205,
100,200)")
img.show() # сиреневый прямоугольник
```



```
for x in xrange(640):
    for y in xrange(480):
        img.putpixel((x,y),(0,160,0))
img.save("okno.png", "PNG")
img.show() # зеленый прямоугольник
```



10. Создайте прямоугольник заданного цвета (функциональная раскраска).

```
img = Image.new("RGB", (640, 480), "rgb(205,
100,200)")
img.show() # сиреневый прямоугольник
for x in xrange(640):
    for y in xrange(480):
        img.putpixel((x,y),
        x/3,(x+y)/6,y/3))
img.save("okno.png", "PNG")
img.show() # прямоугольник с функциональной раскраской
```

Литература к работе

1. Python Imaging Library (Fork), https://pypi.org/project/Pillow/
2. Прохоренок, Н. А. Python 3. Самое необходимое: Пособие / Прохоренок Н.А., Дронов В.А. - СПб:БХВ-Петербург, 2016. - 464 с. ISBN 978-5-9775-3631-8. - Текст : электронный. - URL: https://codernet.ru/books/python/python_3_samoe_neobxodimoe_proxorenok/
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

ЛАБОРАТОРНАЯ РАБОТА 4.
КАНАЛЫ ЦИФРОВОГО ИЗОБРАЖЕНИЯ.
ГИСТОГРАММА ИЗОБРАЖЕНИЯ.

Цель работы: ознакомление со способами извлечения каналов из цифрового изображения и сборки изображения из заданных каналов, а так же научиться строить гистограмму изображения в Python с использованием внешней библиотеки PIL.

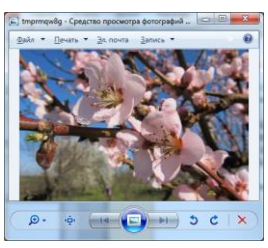
Подготовка к выполнению заданий.

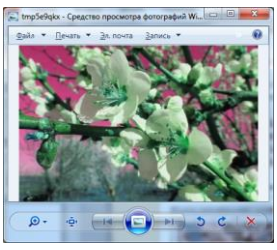
1. Поместите в текущую папку набор цветных изображений и изображений в оттенках серого.
2. Поместите не в текущую папку набор других цветных изображений и изображений в оттенках серого.
3. На компьютере должен быть выход в интернет.
4. Установите внешнюю библиотеку PIL.

Задание 1. Замена каналов. Методы Getpixel, putpixel

Сначала откройте изображение. Затем поменяйте каналы в этом изображении. Для этого выполните следующее:

```
img = Image.open("tsveti.jpg")
img.show()  # цветы
for x in xrange(img.size[0]):
    for y in xrange(img.size[1]):
        r,g,b = img.getpixel((x,y))
        img.putpixel((x,y),(b, r,g))
img.show()  # замена каналов
```





Задание 2. Сборка изображения из известных каналов изображения.

1. Выделите каналы изображения.

```
>>> img = Image.open("foto.jpg")
>>> R, G, B = img.split()
```

`split()` – выдает каналы изображения в виде кортежа.

2. Соберите изображение из каналов, полученных выше.

```
>>> img2 = Image.merge("RGB", (R, G, B) )
>>> img2.mode
'RGB'
>>> img2.show()
```

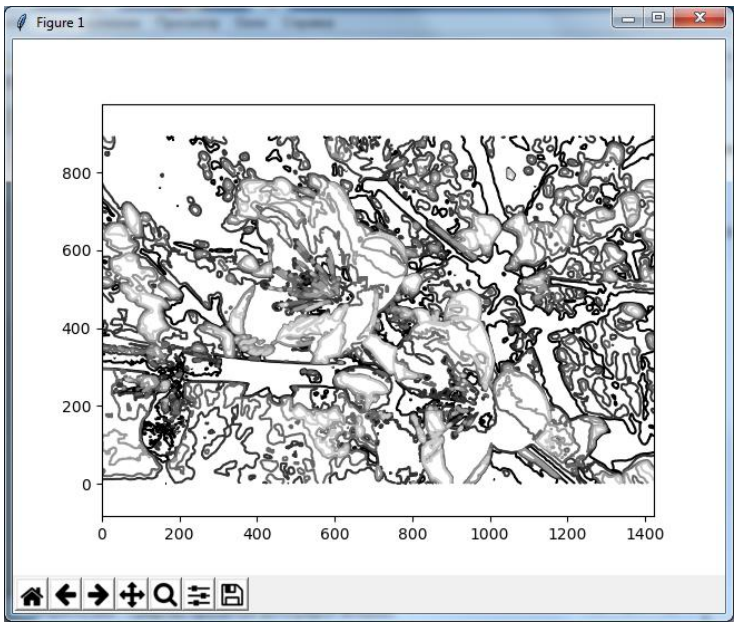
Задание 3. Построение гистограммы.

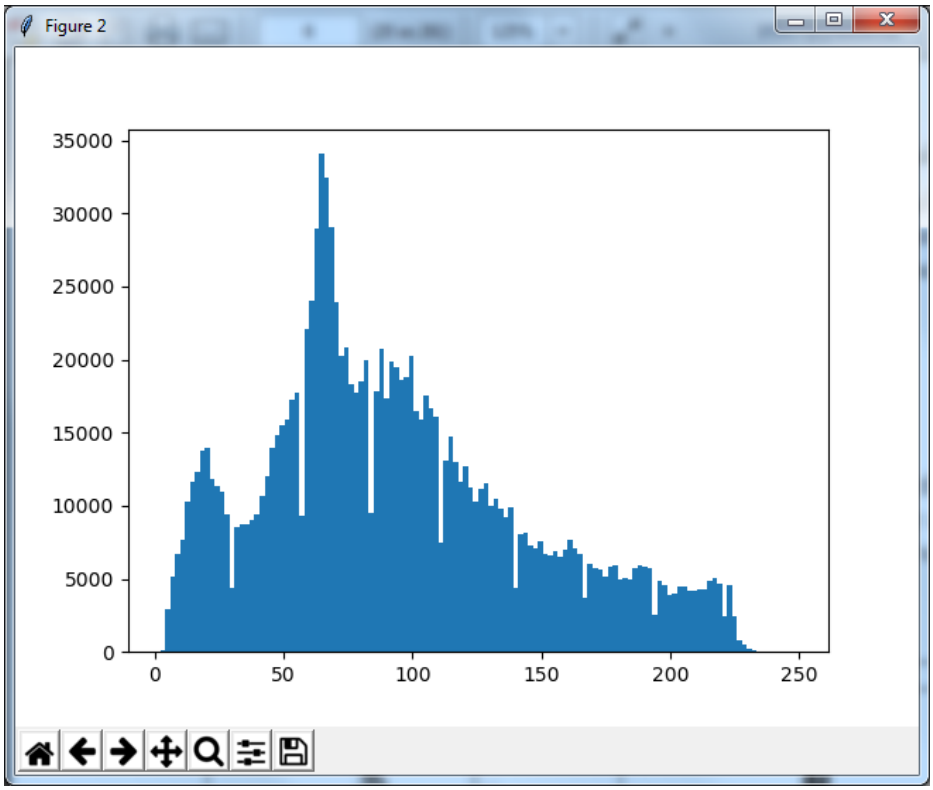
Отройте изображение в переменную im и постройте гистограмму изображения.

Ниже приведен код построения гисторамммы.

```
40 """
41 figure()     #выдать окно
42 hist(im.flatten(),128)          #построить гистограмму
43 show()
```

Ниже показано открытое исходное изображение и его гистограмма.





Литература к работе

1. Python Imaging Library (Fork), https://pypi.org/project/Pillow/
2. Прохоренок, Н. А. Python 3. Самое необходимое: Пособие / Прохоренок Н.А., Дронов В.А. - СПб:БХВ-Петербург, 2016. - 464 с. ISBN 978-5-9775-3631-8. - Текст : электронный. - URL: https://codernet.ru/books/python/python_3_samoe_neobxodimoe_proxorenok/
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

ЛАБОРАТОРНАЯ РАБОТА 5.
МАНИПУЛИРОВАНИЕ ИЗОБРАЖЕНИЕМ

Цель работы: научится в коде Python с использованием внешней библиотеки PIL копировать изображение, изменять размеры изображения , вырезать кусок изображения и изменять размеры вырезанного куска, вращать изображение.

Подготовка к выполнению заданий.

1. Поместите в текущую папку набор цветных изображений и изображений в оттенках серого.
2. Поместите не в текущую папку набор других цветных изображений и изображений в оттенках серого.
3. На компьютере должен быть выход в интернет.
4. Установите внешнюю библиотеку PIL.

Задание 1. Копирование и изменение размера изображения.

1. Скопируйте открытое изображение в и выведите на экран полученную копию

```
# Подключаем модуль:
from PIL import Image
# Открываем файл:
img = Image.open("foto.jpg")
# Создаем копию:
img2 = img.copy()
# Просматриваем копию:
img2.show()
```

2. Создайте уменьшенную версию изображения указанного размера.

Сначала откройте некоторое изображение и выведите его размеры

```
img=Image.open("foto.jpg")
img.size  # Исходные размеры изображения
```

В итоге в консоль выведется размер этого изображения:

```
(800, 600)
```

Измените размер изображения пропорционально с помощью метода

thumbnail (<Размер> [, <Фильтр>]).

Обратите внимание на то, что изменение размера производится пропорционально, иными словами, за основу берется минимальное значение, а второе значение вычисляется пропорционально первому.

Метод изменяет само изображение и ничего не возвращает.

Фильтры **NEAREST, BILINEAR, BICUBIC или ANTIALIAS .**

```
img.thumbnail((400, 300), Image.ANTIALIAS)
img.size # Изменяется само изображение
```

В итоге в консоль выведется размер измененного изображения:

```
(400, 300)
img = Image.open("foto.jpg")
img.thumbnail((400, 100), Image.ANTIALIAS)
img.size # Вывод размера изображения
```

Увидим, что размер изменяется пропорционально

В итоге в консоль выведется размер измененного изображения:

```
(133, 100)
```

3. Создайте версию изображения указанного размера

Сначала откройте некоторое изображение и выведите его размеры

```
img = Image.open("foto.jpg")
img.size # Исходные размеры изображения
```

В итоге в консоль выведется размер измененного изображения:

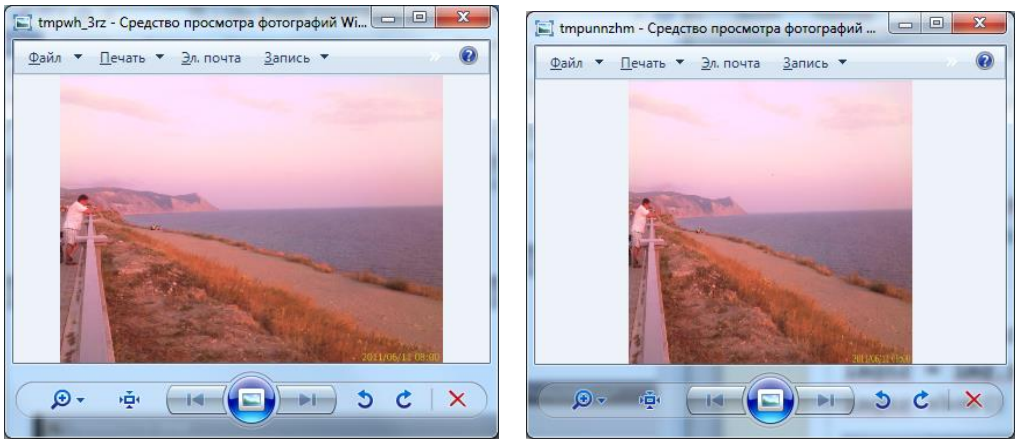
```
(800, 600)
```

Теперь изменим само изображение

```
img.show()
newsize = (400, 400)
imgnr = img.resize(newsize)
```

imgnr.show() # Изменяется само изображение

В итоге получим:



Задание 2. Вырезка куска изображения и изменение его размеров

1. Вырежьте прямоугольный фрагмент из исходного изображения.

```
>>> img = Image.open("foto.jpg")
>>> img2 = img.crop( [0, 0, 100, 100] ) # Помечаем фрагмент
>>> img2.load() # Считываем фрагмент, создавая новое изображение
>>> img2.size
(100, 100)
```

2. Измените размеры куска изображения

   Сначала откройте некоторое изображение и выведите его рамеры

   img = Image.open("tsveti.jpg")

   img.size # вывод размеров изображения

   img.show()

   В итоге откроется исходное изображение и в консоль выведется размер исходного изображения:

   (668, 438)

   Затем возьмем кусок изображения

   box = (100,100,300,300) #берем кусок изображения:

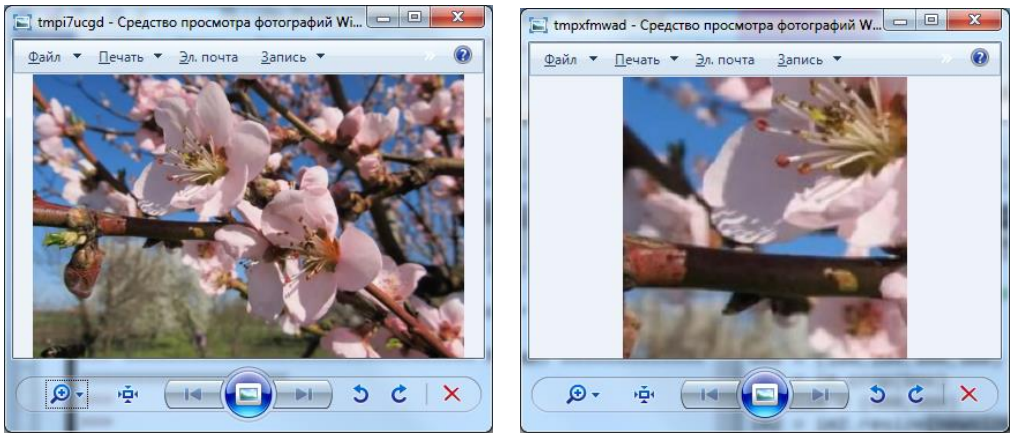
   Затем изменяем само изображение

   img2 = img.crop(box)

   newsize = (400, 400)

```
img2nr = img2.resize(newsize)
img2nr.show() # Изменяется само изображение
```

Получим в итоге:



Задание 3. Вращение изображения

1. Осуществите вращение изображения. Но сначала откройте некоторое изображение и выведите его размеры в консоль.

```
img = Image.open("foto.jpg")
img.size # Исходные размеры изображения
```

В консоль выведется размер исходного изображения:

```
(800, 600)
```

Затем осуществите поворот на 90 градусов

```
img2 = img.rotate(90)# Поворот на 90 градусов
img2.size
```

В консоль выведется размер повернутого на 90 градусов изображения:

```
 (600, 800)
img3 = img.rotate(45, Image.NEAREST)
img3.size # Размеры сохранены, изображение обрезано
(800, 600)
img4 = img.rotate(45, expand=True)
img4.size # Размеры увеличены, изображение полное
(991, 990)
```

2. Получите горизонтальный и зеркальный образ изображения:

```
>>> img2 = img.transpose(Image.FLIP_LEFT_RIGHT)
>>> img2.show() # Горизонтальный зеркальный образ
>>> img3 = img.transpose(Image.FLIP_TOP_BOTTOM)
>>> img3.show() # Вертикальный зеркальный образ
```

и следующими способами:

```
>>> img4 = img.transpose(Image.ROTATE_90)
>>> img4.show() # Поворот на 90° против часовой стрелки
>>> img5 = img.transpose(Image.ROTATE_180)
>>> img5.show() # Поворот на 180°
```

Литература к работе

1. Python Imaging Library (Fork), https://pypi.org/project/Pillow/
2. Прохоренок, Н. А. Python 3. Самое необходимое: Пособие / Прохоренок Н.А., Дронов В.А. - СПб:БХВ-Петербург, 2016. - 464 с. ISBN 978-5-9775-3631-8. - Текст : электронный. - URL: https://codernet.ru/books/python/python_3_samoe_neobxodimoe_proxorenok/
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

ЛАБОРАТОРНАЯ РАБОТА 6.
МАНИПУЛЯЦИИ НА ИЗОБРАЖЕНИИ

Цель работы: ознакомление со способами закраски (заливки) части и всего изображения заданным цветом, вывода координат кликнутых на картинке точек, вставки объектов на изображение, рисования на изображении с использованием внешней библиотеки PIL в Python.

Подготовка к выполнению заданий.

Поместите в текущую папку набор цветных изображений и изображений в оттенках серого.

1. Поместите не в текущую папку набор других цветных изображений и изображений в оттенках серого.
2. На компьютере должен быть выход в интернет.
3. Установите внешнюю библиотеку PIL.

Задание 1 . Закраска (заливка) части и всего изображения заданным цветом.

1. Закрасьте область изображения красным цветом. Для этого сначала откройте исходное изображение:

```
img = Image.open("foto.jpg")
```

Затем закрасьте на нем прямоугольную область:

```
img.paste( (255, 0, 0), (0, 0, 100, 100) )
img.show()
```

2. Залейте все изображение зеленым цветом.

```
img = Image.open("foto.jpg")
img.paste( (0, 128, 0), img.getbbox() )
img.show()
```

Задание 2 . Комбинированные манипуляции на изображении.

1. Загрузите изображение, создайте его уменьшенную, а затем вставьте ее в исходное изображение, причем вокруг вставленного изображения отобразите рамку красного цвета.

```
>>> img = Image.open("foto.jpg")
>>> img2 = img.resize( (200, 150) ) # Создаем миниатюру
>>> img2.size
(200, 150)
>>> img.paste( (255, 0, 0), (9, 9, 211, 161) ) # Рамка
>>> img.paste(img2, (10, 10) )       # Вставляем миниатюру
>>> img.show()
```

2. Выведите белую полупрозрачную горизонтальную полосу высотой 100 пикселов. Для этого выполните:

```
>>> img = Image.open("foto.jpg")
>>> white = Image.new("RGB", (img.size[0],100), (255,255,255))
>>> mask = Image.new("L", (img.size[0], 100), 64) # Маска
>>> img.paste(white, (0, 0), mask)
>>> img.show()
```

3. Копируйте и вставьте повернутую часть изображения:

```
box = (100,100,400,400)
region = pil_im.crop(box)

region = region.transpose(Image.ROTATE_180)
pil_im.paste(region,box)
```

Задание 3. Вывод координат кликнутых на картинке точек

Откройте в картинку, выведите ее на экран и напишите программу, выводящую координаты трех кликнутых на картинке точек:

```
56 from PIL import Image
57 from pylab import *
58 im = array(Image.open("tsveti.jpg"))
59 imshow(im)
60 print("Please click 3 points")
61 x = ginput(3)
62 print('you clicked:',x)
```

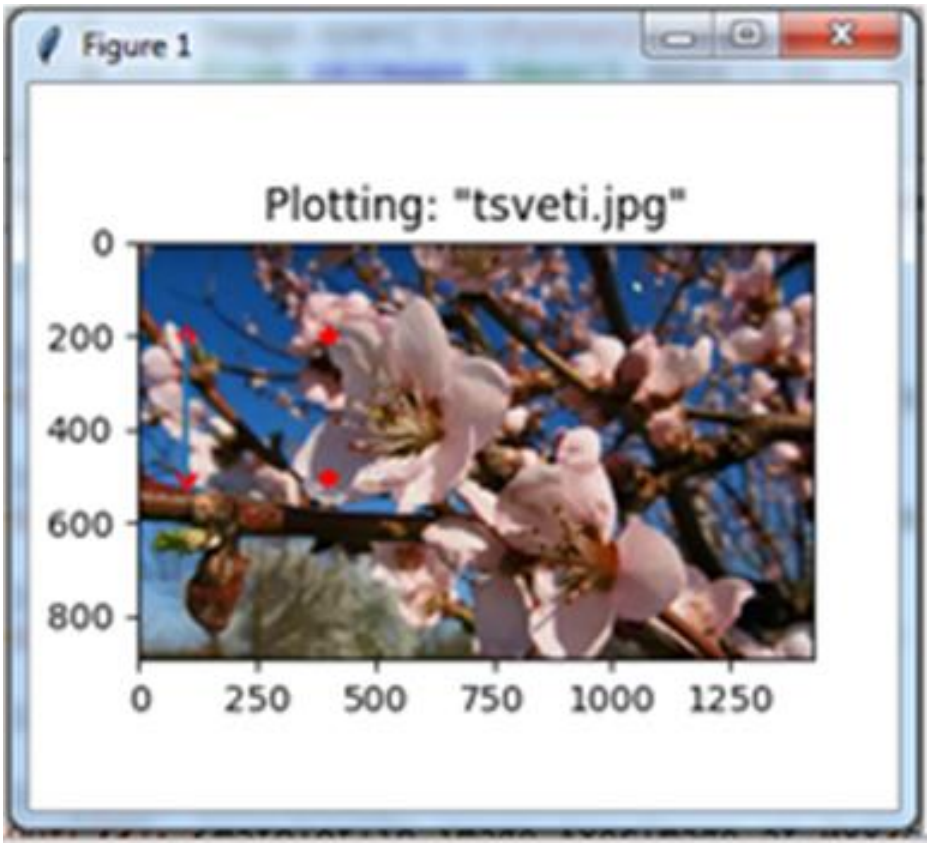
потом на изображении нажмите 3 точки

```
Python                                    ~\Documents\2018 fleshki\Evg4\internet1\Pytho

In [72]: print('you clicked:',x)
you clicked: [(1145.0498472116119, 735.42929585627348), (818.06425673717763,
368.45266194251997), (425.21106530386453, 439.02509154131872)]

In [73]:
```

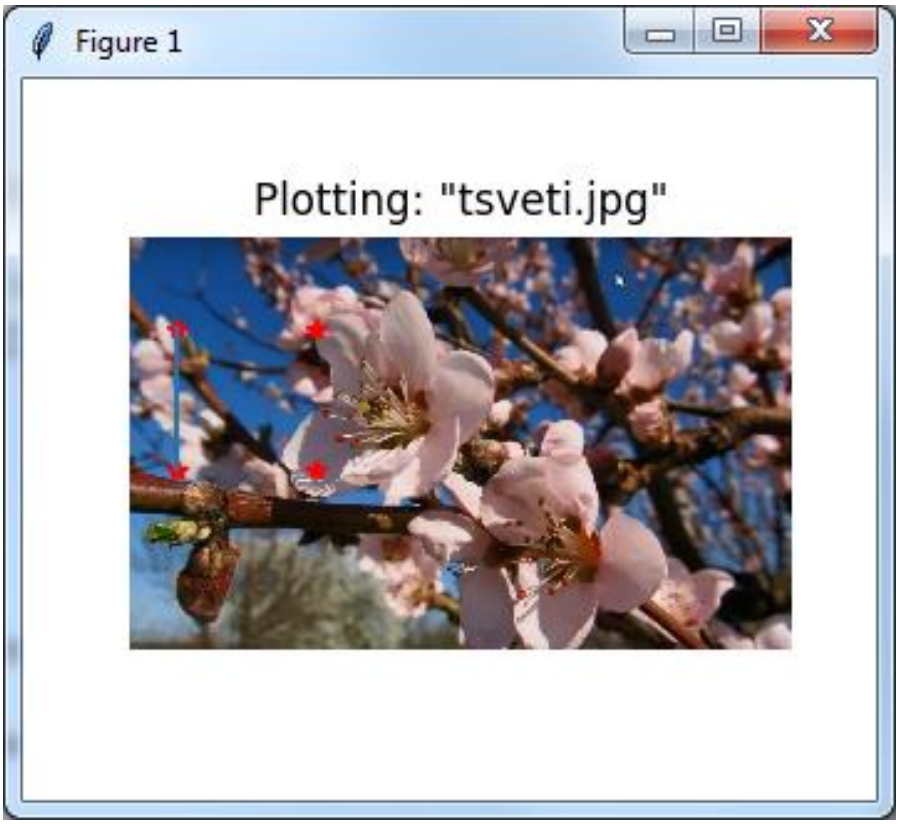
Задание 4. Рисование на изображении

1. Поставьте точки и нарисуйте линии на изображении.

```
1 from PIL import Image
2 from pylab import *
3 # запишем файл-изображение в массив (read image to array)
4 im = array(Image.open("tsveti.jpg"))
5 # plot the image
6 imshow(im)
7 # some points
8 x = [100,100,400,400]
9 y = [200,500,200,500]
10 # plot the points with red star-markers
11 plot(x,y,'r*')
12 # line plot connecting the first two points
13 plot(x[:2],y[:2])
14 # add title and show the plot
15 title('Plotting: "tsveti.jpg"')
16 show()
```




```
17 axis('off')  #убрать оси
18 plot(x,y)    #синяя сплошная линия по умолчанию
19 plot(x,y,'go-') # зеленая линия с круглыми маркерами
20 plot(x,y,'ks:') # bчерная пунктирная линия с квадратными маркерами
```



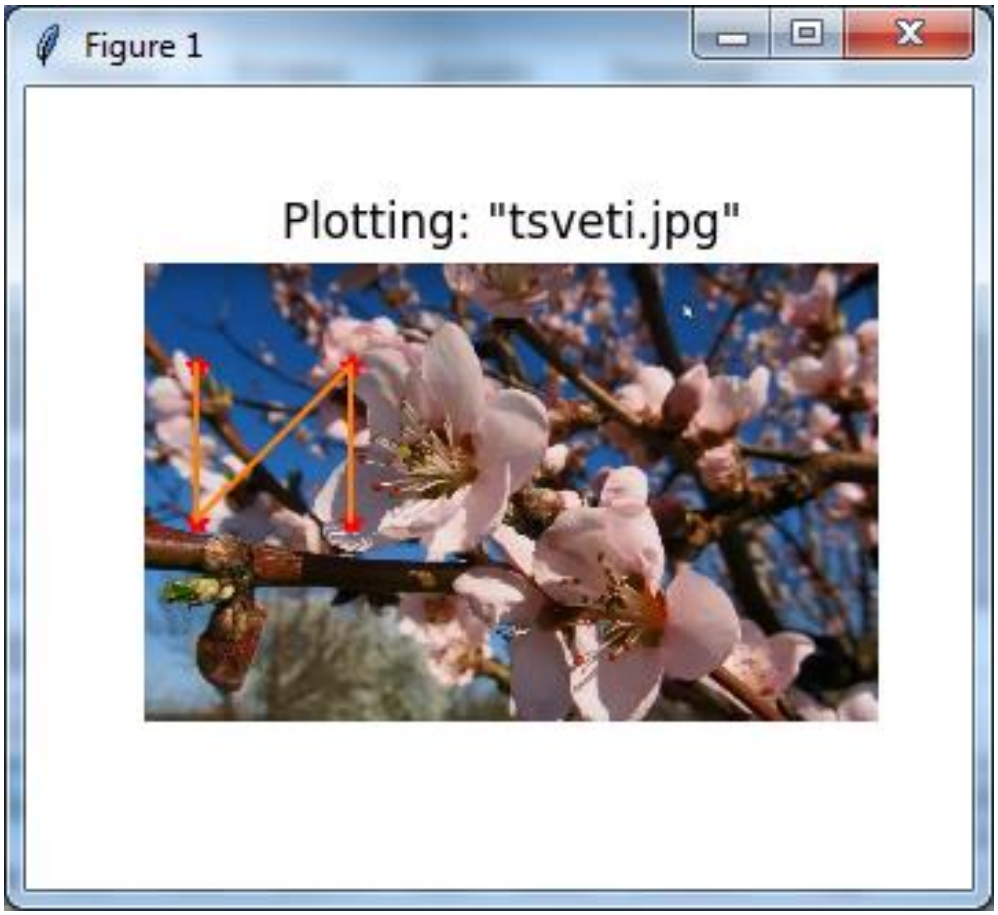


Figure 1
Plotting: "tsveti.jpg"

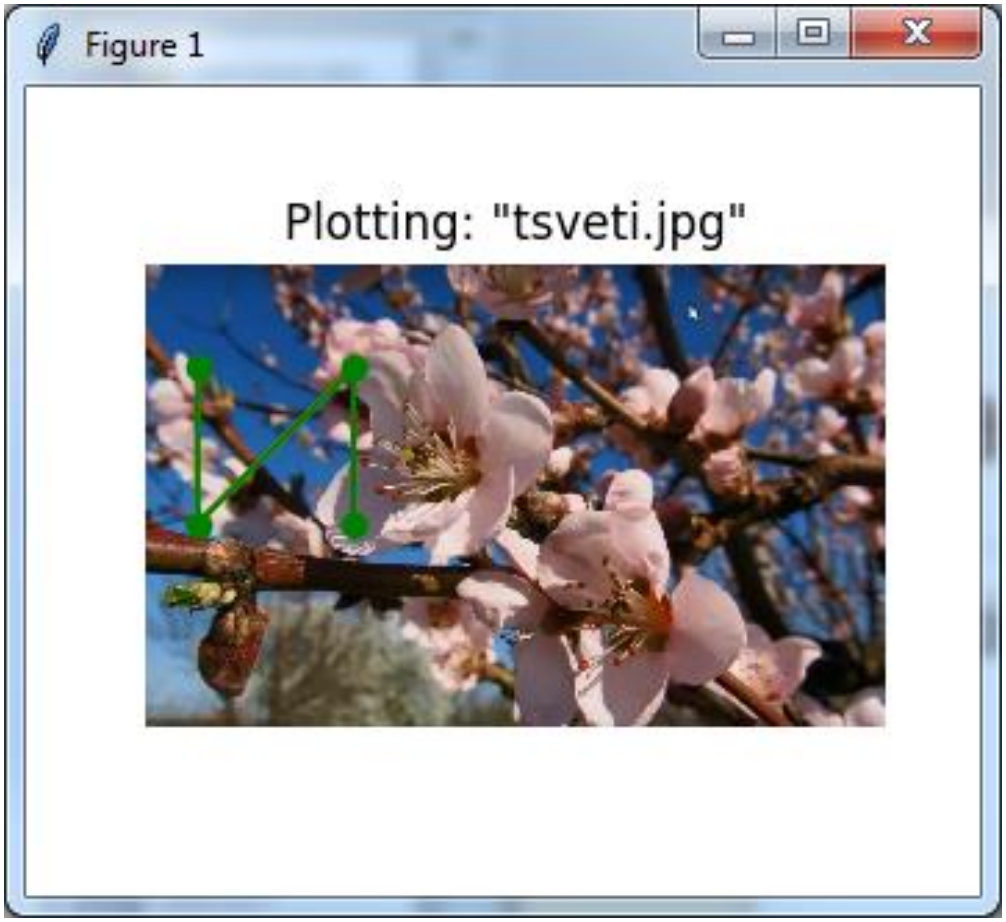
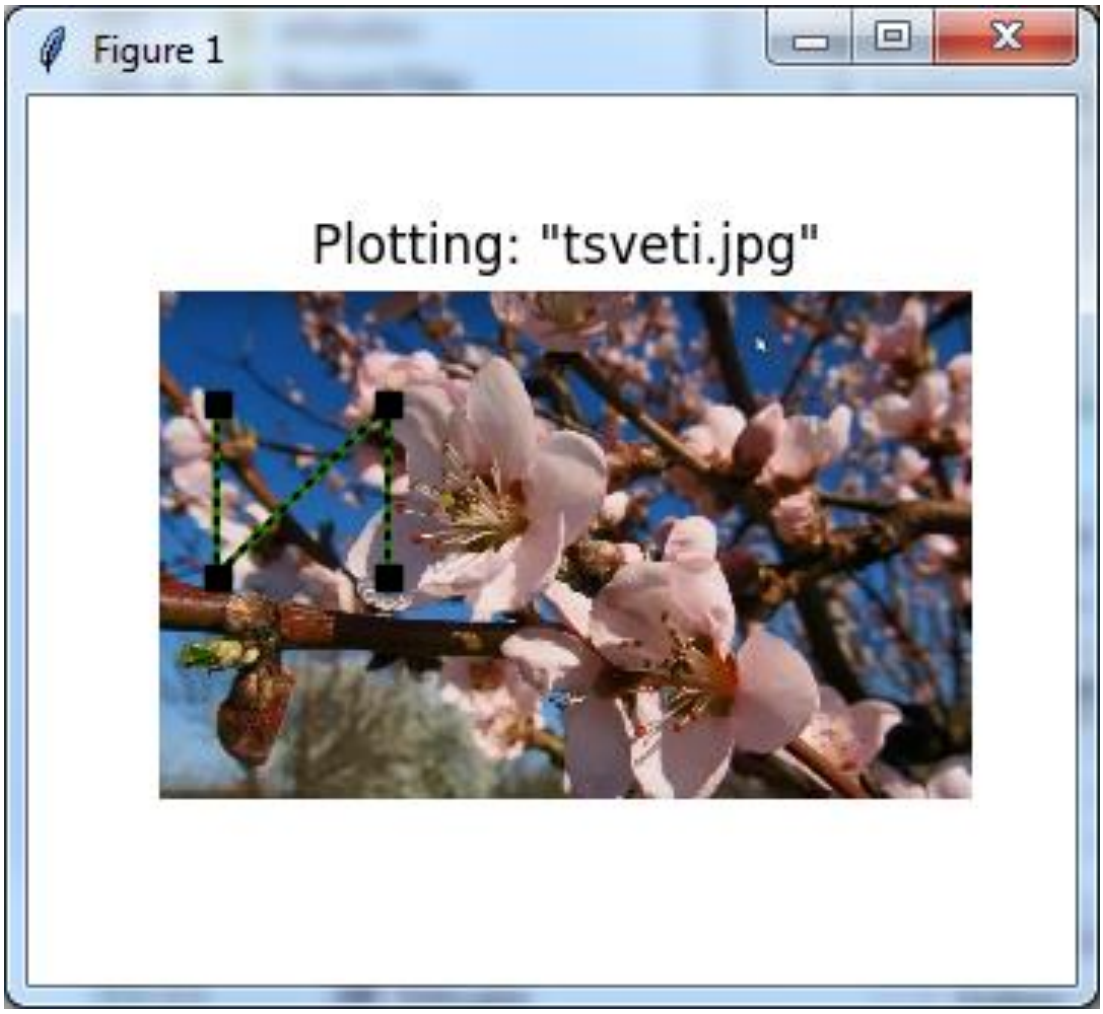


Figure 1
Plotting: "tsveti.jpg"



Литература к работе

1. Python Imaging Library (Fork), https://pypi.org/project/Pillow/
2. Прохоренок, Н. А. Python 3. Самое необходимое: Пособие / Прохоренок Н.А., Дронов В.А. - СПб:БХВ-Петербург, 2016. - 464 с. ISBN 978-5-9775-3631-8. - Текст : электронный. - URL: https://codernet.ru/books/python/python_3_samoe_neobxodimoe_proxorenok/
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

ЛАБОРАТОРНАЯ РАБОТА 7.
ФИЛЬТРАЦИЯ ИЗОБРАЖЕНИЙ. КОНТУРЫ НА ИЗОБРАЖЕНИИ

Цель работы: ознакомление со способами фильтрация изображений и получения контуров объектов на изображении с использованием внешней библиотеки PIL в Python.

Подготовка к выполнению заданий.

1. Поместите в текущую папку набор цветных изображений. Среди них должны быть изображения:
   - резкое, т.е. в котором все детали видны четко,
   - размытое,
   - с шумами разных видов,
   - резкое, но в тенях плохо видны детали.
2. Поместите не в текущую папку набор других цветных изображений. Среди них тоже должны быть изображения:
   - резкое, т.е. в котором все детали видны четко,
   - размытое,
   - с шумами разных видов,
   - резкое, но в тенях плохо видны детали.
3. На компьютере должен быть выход в интернет.
4. Установите внешнюю библиотеку PIL.

Замечание 1. Полезная функция.

В дальнейшем нам понадобятся списки изображений для обработки. Вот как можно создать список имен файлов всех изображений в папке:

```
import os

def get_imlist(path):
  """  Returns a list of filenames for
    all jpg images in a directory. """

  return [os.path.join(path,f) for f in os.listdir(path) if f.endswith('.jpg')]
```

Примите все файлы изображений в список имен файлов (список файлов) и преобразуйте изображения к файлам **JPEG.**

Сначала получите **filelist** указано в замечании Затем выполните:

```
from PIL import Image
import os

for infile in filelist:
  outfile = os.path.splitext(infile)[0] + ".jpg"
  if infile != outfile:
    try:
      Image.open(infile).save(outfile)
    except IOError:
      print "cannot convert", infile
```

Задание 1. Фильтрация изображений.

1. Примените к изображению каждый из следующих фильтров:

   BLUR, CONTOUR, DETAIL, EDGE_ENHANCE, EDGE_ENHANCE_MORE, EMBOSS, FIND_EDGES, SHARPEN, SMOOTH и SMOOTH_MORE

   из модуля **ImageFilter** и выведите отфильтрованное изображение на экран.

```
>>> from PIL import ImageFilter
>>> img = Image.open("foto.jpg")
>>> img2 = img.filter(ImageFilter.EMBOSS)
>>> img2.show()
```

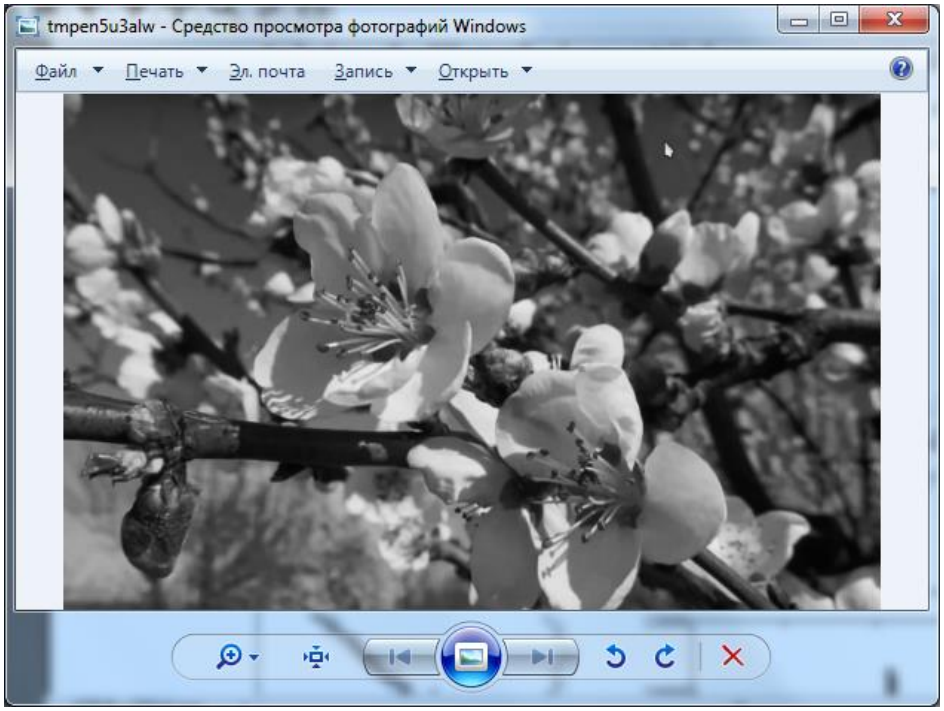
Задание 2. Контуры на изображении (один из способов).

1. Выделите контуры на изображении.

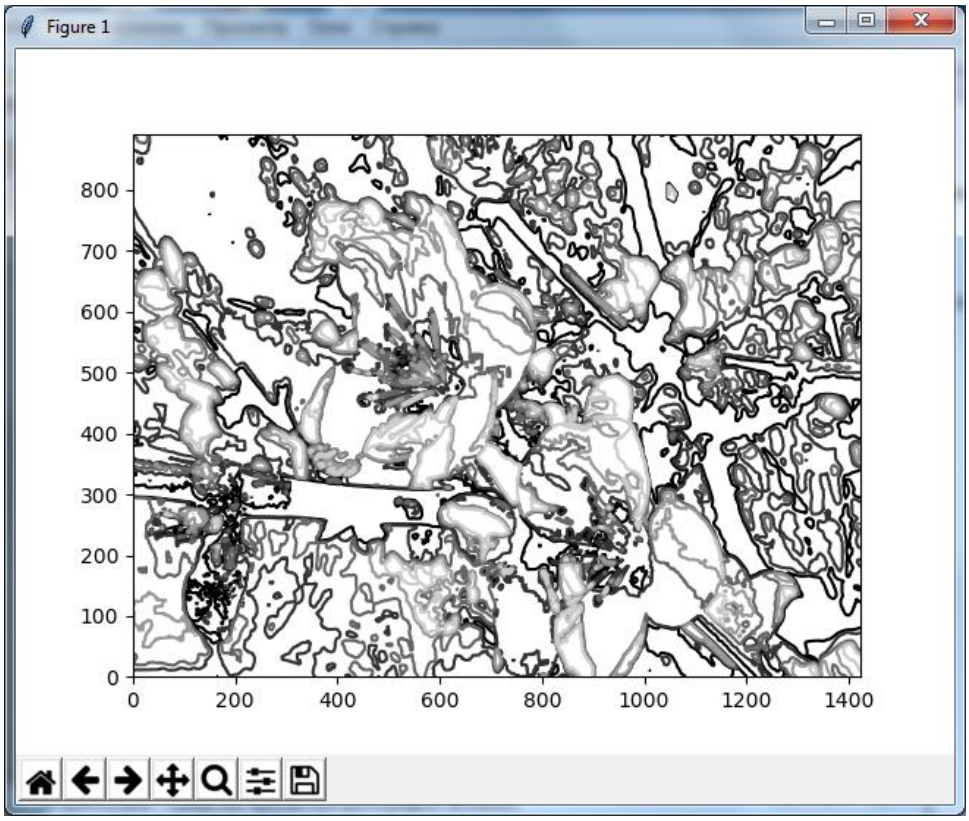
```
from PIL import Image
from pylab import *

# read image to array
im = array(Image.open('empire.jpg').convert('L'))

# create a new figure
figure()
# don't use colors
gray()
# show contours with origin upper left corner
contour(im, origin='image')
axis('equal')
axis('off')
```

Откроем исходное изображение, запустив команду:

```
im.show()
```



После выполнения приведенной выше программы на экране получим:



Литература к работе

1. Python Imaging Library (Fork), https://pypi.org/project/Pillow/
2. Прохоренок, Н. А. Python 3. Самое необходимое: Пособие / Прохоренок Н.А., Дронов В.А. - СПб:БХВ-Петербург, 2016. - 464 с. ISBN 978-5-9775-3631-8. - Текст : электронный. - URL: https://codernet.ru/books/python/python_3_samoe_neobxodimoe_proxorenok/
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

ЛАБОРАТОРНАЯ РАБОТА 8.
СППР «ОПРЕДЕЛЕНИЕ КАЧЕСТВА ПОВЕРХНОСТИ ПОРИСТОГО МАТЕРИАЛА». Часть 1.

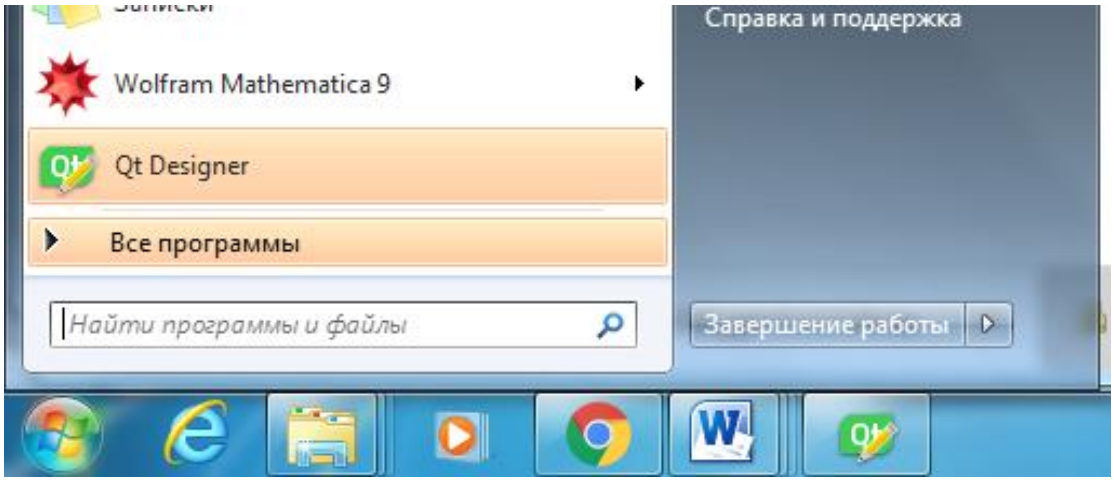
Цель работы: создание интерфейса СППР «Определение качества поверхности пористого материала».

Подготовка к выполнению задания.

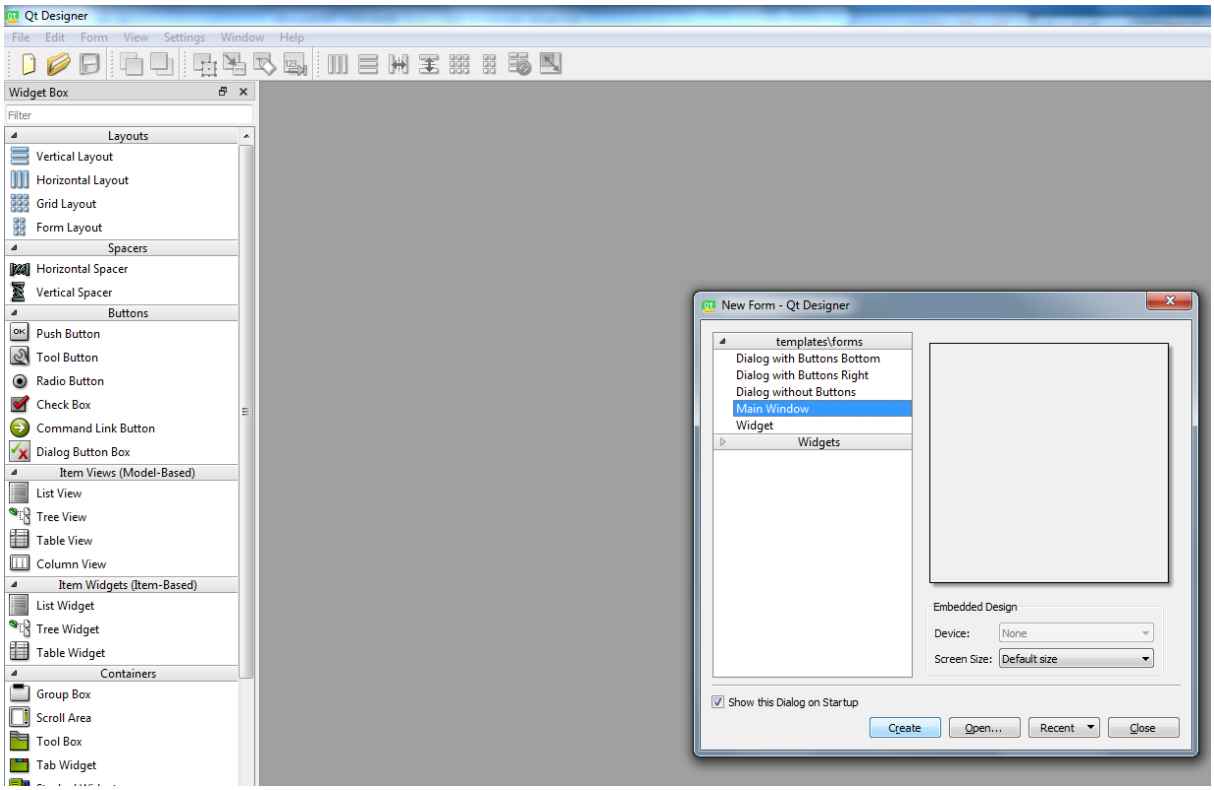
1. На компьютере должен быть выход в интернет.
2. Скачайте и установите внешнюю библиотеку **`PyQt5`**. Ссылки для скачивания: https://sourceforge.net/projects/pyqt/files/, https://www.riverbankcomputing.com или https://www.lfd.uci.edu/~gohlke/pythonlibs/#vlfd.
3. Скачайте и установите программу **`QtDesigner.`** Ссылка для скачивания https://build-system.fman.io/qt-designer-download.

Задание. Создание интерфейса СППР.

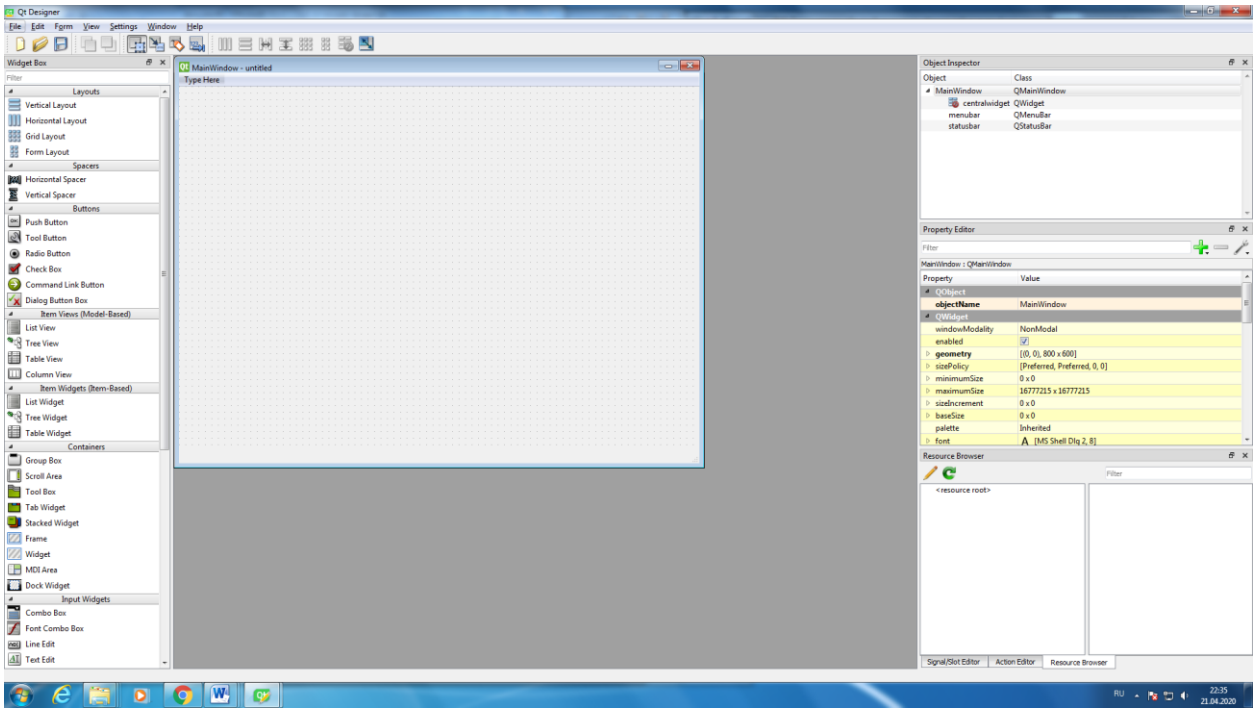
Войдите в программу **`Designer`** , например, через проводник, кликнув на **`Designer:`**



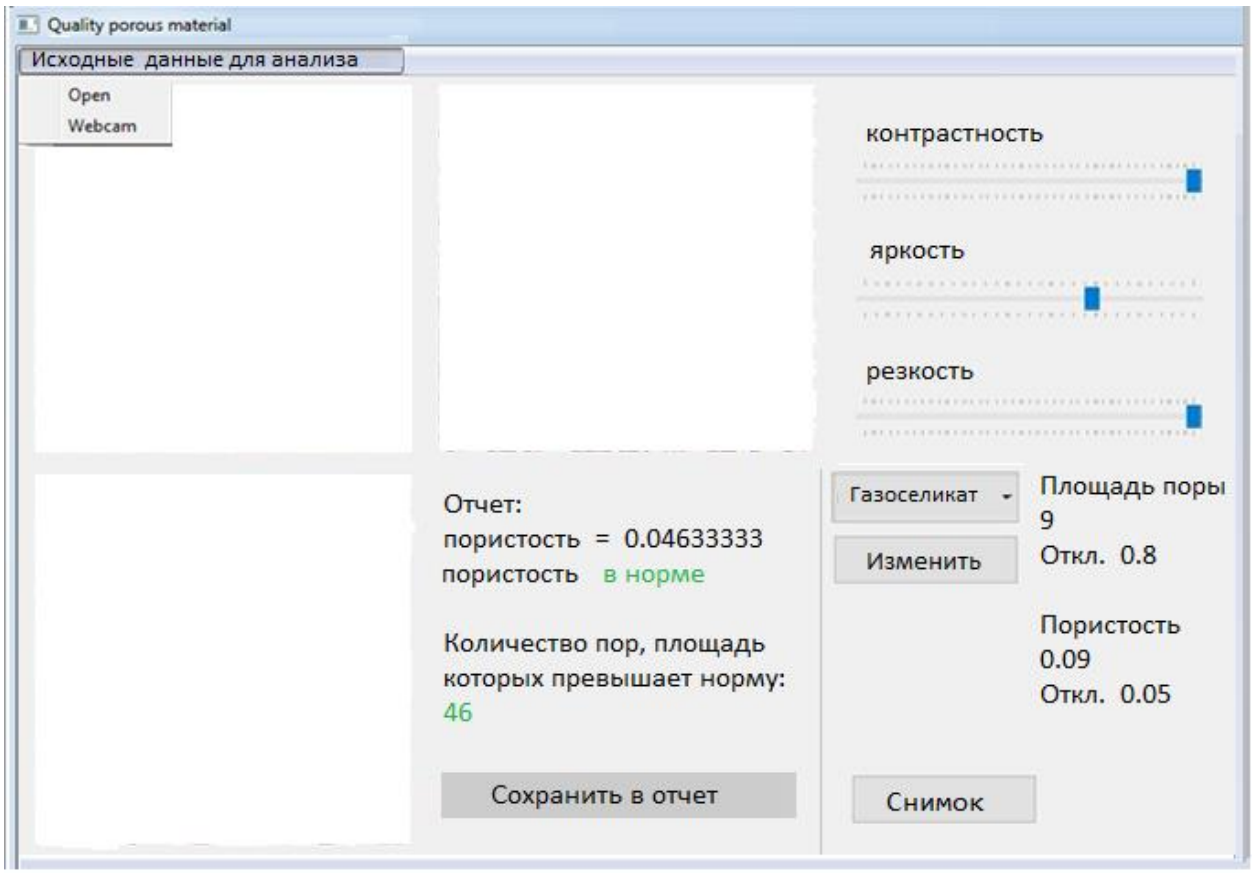
Откроется главное окно программы **`Designer`**. Выберете в нем MainWidows и кликнете Creater:



Откроется форма:



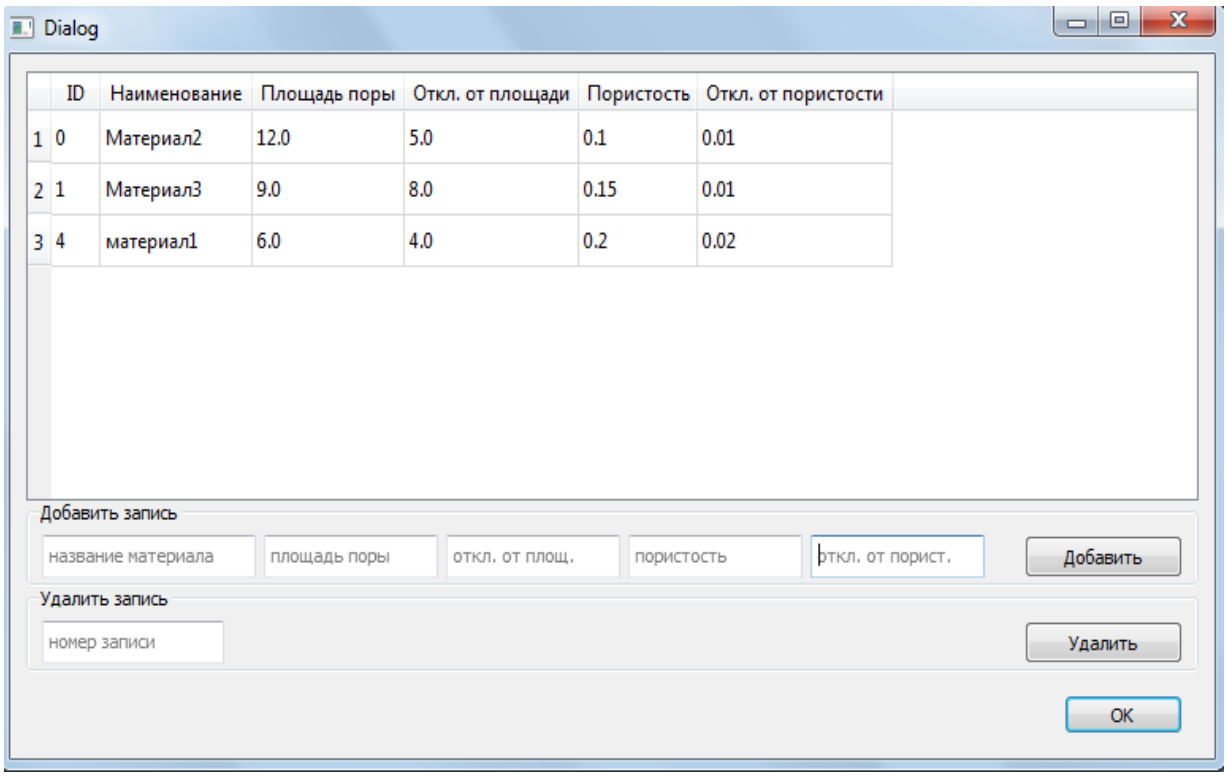
Перетаскивая в рабочую область окна нужные виджеты, создайте интерфейс СППР «Определение качества поверхности пористого материала». Примерный интерфейс этой СППР:



В левом верхнем окне размещается текущее изображение материала, справа от него – изображение для анализа системой, в нижнем окне – оставляются только те поры, площадь которых больше допустимой.

С помощью движков, расположенных в правой области формы, ЛПР может улучшить качество изображения для анализа системой. Тем самым ЛПР может настроить СППР для работы.

После клика на кнопку «Изменить» в готовой СППР должно открываться диалоговое окно для работы с БД материалов. Создайте интерфейс и этого окна:



Исходная база данных в одном из следующих заданий будет прописываться в коде программы. Поэтому при открытии этого диалогового окна будет отображаться в верхней его области. В ней можно увидеть количество материалов, их названия, порядковые номера и параметры. В нижней области мы можем изменить (вставить или удалить) строки в уже существующей базе данных. Таким образом, у пользователя есть возможность добавления нового материала в базу данных, изменения текущих значений параметров, а также удаления материалов из базы. Но пока на этой форме верхнее окно пустое и кнопки не действуют.

Кликнув на кнопке с разворачивающимся меню, расположенной на главной форме СППР выше только что указанной, выбираем из разворачивающегося списка наименование пористого материала для контроля качества. После этого справа от этой кнопки появляются считанные из БД характеристики нормы для выбранного материала.

В кнопке «Исходные данные для анализа» выбирается файл с текущим изображением или указывается, что текущее изображение поступает с камеры в режиме реального времени или берется из файла.

Рядом с нижним окном – область отчета по результатам анализа.

Пока на форме нашей СППР есть виджеты, но они не действуют. Т.о., Вы должны создать файлы с интерфейсом форм нашей системы.

1. Сохраните свои файлы, подготовленные в **Qt Designer** как **uisppr.ui** и **uibd.ui**.
2. Далее можем поступить одним из приведенных ниже двух способов:

   Способ 1. Использовать .ui-файлы напрямую из Python-кода.

   Создайте другой файл, скажем **uisppr.py**, со следующим содержимым:

```
from PyQt5 import uic
from PyQt5.QtWidgets import QApplication

Form, Window = uic.loadUiType("uisppr.ui")

app = QApplication([])
window = Window()
form = Form()
form.setupUi(window)
window.show()
app.exec_()
```

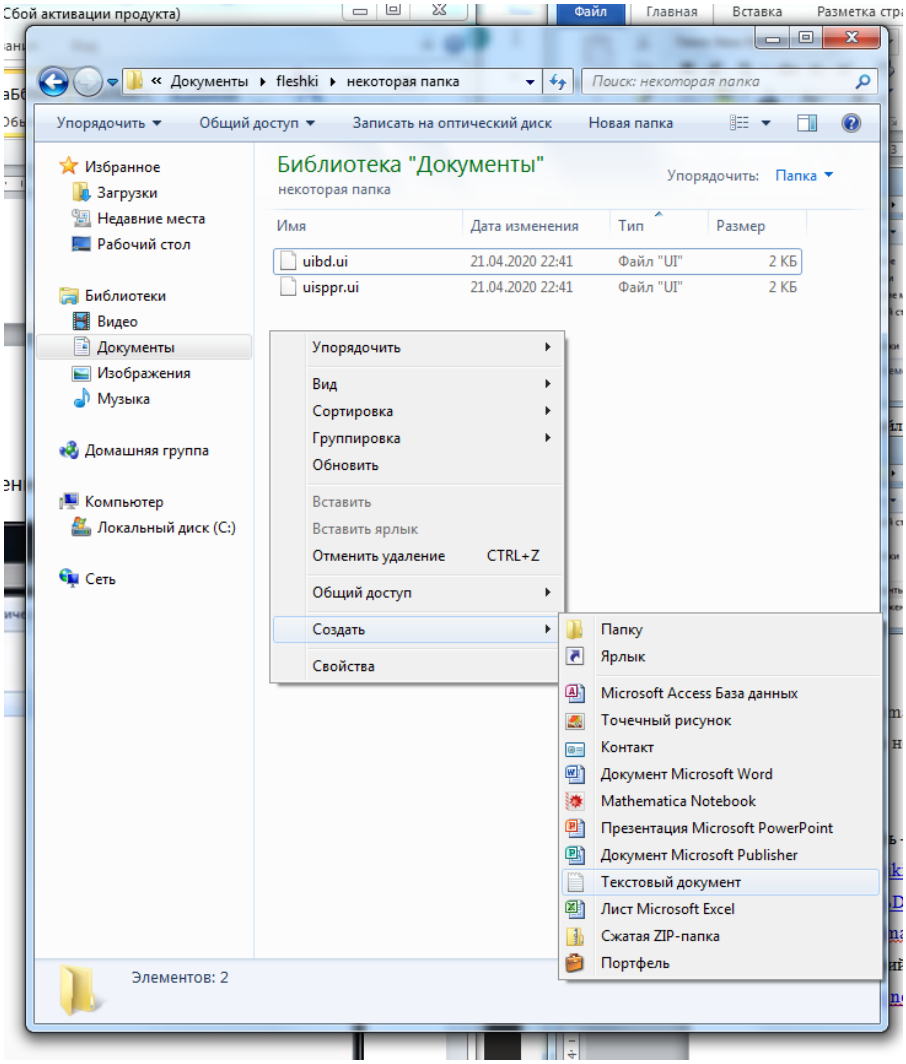
   Аналогично создайте другой файл, скажем **uibd.py**, со следующим содержимым:

```
from PyQt5 import uic
from PyQt5.QtWidgets import QApplication
```

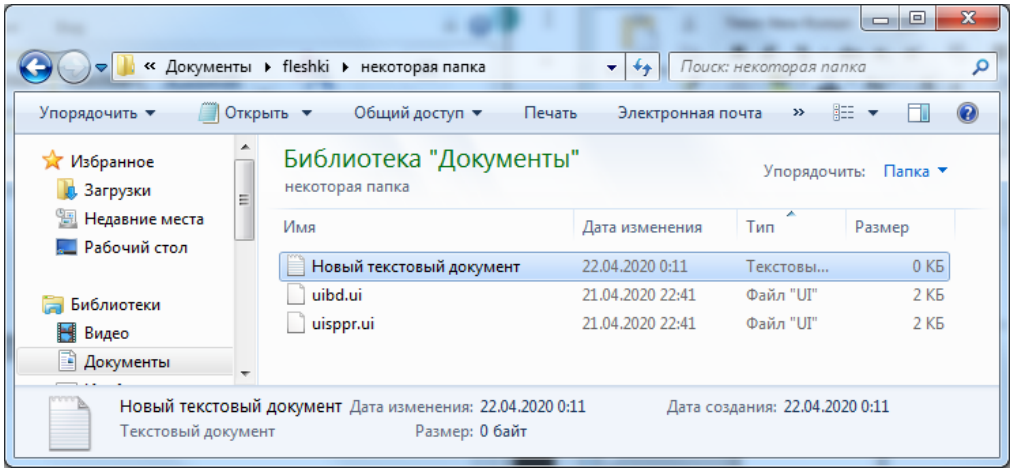
```
Form, Window = uic.loadUiType("uibd.ui")

app = QApplication([])
window = Window()
form = Form()
form.setupUi(window)
window.show()
app.exec_()
```

Способ 2. Конвертировать .ui-файл в Python-файл.

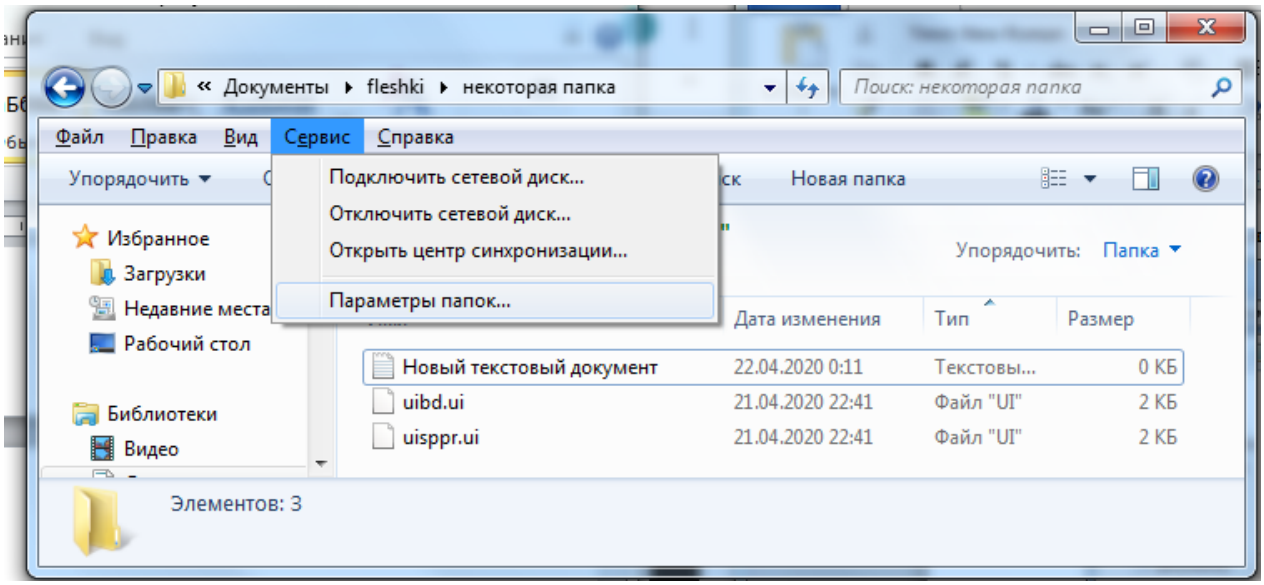
Это можно делать по-разному. Сделайте это так. Сначала создайте **bat**-файл. Для этого сначала создаем текстовый документ, через контекстное меню:



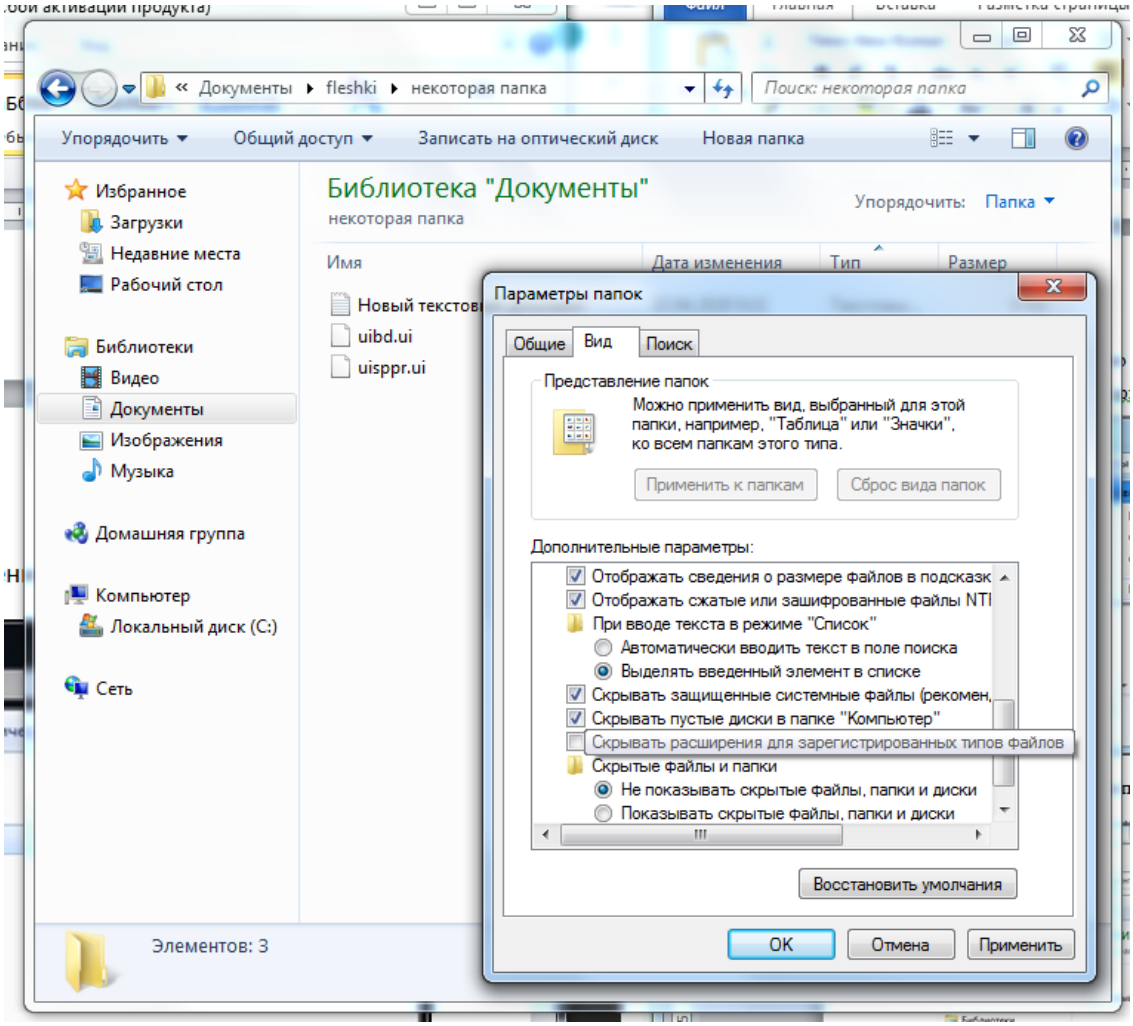
Может так быть, что расширение не отображается:



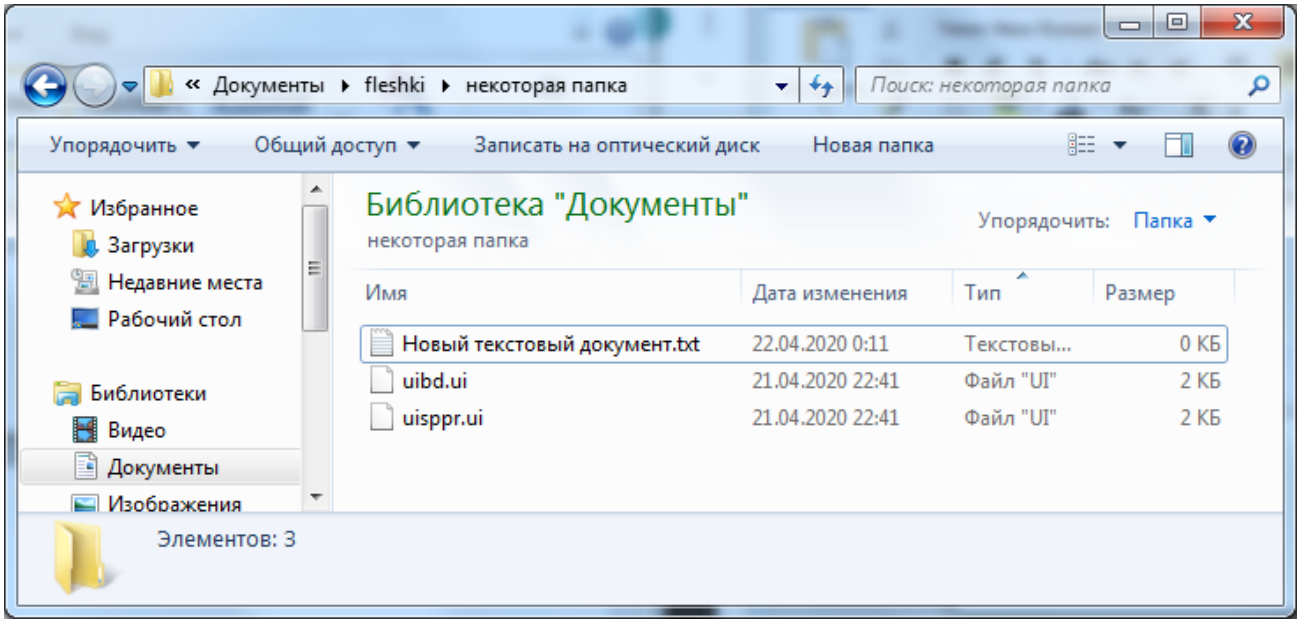
Для того чтобы оно стало видным надо нажать клавишу **`Alt`**. Появится наверху меню. Надо выбрать: «Сервис» - «Параметры папок»:



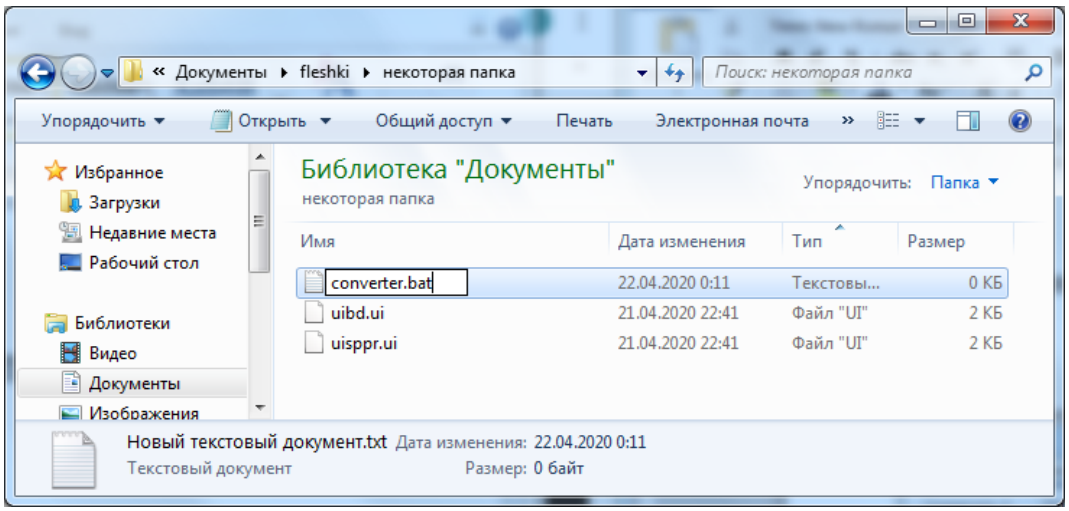
В раскрывшемся окне пойти на вкладку «Вид». Галочки не должно быть:



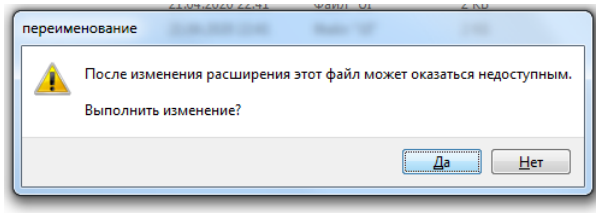
И у вас появится расширение .txt:



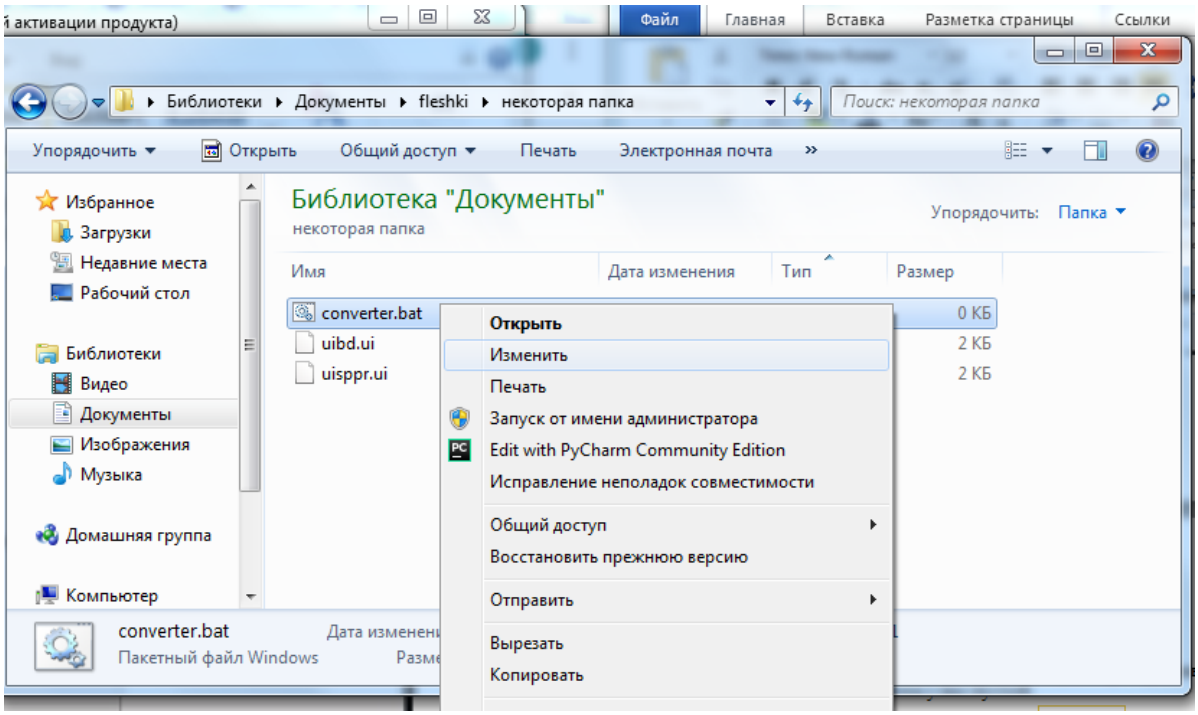
Теперь этот файл переименуем и сами напишем ему расширение .bat:



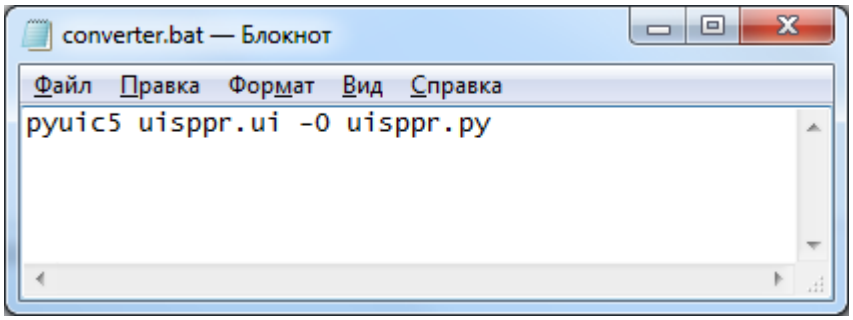
После этого появится сообщение:



Кликаем «**Ok**». Файл **converter.bat** и будет преобразовывать наш .ui-файл в **.py**-файл. Файл converter.bat пока у нас пустой. Изменяем его:



В открывшемся окне пишем команду:



Сохраняем измененный файл и закрываем его.

Запускаем .bat-файл, два раза щелкнув на левую клавишу мыши. Ждем некоторое время и видим, что конвертация ui-файла в **`.py`**-файл **`uisppr.py`** произошла. Открыв и запустив его на выполнение, получим форму с нашим интерфейсом. Аналогичную процедуру сделайте и для файла **`uibd.ui`**.

Литература к работе

1. Руководство по Qt Designer, http://doc.crossplatform.ru/qt/4.5.0/designer-manual.html
2. Qt Documentation, https://doc.qt.io/qt-5/qtdesigner-manual.html
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

ЛАБОРАТОРНАЯ РАБОТА 9.
СППР «ОПРЕДЕЛЕНИЕ КАЧЕСТВА ПОВЕРХНОСТИ ПОРИСТОГО МАТЕРИАЛА». Часть 2.

Цель работы: создание базы данных для СППР «Определение качества поверхности пористого материала».

Подготовка к выполнению задания.

1. На компьютере должен быть выход в интернет.
2. На компьютере должна быть возможность работы в SQL.
3. Скачайте и установите а проект внешнюю библиотеку `sqlite3`.

Задание. Создание БД СППР.

1. Создайте проект **`Project_porosity`**.
2. Создайте в этом проекте Python-файл с именем **`DSS_porosity`**.
3. Поместите в текущую папку, т.е. в папку проекта **`Project_porosity`**, набор изображений (цветных и в оттенках серого) различных пористых материалов, приготовленный ранее.
4. Установите в проект **`Project_porosity`** следующие внешние библиотеки: NumPy, SQLite, **`PIL`** (**`Pyllow-PIL`**), OpenCV (cv2)..
5. В Python-файле **`DSS_ porosity создайте`** базу данных СППР, в которой будут записаны нормативные значения характеристик (пористость, допустимая площадь пор, а также допустимые для них средние квадратические отклонения), четырех пористых материалов. Эта база данных должна иметь следующую таблицу:

	ID	Наименование	Площадь поры	Откл. от площади	Пористость	Откл. от пористости
1	0	Материал2	12.0	5.0	0.1	0.01
2	1	Материал3	9.0	8.0	0.15	0.01
3	4	материал1	6.0	4.0	0.2	0.02

Для этого наберите и запустите на выполнение скрипт создания базы данных, например, из четырех материалов:

```
import numpy as np
import cv2
import sqlite3
import PIL.Image as Img
import PIL.ImageEnhance as Enhance

    rows = [
                (0, 'Материал2', 12.0, 5.0, 0.1, 0.01),
                (1, 'Материал3', 9.00, 8.0, 0.15, 0.01),
                (2, 'Материал4', 15.0, 8.0, 0.2, 0.5),
                (3, 'Материал5', 14.0, 7.0, 0.3, 0.7),
            ]
        conn = sqlite3.connect(self.db_name)
        cur = conn.cursor()
        cur.execute("""CREATE TABLE Materials
                            (ID    INTEGER NOT NULL PRIMARY KEY
                           AUTOINCREMENT, NAME TEXT,
                           PORE_AREA_MEAN REAL NOT NULL,
                           PORE_AREA_STD REAL NOT NULL,
                           POROUS_MEAN REAL NOT NULL,
                           POROUS_STD REAL NOT NULL
                           )""")
        cur.executemany("""INSERT INTO Materials values
(?,?,?,?,?,?)""", rows)
        conn.commit()
        conn.close()
```

Литература к работе

1. Python Imaging Library (Fork), https://pypi.org/project/Pillow/
2. Прохоренок, Н. А. Python 3. Самое необходимое: Пособие / Прохоренок Н.А., Дронов В.А. - СПб:БХВ-Петербург, 2016. - 464 с. ISBN 978-5-9775-3631-8. - Текст : электронный. - URL: https://codernet.ru/books/python/python_3_samoe_neobxodimoe_proxorenok/
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

ЛАБОРАТОРНАЯ РАБОТА 10.
СППР «ОПРЕДЕЛЕНИЕ КАЧЕСТВА ПОВЕРХНОСТИ ПОРИСТОГО МАТЕРИАЛА». Часть 3.

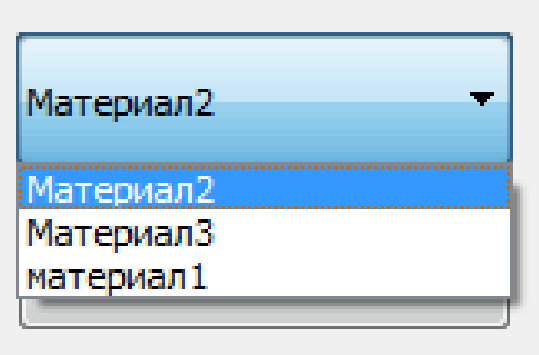
Цель работы: создание системы управления базой данных для СППР «Определение качества поверхности пористого материала».

Подготовка к выполнению задания.

1. На компьютере должен быть выход в интернет.
2. Откройте созданный на практическом занятии 9 проект **`Project_porosity`**.
3. Ознакомьтесь с приложением 1.

Задание 1. Создание системы управления БД СППР.

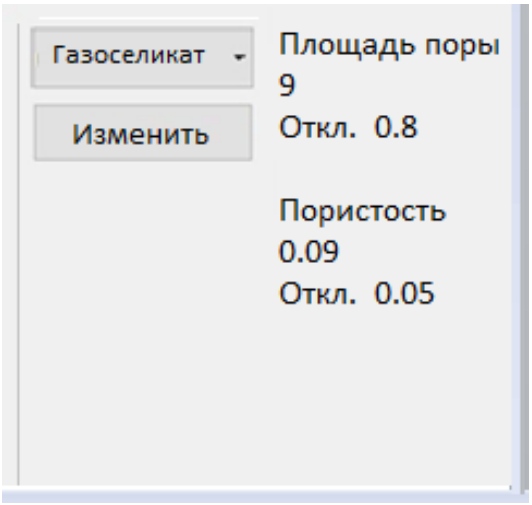
1. Откройте в проекте **`Project_porosity`** файл **`DSS_porosity`**, созданный на практическом занятии 9.
2. Напишите код для формирования списка и выбора из него нужного для анализа материала:



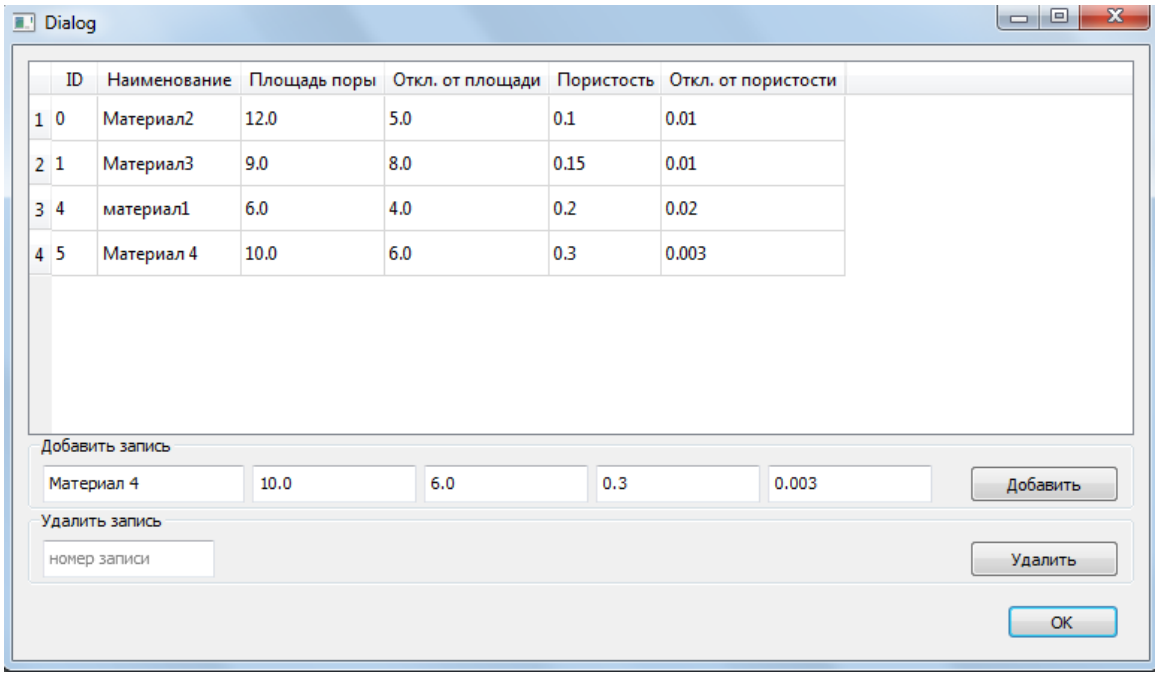
В данном виджете выбирается нужный нам материал из базы данных, уже заранее внесенный в систему со своим названием, порядковым номером и параметрами, для которого мы будем ставить эксперименты из окна СППР (см. практическое занятие 8). Она создается с помощью компонента QComboBox.

Напишите код такой, чтобы после выбора материала из этого списка считывались соответствующие нормативные значения и отображались справа

от этой кнопки (см. практическое занятие 8):



3. Напишите код для открытия и функционирования всех виджетов формы для изменения БД, которая должна открываться при нажатии на кнопку «Изменить» из окна СППР (см. практическое занятие 8). Лицу, принимающему решение, через поля ввода на ней должна быть предоставлена возможность изменения указанных выше допустимых значений средних квадратических отклонений. Так же должна быть возможность добавления образца материала, если его нет в базе данных СППР, а так же удаления записи из БД пористых материалов. Вид этой формы должен быть следующий:



Ниже представлен код обработки нажатия на кнопку добавления .

```
        def push_button_add_click(self):
        material_name =
self.text_edit_material_name.toPlainText()
        material_area =
self.text_edit_material_area.toPlainText()
        material_area_std =
self.text_edit_material_area_std.toPlainText()
        material_porous =
self.text_edit_material_porous.toPlainText()
        material_porous_std =
self.text_edit_material_porous_std.toPlainText()

        data = [material_name, material_area,
material_area_std, material_porous, material_porous_std]

        flag = [True if (m is not None and m != '') else
False for m in data ]

        if flag:
            try:
                material_area = float(material_area)
                material_area_std = float(material_area_std)
                material_porous = float(material_porous)
                material_porous_std =
float(material_porous_std)
                connect = sqlite3.connect(self.db_name)
                crsr = connect.cursor()
                crsr.execute("""INSERT INTO Materials(NAME,
PORE_AREA_MEAN, PORE_AREA_STD, POROUS_MEAN, POROUS_STD)
                VALUES (?,?,?,?,?)""", (material_name,
material_area, material_area_std,
                                        material_porous,
material_porous_std))
                connect.commit()
                connect.close()
                self.load_materials()
                self.fill_table()

            except Exception as e:
                print(e)
```

В данной форме представлена так же команда удаления записи из базы данных. Для удаления записи из БД ЛПР необходимо ввести в ячейку номер материала и нажать на кнопку удаления. После чего произойдет удаление строки из таблицы базы данных. Ниже представлен код обработки нажатия на кнопку «Удалить».

```
    def push_button_delete_click(self):
        index = self.text_edit_material_id.toPlainText()
        try:
            index = int(index) - 1
            if 0 <= index <= len(self.materials)-1:
                connect = sqlite3.connect(self.db_name)
                crsr = connect.cursor()
                row = self.materials.pop(index)
                id = row[0]
                crsr.execute('DELETE FROM Materials WHERE ID=?',
(id,))
                connect.commit()
                connect.close()
                self.load_materials()
                self.fill_table()
        except Exception as e:
            print(e)

        self.text_edit_material_id.setPlainText('')
        self.text_edit_material_id.clear()
```

Литература к работе

1. Python Imaging Library (Fork), https://pypi.org/project/Pillow/
2. Прохоренок, Н. А. Python 3. Самое необходимое: Пособие / Прохоренок Н.А., Дронов В.А. - СПб:БХВ-Петербург, 2016. - 464 с. ISBN 978-5-9775-3631-8. - Текст : электронный. - URL: https://codernet.ru/books/python/python_3_samoe_neobxodimoe_proxorenok/
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

ЛАБОРАТОРНАЯ РАБОТА 11.
СППР «ОПРЕДЕЛЕНИЕ КАЧЕСТВА ПОВЕРХНОСТИ ПОРИСТОГО МАТЕРИАЛА». Часть 4.

Цель работы: создание формы для улучшения качества изображения, предназначенного для анализа в СППР «Определение качества поверхности пористого материала».

Подготовка к выполнению заданий.

1. Откройте созданный на практическом занятии 9 проект **`Project_porosity`**.
4. Поместите в текущую папку набор изображений (цветных и в оттенках серого) различных пористых материалов.
5. Ознакомьтесь с приложением 1

Задание 1. Создание формы для улучшения качества изображения.

1. Откройте в проекте **`Project_porosity`** файл **`DSS_porosity`**.
2. Опишите метод, обрабатывающий изменение значений контрастности, яркости, резкости:

```
        def set_transformed_frame(self, image):
            image = Img.fromarray(image)
            image =
    Enhance.Contrast(image).enhance(self.contrast_slider.value()/10)
            image =
    Enhance.Brightness(image).enhance(self.brightness_slider.value()/10)
            image =
    Enhance.Sharpness(image).enhance(self.sharpness_slider.value()/10)
            image = np.array(image)
            self.transform_img = image
            image = gui.QImage(image.data, image.shape[1], image.shape[0],
    gui.QImage.Format_RGB888)
            self.transformed_frame.setPixmap(gui.QPixmap.fromImage(image))
```

2. Реализуйте улучшение изображения для анализа, которое должно размещаться в окне слева от движков управления контрастностью, яркостью, резкостью изображения (см. практическое занятие 8):

Для управления контрастностью, яркостью, резкостью исходного изображения используются такие компоненты библиотеки PyQt5, как QSlider. Задайте диапазоны возможных значений конрастности, яркости и резкости по примеру кода, приведенного ниже:.

```
self.contrast_slider = wdgts.QSlider(core.Qt.Horizontal)
self.contrast_slider.setRange(-200, 200)
self.contrast_slider.setTickPosition(wdgts.QSlider.TicksBothSides)
self.contrast_slider.setValue(10)
```

Литература к работе

1. Python Imaging Library (Fork), https://pypi.org/project/Pillow/
2. Прохоренок, Н. А. Python 3. Самое необходимое: Пособие / Прохоренок Н.А., Дронов В.А. - СПб:БХВ-Петербург, 2016. - 464 с. ISBN 978-5-9775-3631-8. - Текст : электронный. - URL: https://codernet.ru/books/python/python_3_samoe_neobxodimoe_proxorenok/
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

ЛАБОРАТОРНАЯ РАБОТА 12.
СППР «ОПРЕДЕЛЕНИЕ КАЧЕСТВА ПОВЕРХНОСТИ ПОРИСТОГО МАТЕРИАЛА». Часть 5.

Цель работы: Реализация создание формы для улучшения качества изображения, предназначенного для анализа в СППР «Определение качества поверхности пористого материала».

Подготовка к выполнению заданий.

1. Откройте созданный на практическом занятии 9 проект **`Project_porosity`**.
2. Ознакомьтесь с приложением 1.

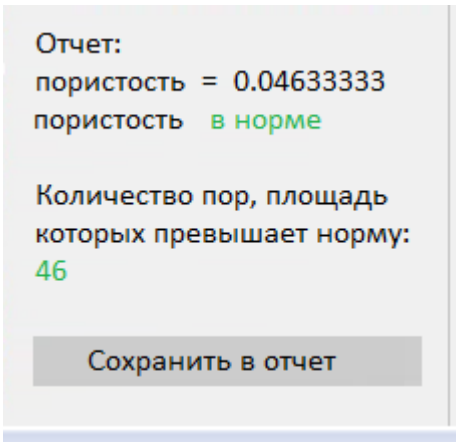
Задание 1. Реализация алгоритма определения пор и вывода результата.

1. Откройте в проекте **`Project_porosity`** файл **`DSS_porosity`**.
2. Теперь рассмотрим основную часть анализа изображения. Ниже представлен программный код конечной обработки изображения, обнаружения контуров пор, вычисления пористости и определения аномальных пор.

```
def explore(self, image):
        """
        Входной параметр:
        image - исследуемое изображение
        Выход:
        image - изображение с контурами пор
        area_c - отношение площади всех пор ко всей площади
изображения (пористость)
        len(bad_conrours) - количество 'плохих' пор
        """
        image = np.copy(image)
        # дополнительная обработка шумов
        blured = cv2.GaussianBlur(image, (5, 5), 0)
        # конвертация BGR формата в формат HSV
        hsv = cv2.cvtColor(blured, cv2.COLOR_BGR2HSV)

        lower_black = np.array([0, 0, 0])
```

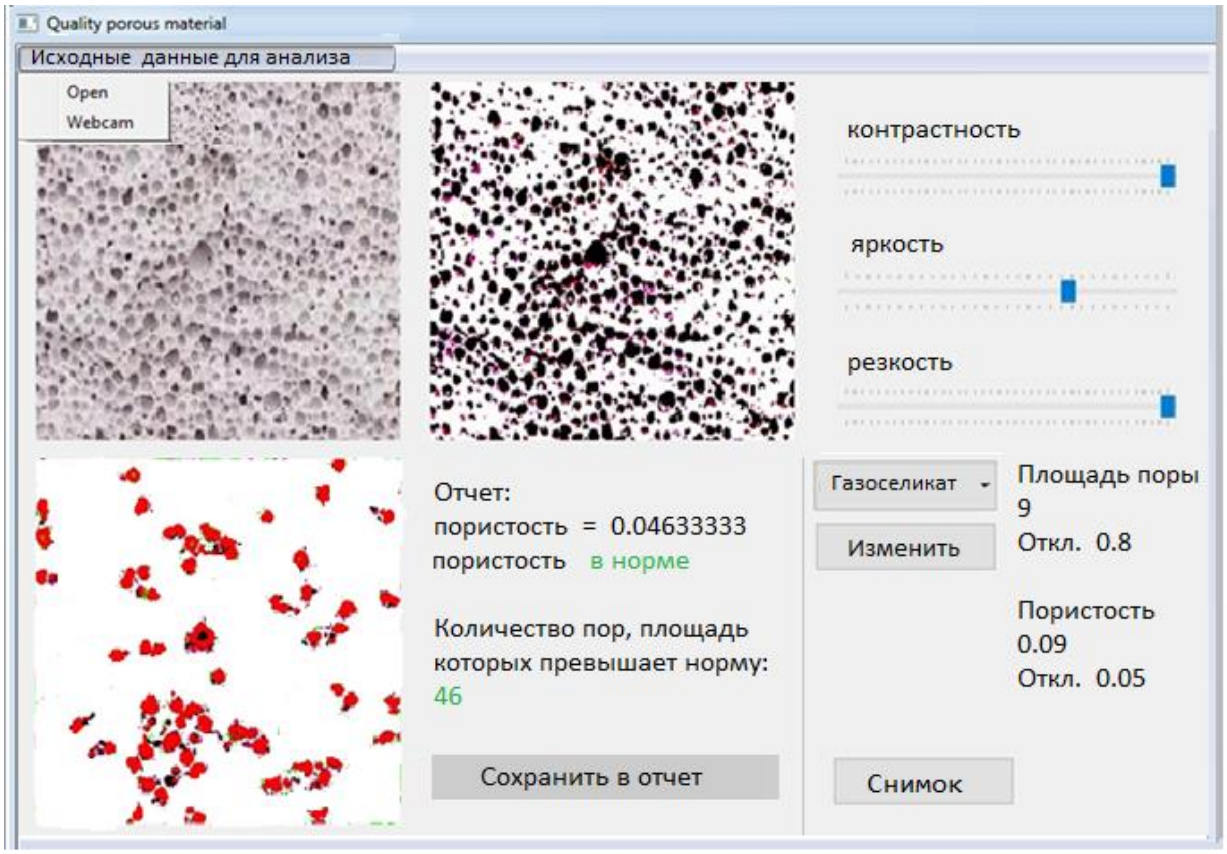
```
            upper_black = np.array([120, 120, 120])
            # определяем маску для обнаружения контуров пор.
            # будут выделены поры в заданном диапозоне
            mask = cv2.inRange(hsv, lower_black, upper_black)
            # получаем массив конутров
            _, contours, _ = cv2.findContours(mask,
cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

            good_contours = []
            bad_contours = []
            area_c = 0
            # находим поры, не превышающие нормативную площадь
            for contour in contours:
                # также подсчитываем общую площадь пор
                area_c += cv2.contourArea(contour)
                if self.mat_area - self.mat_area_std <=
cv2.contourArea(contour) <= self.mat_area +
self.mat_area_std:
                    good_contours.append(contour)
                else:
                    bad_contours.append(contour)
            area_c = area_c / (image.shape[0] * image.shape[1])
            # выделяем 'хорошие' поры зеленым цветом
            cv2.drawContours(image, good_contours, -1, (0, 255,
0), 3)
            # выделяем 'плохие' поры красным цветом
            cv2.drawContours(image, bad_contours, -1, (255, 0,
0), 3)
            return image, area_c, len(bad_contours)
```

Отчет по результатам анализа выводится в форму, представленную на рисунке:



Задание 2. Тестирование работы созданной СППР.

Откройте проект «Определение качества поверхности пористого материала»:



Продемонстрируйте работу созданной СППР на разных пористых материалах. Следите за правильностью выполнения системой каждого из задаваемых пользователем запросов, а также следите за динамичностью и интерактивностью работы созданной Вами СППР. Сделайте вывод о качестве работы алгоритма, заложенного в базу знаний созданной СППР.

Литература к работе

1. Пористость – определение, https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D0%B8%D1%81%D1%82%D0%BE%D1%81%D1%82%D1%8C
2. NEXSYS ImageExpert™ Sample 2. Программа для качественного анализа изображений структур методом сравнения с эталонными шкалами, http://www.nexsys.ru/nexsys_iesam2x.htm
3. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.
4. Валеев Т.Р. Создание СППР «Определение качества поверхности пористого материала» в C#.- Курсовая работа, 3 курс, 2017.
5. Комплекс видеоизмерительный для анализа микроструктур и макроструктур материалов «VESTRA Imaging System», http://latemi.ru/catalog/vestra-imaging-system/

ЛАБОРАТОРНАЯ РАБОТА 13.
РАСПОЗНАВАНИЕ ОБЪЕКТОВ НА КАРТИНКЕ

Цель работы: написать программу, распознающую объекты на картинке.

Подготовка к выполнению задания.

1. На компьютере должен быть выход в интернет.
2. Установите ImageAI и необходимые для ImageAI внешние библиотеки (см. приложение 2).
3. Зайдите на официальный сайт `GitHub`, скачайте модель `RetinaNet` и поместите эту модель в ту же папку, где находится **`main.py`.**

Ссылки для скачивания:

- **`ImageAI`** документация**:** https://imageai.readthedocs.io/en/latest/.
- **`ImageAI GitHub`** https://github.com/OlafenwaMoses/ImageAI/tree/master/imageai/Detection,

Задание. Распознавание объектов с картинки.

1. Подключите библиотеку **`imageai.Detection,`** т.е. наберите:

```
1    from imageai.Detection import ObjectDetection
```

Класс **`ObjectDetection`** позволит обнаруживать различные объекты на картинках.

2. Подключите библиотеку **`OS`** -это встроенная библиотека в **`Python`** . Она позволяет работать с операционной системой.

```
2    import os
```

3. Создайте новую переменную **`exec_path`** , в нее поместим **`getcwd()`** -эта функция позволяет определить путь к этому проекту- нам этот путь понадобится для того чтобы мы моги обнаружить различные объекты(

например тот же самый объект **RetinaNet** , который нам и потребуется для того чтобы мы могли обнаруживать различные объекты)

```
3
4       exec_path = os.getcwd()
```

4. Далее создайте на основе **ObjectDetection** несколько свойств(объектов).

Первым делом создаем переменную **detector** и в нее поместим класс **ObjectDetection()**, т.е. создаем объект на основе класса **ObjectDetection()**

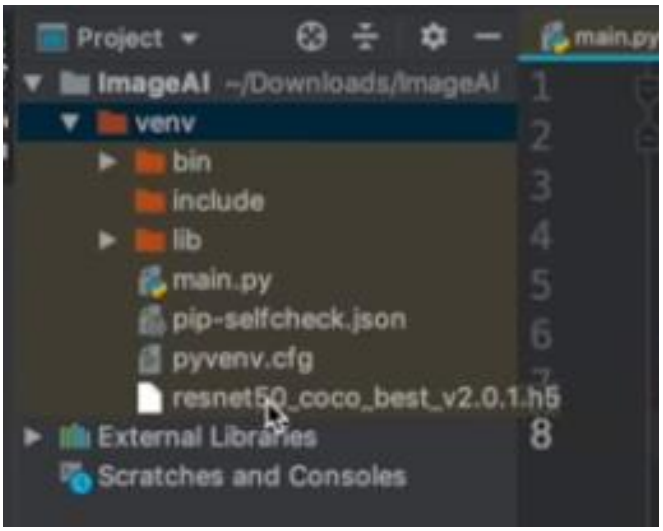
```
5
6       detector = ObjectDetection()
```

5. Далее берем **detector** и через точку обращаемся к различным функциям, которые находятся в классе **ObjectDetection.**

Первая функция это функция **setModelTypeAsRetiaNet** . Данная функция позволяет установить то, что мы используем RetiaNet модель для обнаружения объектов.

```
7       detector.setModelTypeAsRetinaNet()
```

Далее устанавливаем путь к нашей модели с именем **resnet50_coco_best_v2.0.1.h5** (см на рисунок ниже на этом рисунке стрелочкой указано это имя):



Для этого используем полный путь, т.е будем писать в коде **exec_path** (см. ниже), т.е. говорим, где вообще находится данный файл (в этой переменной находится полный путь к данному проекту), а дальше говорим, какой файл мне необходимо открыть, т.е. как бы открываем. Это будет наша модель:

```
detector.setModelPath(os.path.join(
    exec_path,"resnet50_coco_best_v2.0.1.h5")
)
```

6. Затем обращаемся опять к **detector** подгружаем данную модель (поэтому используем функцию **loadModel()**

```
detector.loadModel()
```

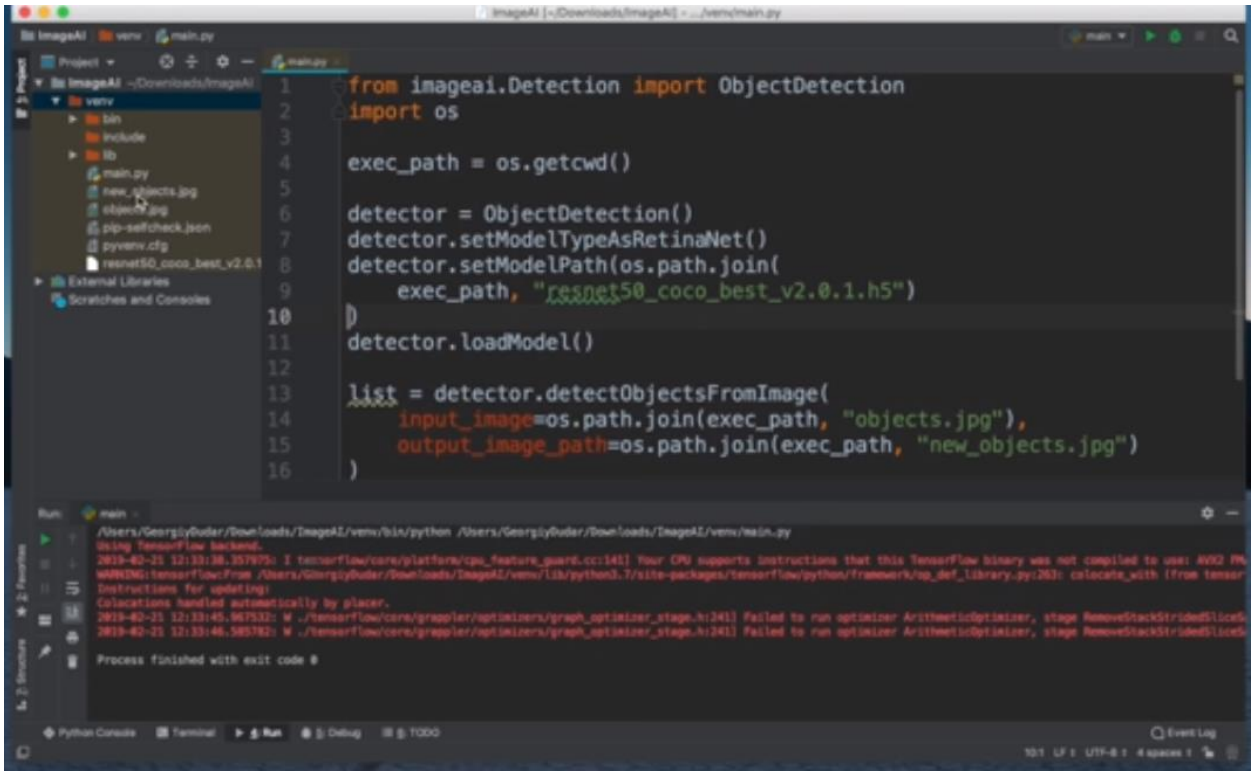
7. Дальше находим различные совпадения на какой либо картинке. Для этого создадим **list.** Это будет наш список, в который будут помещаться различные обнаружения. В него мы запишем следующее: мы берем **detector,** берем функцию **detectObjectsFormmage** (т.е. обнаружить объекты из изображения).

   Внутри указываем **input_image**, т.е. исходное изображение, с которого мы будем обнаруживать все объекты. Чтобы указать расположение этого изображения используем модуль **OS**, дальше используем переменную **path.** Дальше используем функцию **join,** т.е. объединяем несколько строк между собой: **elec_path** - это полный путь к данному проекту. Указываем название картинки, которую хотим открыть **objects.jpg**.

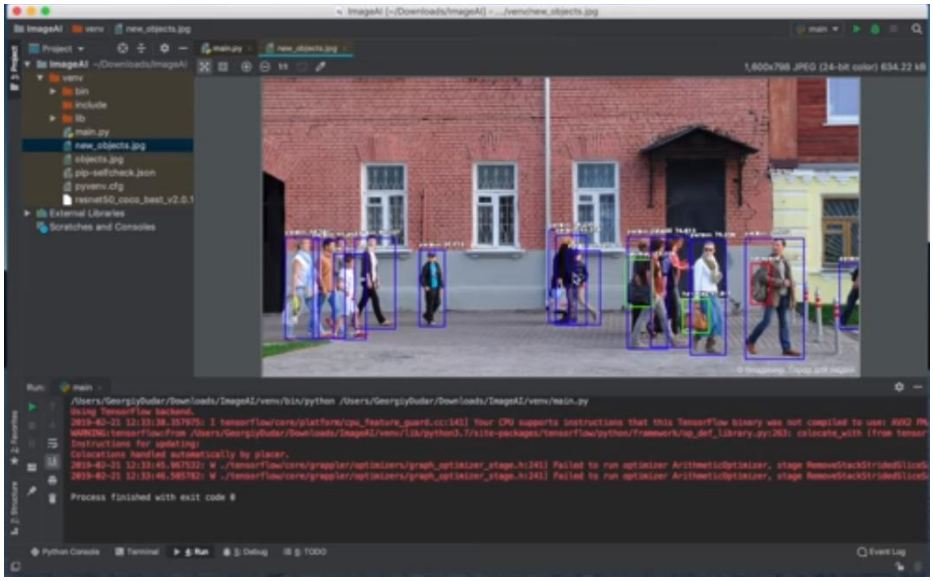
   Путь **output_image_path** - это то место, куда мы сохраним полученную фотографию. Пишем знак равно: =, и копируем из предыдущей строчки, меняя только название файла **nev_Objects.jpg** , в который сохраняем.

```
12
13  list = detector.detectObjectsFromImage(
14      input_image=os.path.join(exec_path, "objects.jpg" ),
15      output_image_path=os.path.join(exec_path, "nev_objects.jpg")
16  )
```

Запустим на решение. Немного ждем:



Процесс закончился. Видим, что у нас создался новый файл. Если откроем его, то увидим на изоброажении выделенные программой объекты:



Все те объекты, которые были найдены, обведены в рамку и прописано (в процентах) на сколько они подходят под определенный объект.

Можем еще дописать дополнительные характеристики:

```
13  list = detector.detectObjectsFromImage(
14      input_image=os.path.join(exec_path, "objects.jpg" ),
15      output_image_path=os.path.join(exec_path, "nev_objects.jpg"),
16      minimum_percentage_probability=30,
17      display_percentage_probablity=False,
18      display_object_name=False
19  )
```

16-я строка - это минимальный процент точности. Т.е. то, что ниже числа, указанного после знака равенства, не будет выделяться в рамку (true-истина т. е. будет указываться, false-не будет указываться).

Запустим дополненную програму на решение. Немного ждем.

Когда процесс обработки закончится увидим, что у нас создался новый файл.

Если откроем его, то увидим:

Литература к работе

1. Official English Documentation for ImageAI! — ImageAI 2.1.5, https://imageai.readthedocs.io/en/latest/
2. ImageAI GitHub https://github.com/OlafenwaMoses/ImageAI/tree/master/imageai/Detection
4. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

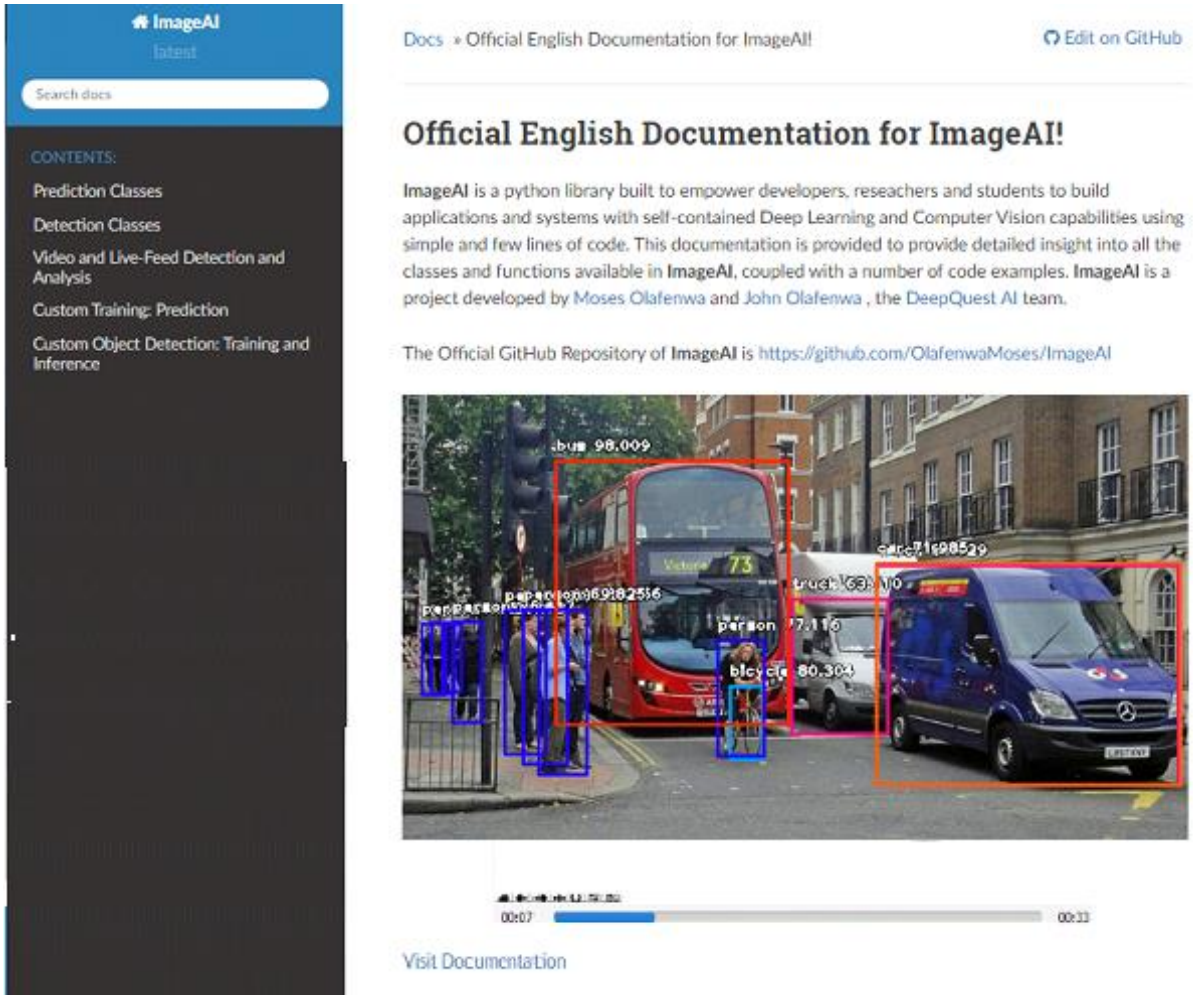
ЛАБОРАТОРНАЯ РАБОТА 14.
ОТСЛЕЖИВАНИЕ ОБЪЕКТОВ НА ВИДЕО

Суть работы: Написать программу для отслеживания объектов на видео.

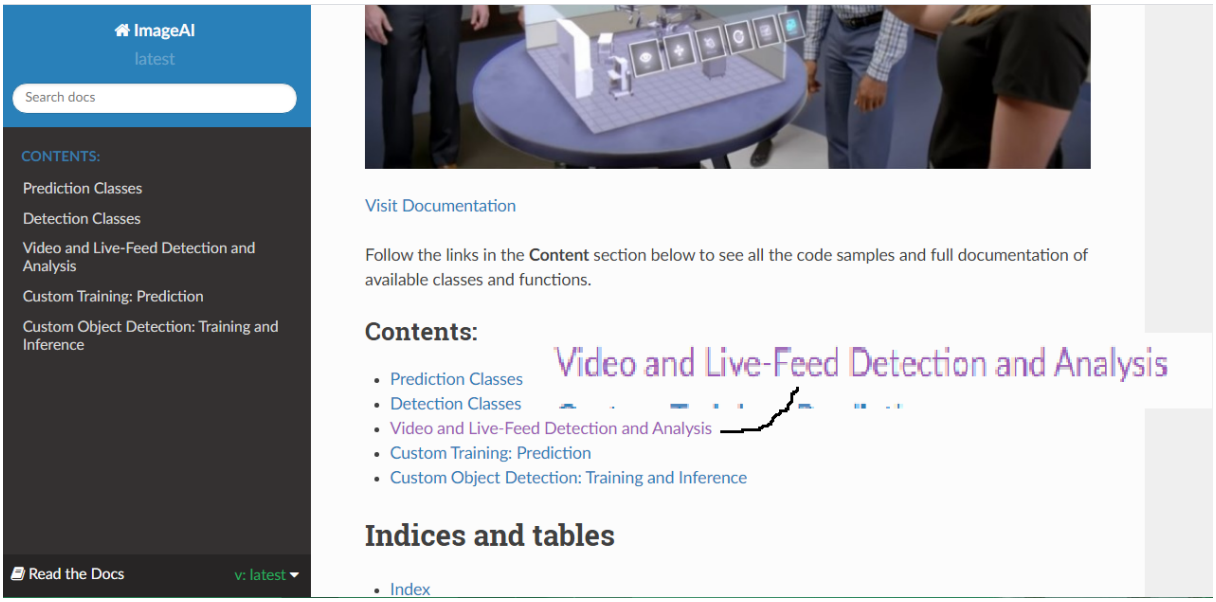
Подготовка к выполнению задания.

Скачайте на рабочий стол файл **`yolo.h5`** (см. п.1 ниже) ***и*** любой файл с видео небольшой продолжительности (12-15 секунд), дайте ему имя **`traffic.mp4`** (см. п2 ниже).

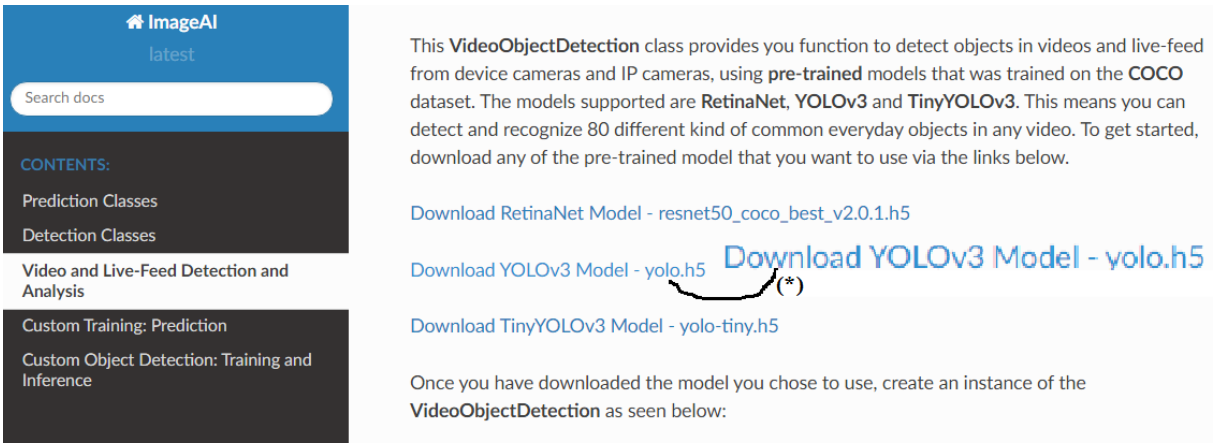
п.1 Перейдем на официальный сайт документации по **`ImageAI.`** Ссылка на официальный сайт: https://imageai.readthedocs.io/en/latest/. Выйдем на страницу:



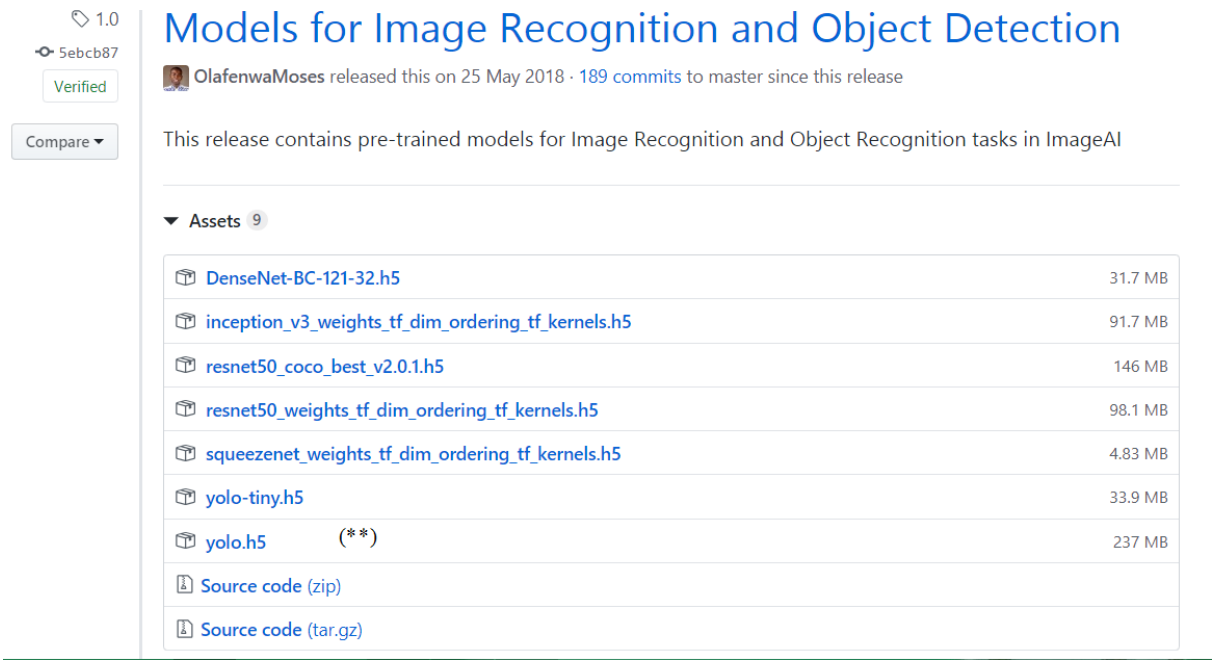
Прокрутите эту страницу сайта еще ниже, пока не увидите:



Кликните по выделенной строке (см. предыдущий скрин) получим:



Можно кликнуть все три строчки. Для начала скачайте, **`YOLOV3 Model`**. Кликнув на строку, помеченную на скришоте символом **(*)),** получим:



Найдите в этом списке **`yolo.h5`** (см. (**)), скачайте его на рабочий стол.

п.2 Скачайте любой файл с видео небольшой продолжительности (12-15 секунд) и дайте ему имя **`traffic.mp4`**.

Закоментируйте код предыдущего практического занятия. Это строки 1-19. Ниже продолжите написание кода для видео. Теперь можно приступать к набору кода для настоящей работы.

Задание. Отслеживание объектов на видео.

1. Подключите библиотеку **`imageai.Detection.`** Класс **`VideoObjectDetection`** позволит обнаруживать различные объекты на видео.

```
20
21  from imageai.Detection import VideoObjectDetection
```

2. Подключите библиотеку **`OS`** - это встроенная библиотека в **`Python`** . Она позволяет работать с операционной системой.

```
22      import os
```

3. Создайте новую переменную **execution_path**, в которой указывается путь к данному проекту:

```
23
24      execution_path = os.getcwd()
```

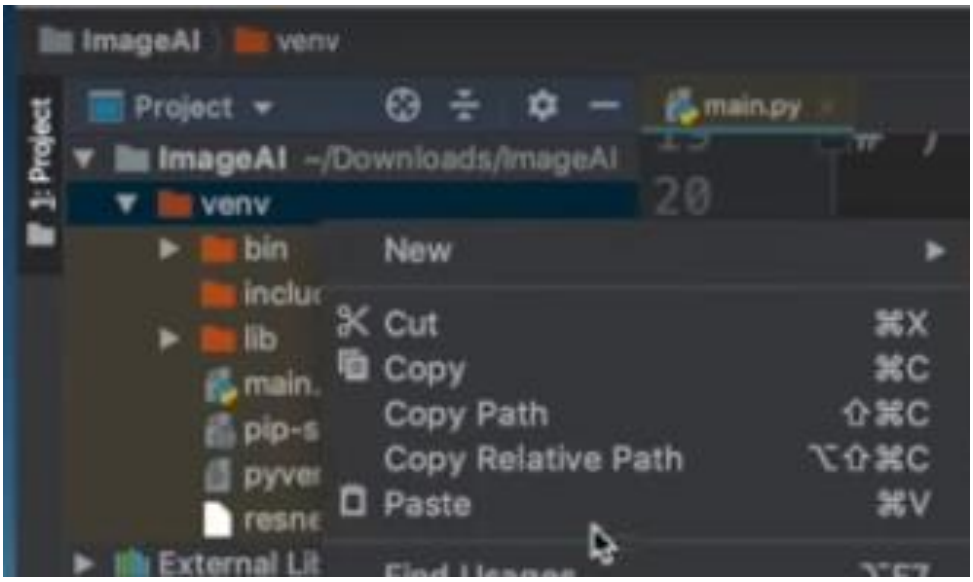
4. Создайте на основе класса **VideoObjectDetection** определенный объект **detector**

```
25
26      detector = VideoObjectDetection()
```

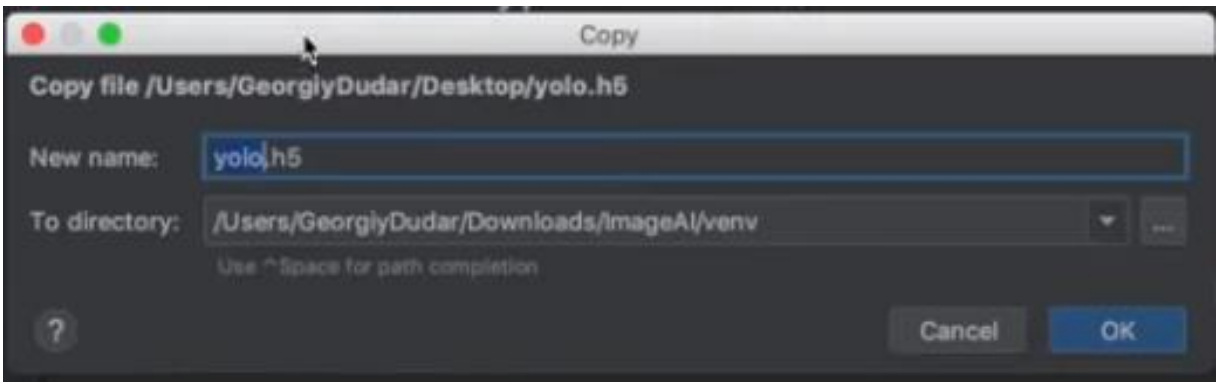
5. Наберите команду

```
27      detector.setModelTypeAsYOLOv3()
```

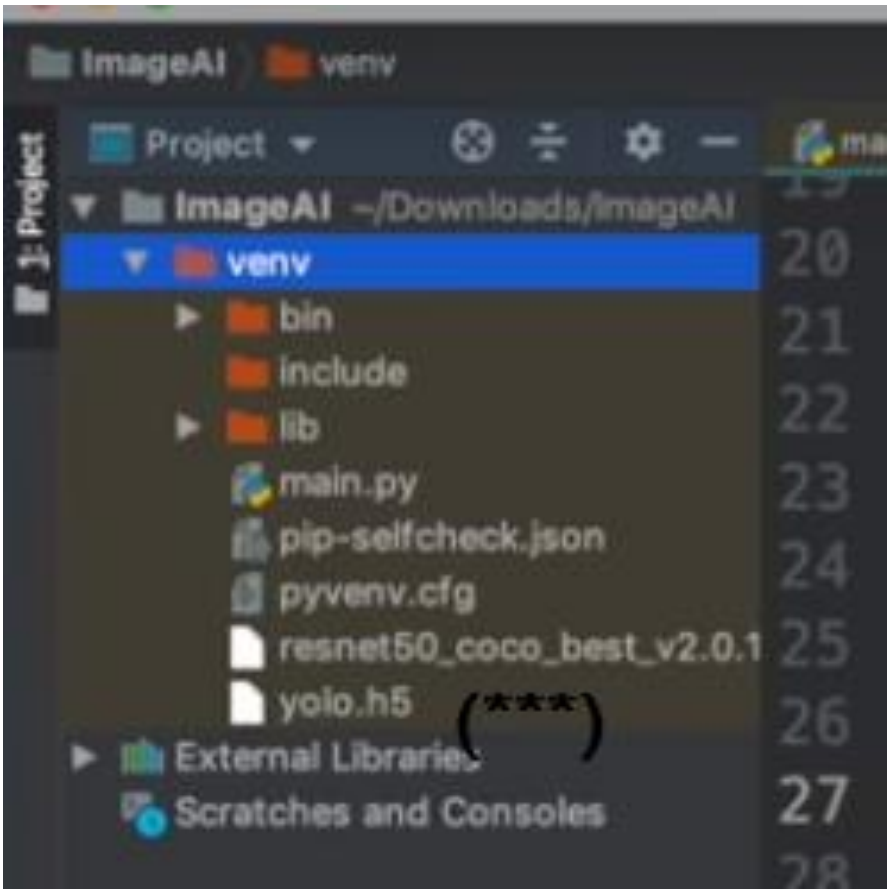
Здесь используем модель **YOLOv3()**, причем это та модель, которая была скачена нами ранее на рабочий стол. Давайте ее сразу скопируйте с рабочего стола и вставьте в сам проект (т.е. идем по цепочке `venv-Paste`):



Вставьте теперь имя этого файла в поле ввода:



Затем нажимаем кнопку **ок.** Увидим, что в текущей папке появился этот файл **yolo.h5** (см. ниже (***))



Как видите, мы эту модель находим в данном проекте, т.е. по пути к данному проекту. Поэтому в коде пишем:

```
28    detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
```

На этом скриншоте в строке 28 после запятой видим " **yolo.h5**" - это сам этот объект **yolo.h5**, его мы добавили в текущую папку (см.(***) на скриншоте перед последним скриншотом).

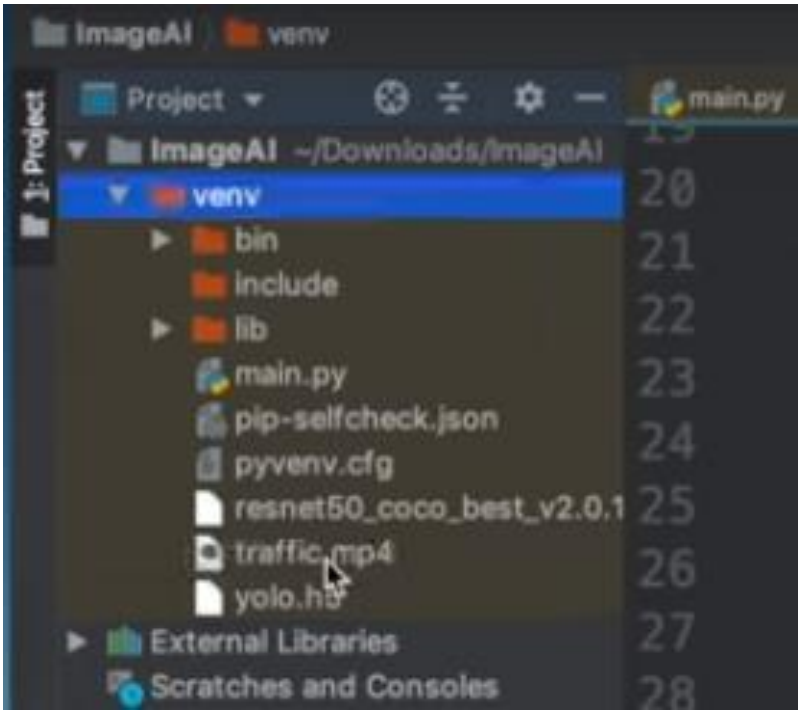
6. Дальше подгружаем саму модель

```
29    detector.loadModel()
```

7. Затем создаем список кадры (frames).

```
30
31  video_path = detector.detectObjectsFromVideo(
32      input_file_path=os.path.join(execution_path, "trafic.mp4"),
33      output_file_path=os.path.join(execution_path, "trafic_detected"),
34      frames_per_second=20,
35      log_progress=True
36  )
```

3десь через функцию **detectObjectsFormVideo** говорим с какого файла мы все берем (см.в коде `input_file_path=`). В данном случае мы берем с файла **trafic.mp4**. Это видео, скаченное нами ранее на рабочий стол. Скопируйте его в текущую папку нашего проекта. Видим, что файл **trafic.mp4** добавился:



Далее говорим, в какой файл будем это все конвертировать (см.в коде `output_file_path=`). Здесь мы все это конвертируем в файл **trafic_detected.mp4**. Названия можем прописывать любые,

например назвали здесь **"trafic_detected"**.

Дальше говорим сколько будет кадров (frames) в секунде. Мы устанавливаем 20.

Дальше мы говорим, что `все различные логи, т.е то как программа обрабатывает наше видео, будут выписываться в консоль.

Эта же команда крупнее:

```
video_path = detector.detectObjectsFromVideo(
    input_file_path=os.path.join(execution_path, "trafic.mp4"),
    output_file_path=os.path.join(execution_path, "trafic_detected"),
    frames_per_second=20,
    log_progress=True
)
```

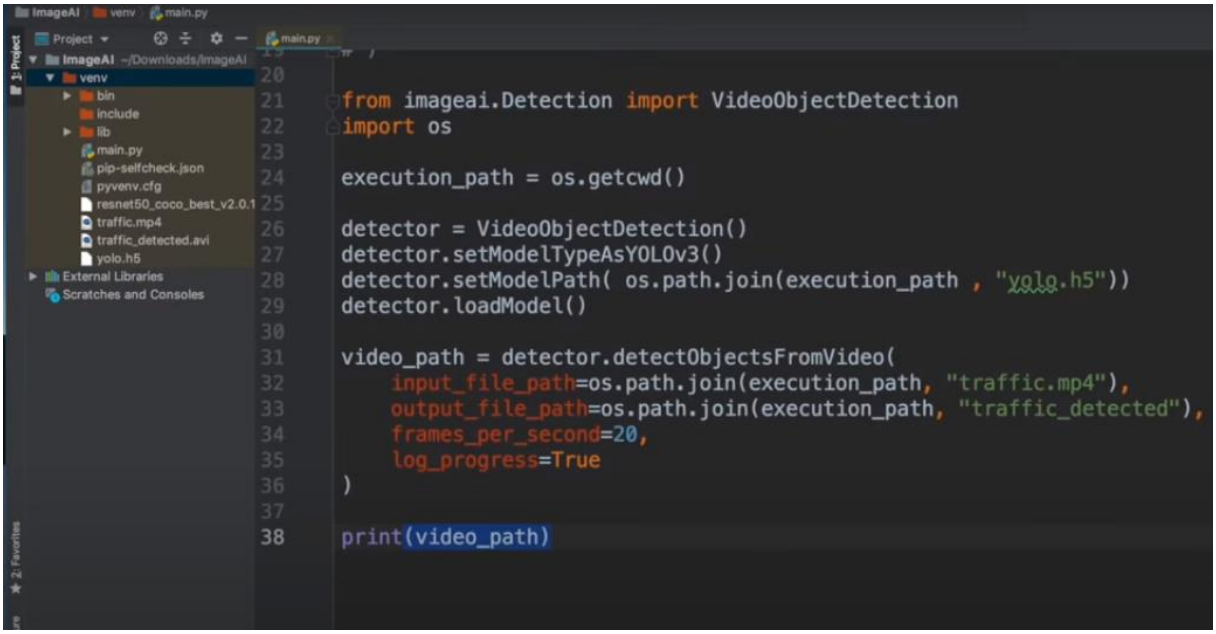
8. Далее выписываем **video_path** в саму консоль

```
37
38  print(video_path)
```

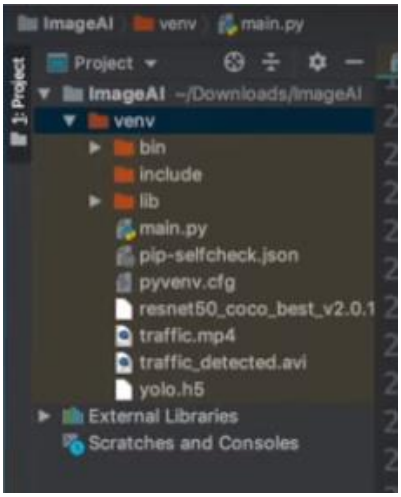
9. Теперь запустите программу на выполнение.

```
21  from imageai.Detection import VideoObjectDetection
22  import os
23
24  execution_path = os.getcwd()
25
26  detector = VideoObjectDetection()
27  detector.setModelTypeAsYOLOv3()
28  detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
29  detector.loadModel()
30
31  video_path = detector.detectObjectsFromVideo(
32      input_file_path=os.path.join(execution_path, "trafic.mp4"),
33      output_file_path=os.path.join(execution_path, "trafic_detected"),
34      frames_per_second=20,
35      log_progress=True
36  )
37
38  print(video_path)
```

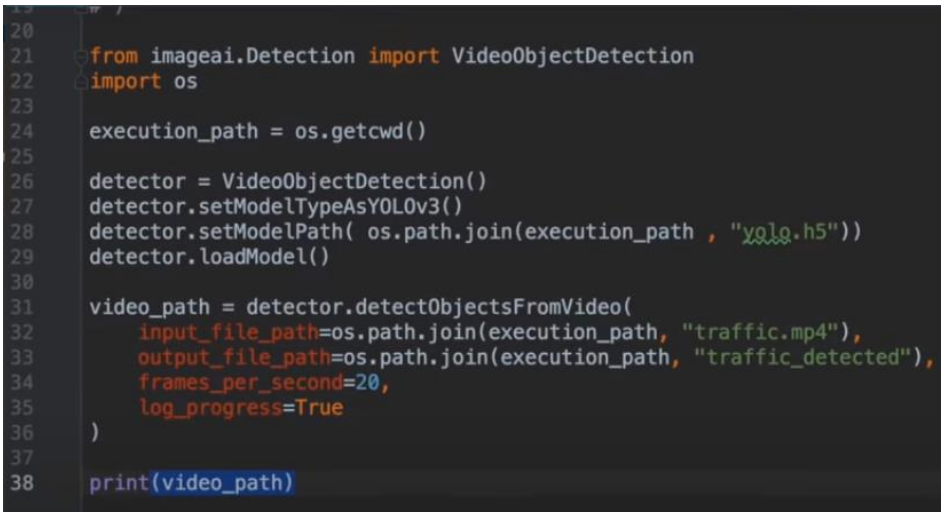
В итоге проект должен выглядеть следующим образом:



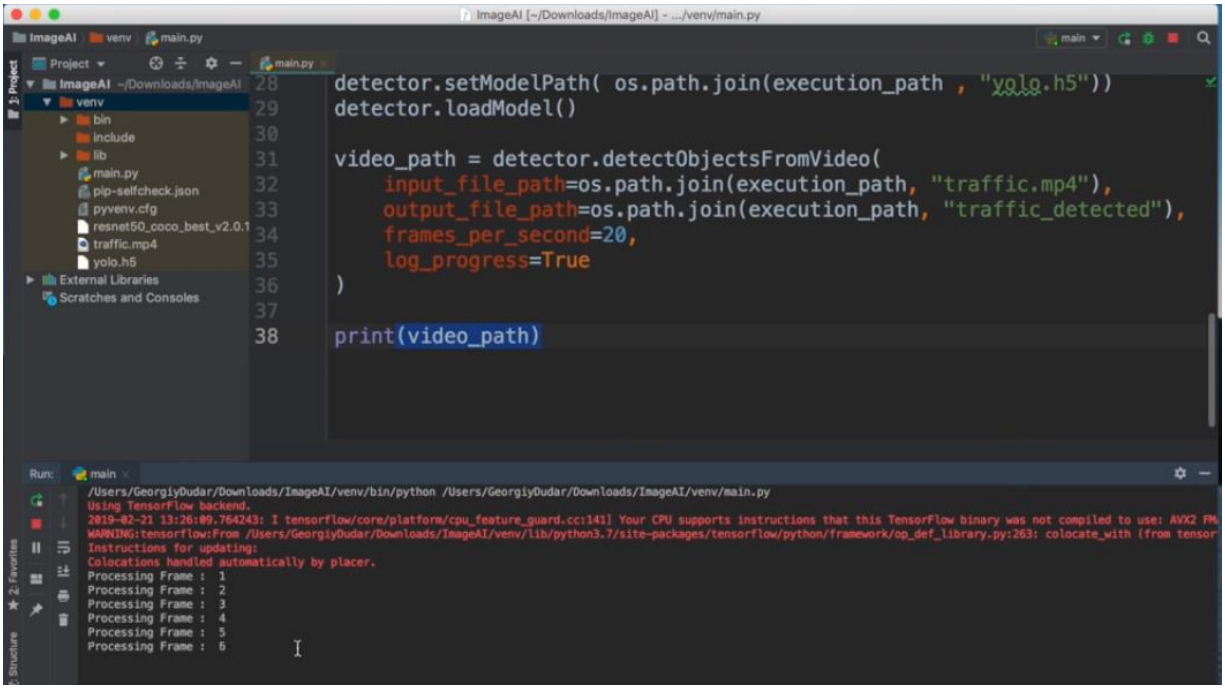
Левая часть окна, которое приведено выше, крупнее:



Правя часть этого же окна, которое приведено выше, тоже крупнее:



10. После запуска этой программы видим, что видео обрабатывается. И обрабатывается покадрово:



Когда процесс закончится увидим, что у нас создался новый файл. Откроем его. Увидим, что объекты на видео распознаются и берутся в рамку.

Таким образом, при помощи данной библиотеки мы можем распознавать объекты, как на видео, так и просто на фотографиях.

Литература к работе

1. Official English Documentation for ImageAI! — ImageAI 2.1.5, https://imageai.readthedocs.io/en/latest/.
2. Шустова К.П. Программные методы обработки изображений и распознавания образов. Практикум / учебное пособие. – Казань: Казанский гос. энерг. ун-т, 2020.–82 с.

Приложение 1.

Методы и классы Python, используемые для создания СППР «Определение качества поверхности пористого материала»

В программе использованы следующие методы для работы с изображениями:

cv2.VideoCapture - метод для захвата видео;

cv2.cvtColor – метод конвертации цветовых каналов изображения;

cv2.cvtGaussianBlur -метод удаления шумов с изображения;

cv2.findContours - метод поиска контуров. Поиск контуров использует для работы монохромное изображение и работает так, что все пиксели картинки с ненулевым цветом будут интерпретироваться как 1, а все нулевые останутся 0;

cv2.contourArea - метод задающий площадь контура;

cv2.drawContours - метод изображающий контуры в кадре.

В программе использовать следующие классы:

PIL.Image.open - класс, представляющий объект изображения;

PIL.ImageEnhance.Contrast - класс, изменяющий контрастность изображения;

PIL.ImageEnhance.Brightness - класс, изменяющий яркость изображения;

PIL.ImageEnhance.Sharpness - класс, изменяющий резкость изображения;

В программе для работы с БД использовать следующие методы:

sqlite3.connection;

sqlite.connection().cursor;

sqlite.connection().cursor().execute.

Приложение 2.

Последовательность установки

*библиотек для работы с **ImageAI**.*

- **Python** 3.5.1 или выше, скачать Python
- **pip3 ,** скачать PyPi
- **Tensorflow** 1.4.0 или выше

```
pip3 install --upgrade tensorflow
```

или

```
pip3 install tensorflow
```

т.к. upgrade - Обновить

- **Numpy 1.13.1 or higher**

```
pip3 install numpy
```

- **SciPy.191 or higher**

```
pip3 install scipy
```

- **OpenCV**

```
pip3 install opencv-python
```

- **Pillov**

```
pip3 install pillov
```

- **Matplotlib**

```
pip3 install matplotlib
```

- **h5py**

```
pip3 install h5py
```

- **Keras**

```
pip3 install keras
```

- **ImageAI**

```
pip3 install https://qithub.com/OlafenwaMoses/ImageAI/releases/down
load/2.0.2/imageai-2.0.2-py
```

(1)

СОДЕРЖАНИЕ

Шустова Евгения Петровна

ВВЕДЕНИЕ В АНАЛИЗ ИЗОБРАЖЕНИЙ НА PYTHON

П Р А К Т И К У М

электронный образовательный ресурс

Корректура автора

Казанский (Приволжский) федеральный университет
Казань, ул. Кремлевская, д.18

88