Обработка изображений

Во-первых, люди, увлекающиеся фотографией, — едва ли не самое многочисленное полупрофессиональное сообщество в мире. Его популярности очень способствует распространение смартфонов и сервисов по работе с фотографиями, таких как Instagram и Pinterest.

Во-вторых, работа с видео сводится к работе с отдельными изображениями. Это относится и к профессиональным техникам наложения фильтров, и даже к работе с хромакеем, без которой не обходится практически ни один современный фильм.

В-третьих, модель представления изображения в памяти компьютера довольно проста. Почти всегда это многомерный массив целых чисел. Даже на начальном этапе изучения программирования эта область интересна как для обучения, так и для применения на практике.

Пока мы оставим за кадром вопросы скорости обработки изображений. С ними можно поэкспериментировать самостоятельно, это позволит обсудить скорость выполнения компилируемого и интерпретируемого кода. Кстати, для замеров времени тоже есть модуль — `timeit`.

Растровые изображения

Мы будем работать с растровыми изображениями, представляющими собой массив (таблицу) пикселей разных цветов.

Давайте посмотрим вот на это изображение.

Если мы приблизим его, увидим пиксели — минимальные единицы изображения, для которых можно определить цвет. Давайте увеличим глаз совы (кстати, ее зовут Рианна).

Итак, изображение можно моделировать списком списков (двумерной таблицей, в которой лежат цвета). Осталось только подумать, как именно кодировать цвета.

Опыт работы со строками, где каждому символу соответствует свой код, должен подсказывать вам, что и с изображениями должно быть так же. Мы можем пронумеровать некоторое количество цветов и указывать их номера в нашем списке списков. Совокупность выбранных цветов будет называться **палитрой**.

В итоге нам нужен способ преобразования цветов в целые числа.
Мы воспользуемся одной из самых популярных моделей представления цвета — RGB (Red, Green, Blue).

Модель RGB

В модели RGB каждый из цветов представляется совокупностью трех компонентов: красного, синего и зеленого. Значение каждого компонента лежит в диапазоне от 0 (минимум) до 255 (максимум), занимая 1 байт в памяти.

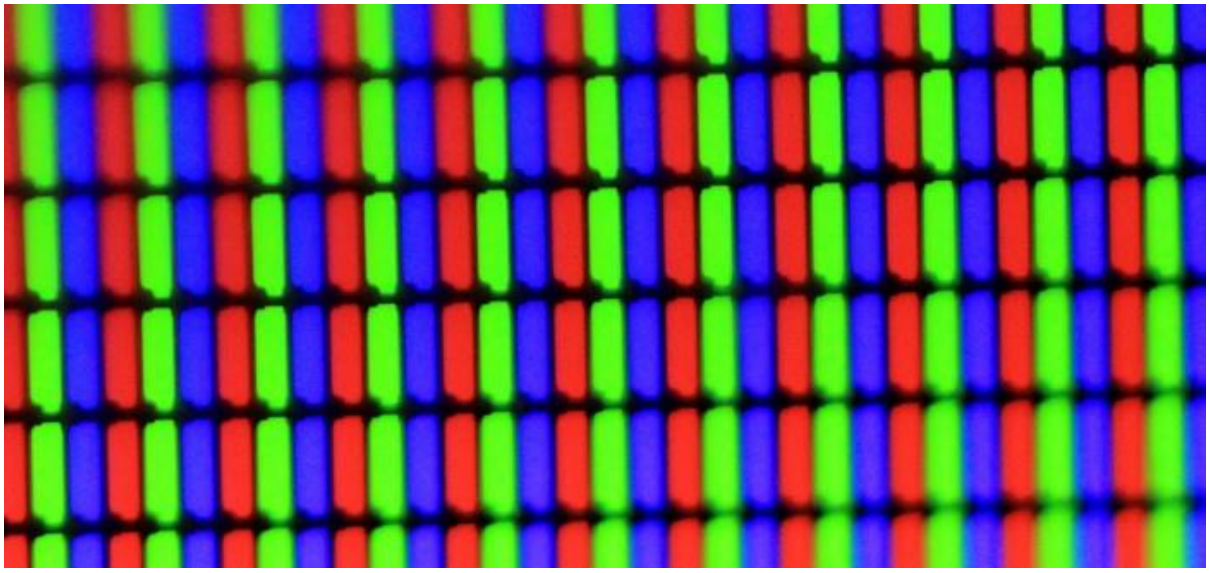
На самом деле модели хранения этих байтов в памяти Python и файле с картинкой бывают очень сложными — например, со сжатием. Однако мы будем работать с исходными, «чистыми» данными.

Итак, каждый цвет — совокупность трех целых чисел (в Python ее можно представить кортежем или списком). Кстати, сумма этих трех чисел говорит о яркости пикселя: чем сумма больше, тем пиксель кажется ярче. На самом деле и тут все сложнее, чем кажется: яркость каждого компонента для глаза не одинакова, однако примем это упрощение.

Например, (0, 0, 0) — черный цвет. Его яркость минимальна, оттенков нет.

- (255, 255, 255) — белый, максимальная яркость;
- (255, 0, 255) — очень насыщенный пурпурный (красный + синий);
- (255, 255, 0) — ярко-желтый (красный + зеленый);
- (100, 100, 100) — серый.

Красный, зеленый и синий выбраны в качестве основных цветов из-за особенностей цветовой чувствительности рецепторов нашего глаза. Кстати, если мы сильно увеличим матрицу смартфона или монитора, который светит чистым белым светом, увидим что-то вроде этого:

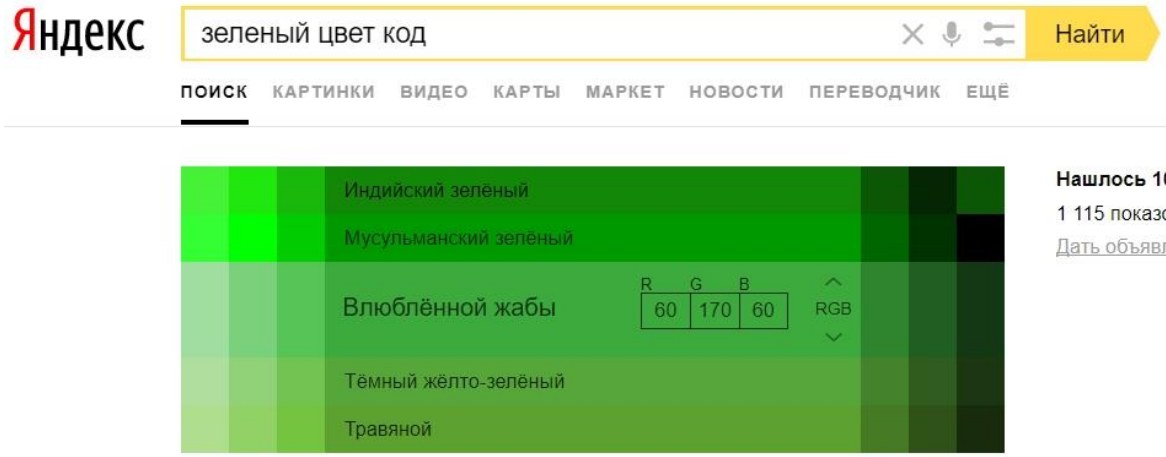


Да-да, это (255, 255, 255).

Итак, для нас изображение — список списков, элементами которого будут кортежи цвета.

Кстати, легко заметить, что в нашей модели всего 256×256×256 = 16777216 разных цветов. Этого вполне достаточно, чтобы человеческий глаз не замечал дискретности (конечного числа оттенков) цветовой модели.

У Яндекса есть специальный барабан, который позволяет знакомиться с оттенками цвета, подбирать цвета и узнать их коды:



PIL. Установка библиотек

Для работы с изображениями мы будем использовать библиотеку `PIL (Python image library)`, а точнее, ее модификацию под названием `Pillow`.

Установка пакетов

Для установки пакетов в Python служит специальная утилита командной строки `pip`, которая является еще и модулем.

Чтобы установить пакет, нужно выполнить команду `pip install <Имя модуля>`. Пакет будет скачан с PyPI и установлен, вы увидите примерно следующее:

```
c:\Python39\Scripts>pip install pillow

Collecting pillow

  Downloading Pillow-4.0.0-cp34-cp34m-win32.whl (1.2MB)

    100% |################################| 1.2MB 485kB/s

Collecting olefile (from pillow)

  Downloading olefile-0.44.zip (74kB)

    100% |################################| 81kB 1.7MB/s

Installing collected packages: olefile, pillow

  Running setup.py install for olefile ... done

Successfully installed olefile-0.44 pillow-4.0.0


c:\Python39\Scripts>%%
```

Может случиться так, что утилита pip не выполнится, тогда необходимо перейти в директорию `Scripts` вашей версии `Python`. Например, она может быть такой: `C:\Python39\Scripts`

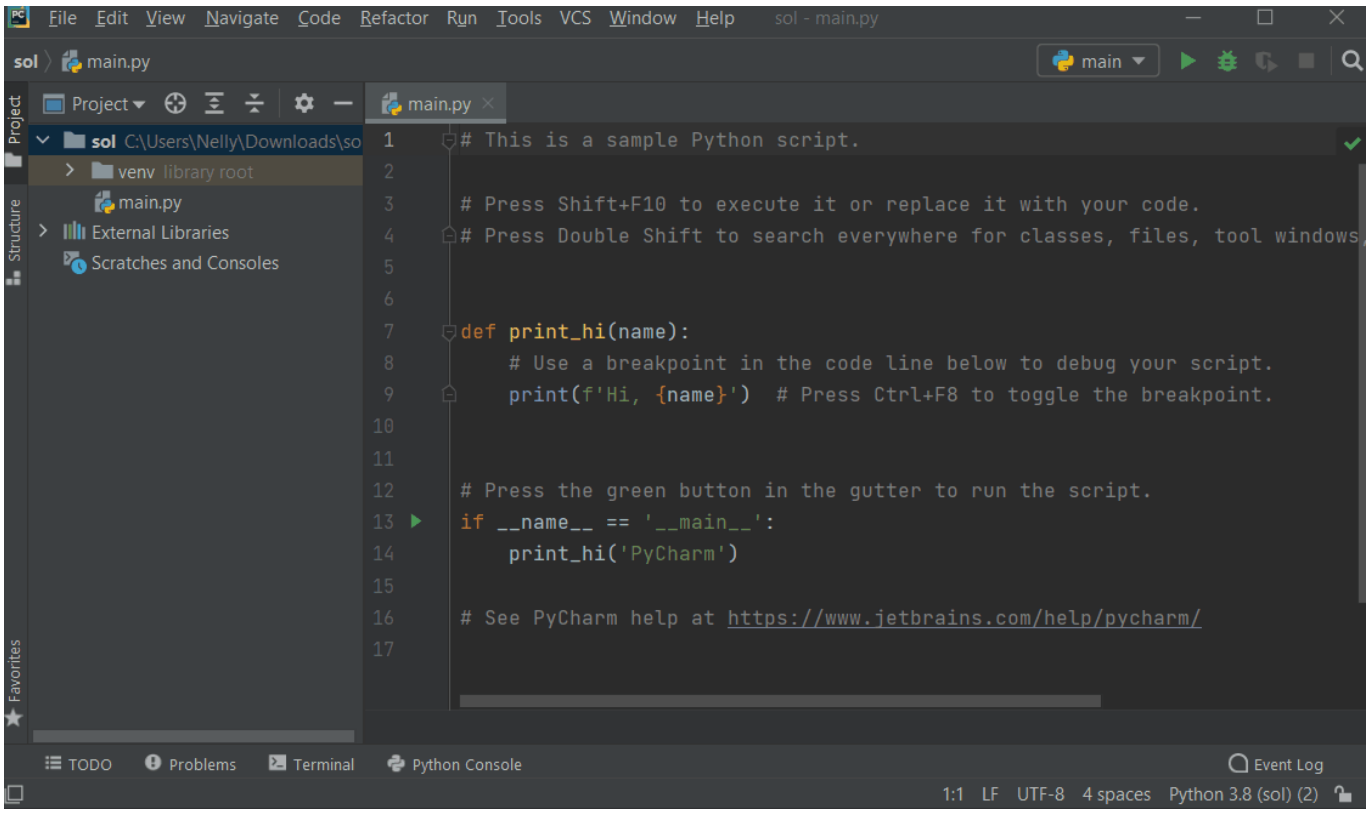
Кроме опции `install` в `pip`, доступны команды:

```
Usage:

  pip  [options]


Commands:

  install             Install packages.

  download            Download packages.

  uninstall           Uninstall packages.

  freeze              Output installed packages in requirements
format.

  list                List installed packages.

  show                Show information about installed packages.

  check               Verify installed packages have compatible
dependencies.

  search              Search PyPI for packages.

  wheel               Build wheels from your requirements.

  hash                Compute hashes of package archives.

  completion          A helper command used for command completion.

  help                Show help for commands.
```

Pillow — не чисто питоновская библиотека, она написана частично на языке C. Поэтому для некоторых версий Python может потребоваться компиляция кода доступным в системе C-компилятором, потому что `pip` сможет скачать только исходные коды библиотеки. Если такого компилятора нет (такое обычно бывает в windows-системах), стоит поискать скомпилированные версии в Интернете (готовые к установке файлы имеют расширение `.whl`). Например, множество популярных библиотек можно найти на странице сайта лаборатории флуоресцентной динамики Калифорнийского университета.

Также чтобы не задумываться о сложностях при установке библиотек, можно установить дистрибутив Anaconda. В нем есть все необходимые библиотеки Python. И не только они.

Необходимую библиотеку можно установить прямо из среды программирования `PyCharm`. Для этого откройте `Terminal`, перейдите в папку `venv/Scripts` своего проекта и запустите установку:



Модельный пример

Рассмотрим пример работы с изображением, в котором мы:

1. Пройдем по каждому пикселю в изображении.
2. Получим для него значение цвета в RGB-нотации.
3. Присвоим этому пикселю новое значение цвета (поменяем составляющие).
4. В конце сохраним получившееся изображение с новым именем.

Начальное изображение в этом примере никак не меняется, но от него можно отталкиваться в дальнейшей работе.

Итак, приступим.

Для работы нам потребуется файл с изображением — `riana.jpg`, который нужно сохранить в тот же каталог, где будет лежать программа по его обработке.

```
from PIL import Image


im = Image.open("riana.jpg")

pixels = im.load() # список с пикселями

x, y = im.size # ширина (x) и высота (y) изображения
```

```
for i in range(x):
    for j in range(y):
        r, g, b = pixels[i, j]
        pixels[i, j] = g, b, r

im.save("riana2.jpg")
```

Для работы с изображением нам нужен объект `Image`, который находится в библиотеке `PIL` (пишется большими буквами).

Функция open

Мы открываем изображение с диска функцией `open`.
В функции `open` в скобках указывается или абсолютный путь к файлу, или просто имя файла, если файл размещен в том же каталоге, что и сама программа.

Потом получаем список пикселей этого изображения, используя функцию `load`. Ее применяем к объекту, загруженному в переменную `im`. После применения функции получаем двумерный список, где для каждого пикселя хранится кортеж — цвет пикселя в палитре RGB.

Важно!

Обратите внимание: `pixels` устроен так, что индексация в нем идет кортежами, поэтому здесь запись `pixels[i, j]`, а не `pixels[i][j]`, что, возможно, было бы удобнее и привычнее. Это особенность библиотеки: создателям показалось, что так будет архитектурно уместнее.

С помощью атрибута `size` объекта `im` мы можем получить размер изображения, который хранится в виде кортежа: сначала ширину, потом высоту изображения в пикселях, что соответствует размерности `pixels`.

Далее переберем все элементы `pixels` (двумя циклами `for`) и для каждого элемента получим значение трех компонентов цвета. Запишем в массив `pixels` эти значения, но изменив порядок значений.

Для получения трех компонентов цвета каждого пикселя мы используем множественное присваивание, поэтому пишем

```
r, g, b = pixels[i, j]
```

вместо

```
pixel = pixels[i, j]
r = pixel[0]
g = pixel[1]
```

```
b = pixel[2]
```

Множественное присваивание позволяет писать более простой и лаконичный код. Именно так мы поступили и в случае с вычислением `x` и `y`.

Затем при помощи функции `save` сохраняем измененный список пикселей в файл изображения с именем riana2.jpg.

Важно!

В данном случае появляется новая картинка в том же месте, где находилась начальная. Начальное изображение осталось без изменений, а новое получено из начального изменением значений цветовых компонентов для каждого пикселя.

Фильтры

Когда-то `Instagram` превратился из заурядной социальной сети в очень популярный феномен именно из-за удачной реализации встроенных фильтров. Фильтры можно было накладывать на фотографии, которые после этого обычно становились красивыми, похожими на профессиональные.

Фильтры очень широко применяются в киноиндустрии. Сравните цветовую гамму молодежных комедий или современных блокбастеров, например, с классическим «Шерлоком Холмсом».

Иначе говоря, фильтры невероятно востребованы — начиная от самых простых и заканчивая работами с привлечением искусственного интеллекта: например, в проекте Prisma.

Фильтры

Фильтр можно воспринимать как любое преобразование заданного изображения.

Чтобы добиться лучшего эффекта, их можно накладывать последовательно.

В библиотеке PIL реализовано много встроенных фильтров и инструментов (вырезание, изменение размеров и т. д.). Фактически это такой программируемый мини-Photoshop, но мы попытаемся поработать с фильтрами самостоятельно, чтобы поучиться восприятию цветовой палитры и алгоритмизации.

Для начала попробуем превратить изображение в черно-белое.

Черно-белое изображение

Черно-белое изображение содержит только информацию о яркости, но не о цветах. У таких изображений все три компоненты имеют одинаковое значение, поэтому мы можем просто «размазать» суммарную яркость пикселя поровну по трем компонентам.

```
for i in range(x):
    for j in range(y):
        r, g, b = pixels[i, j]
        bw = (r + g + b) // 3
        pixels[i, j] = bw, bw, bw
```

Можно сказать, что мы слили содержимое контейнеров R, G, B в одну емкость, а затем разлили обратно, но уже поровну в каждый контейнер. Суммарная яркость пикселя осталась прежней, но информация о цвете не сохранилась. Фотография же стала более «задумчивой».

Попробуем поменять местами зеленый и синий каналы:

```
for i in range(x):
    for j in range(y):
        r, g, b = pixels[i, j]
        pixels[i, j] = r, b, g
```

Негатив

Давайте подумаем над тем, как получить негатив. Если в позитиве белое изображение (255), в негативе должно быть черное (0) и наоборот.
То есть для значения x негативом будет 255 — x.

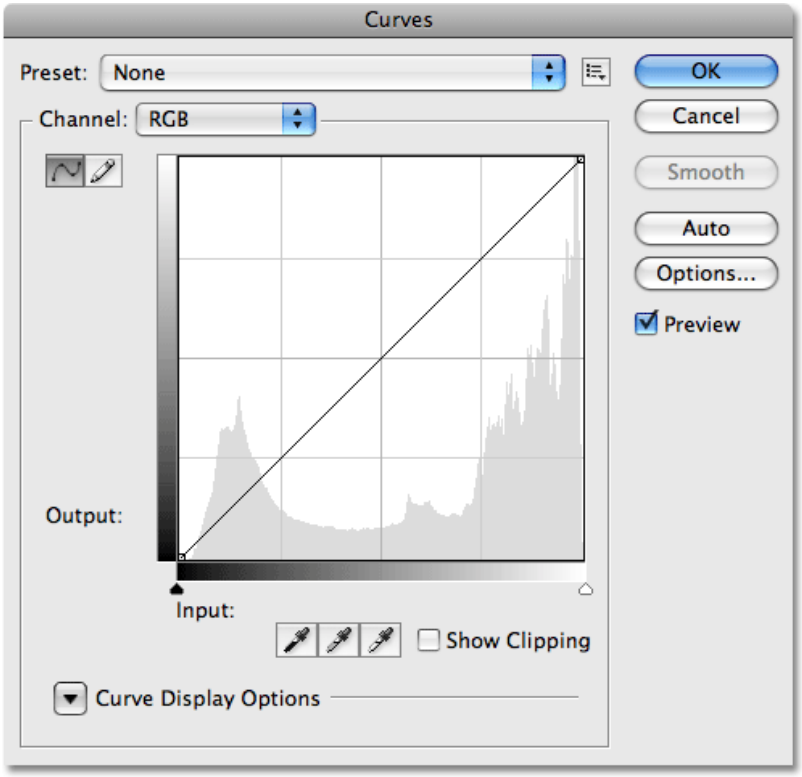
```
for i in range(x):

    for j in range(y):

        r, g, b = pixels[i, j]

        pixels[i, j] = 255 - r, 255 - g, 255 - b
```

Как видим, в негативе можно рассмотреть некоторые детали, которые не видны в позитиве.

Во многих редакторах, включая `Photoshop`, есть инструмент «Кривые» (`Curves`).

Один из самых частых случаев применения кривых — это фотография с очень светлыми и темными участками: например, солнечный свет и тень на пейзажном фото. При неправильно выставленных параметрах съемки фотоаппарат ориентируется по яркости самого яркого участка. Тень при этом становится очень темной, лишенной деталей. Можно поработать с кривыми, обрабатывая именно такие изображения.

Этот инструмент позволяет задать функцию, меняющую яркость всего пикселя или отдельной компоненты в зависимости от исходной яркости. Изначально эта функция представляет собой прямую y = x.



В Python можно написать функцию, которая работает как инструмент Curves. Например, мы можем высветлить темные участки в изображении, не трогая светлые. Это очень частая операция: например, когда на снимке светлое небо и очень темное здание, потому что фотоаппарат подстроился под яркость неба.

Высветление

«Высветлить» означает увеличить значения всех цветовых компонентов на какой-то коэффициент. Важно помнить, что эти значения не могут быть больше 255.

```
def curve(pixel):
    r, g, b = pixel
    brightness = r + g + b if r + g + b > 0 else 1
    if brightness < 60:
        k = 60 / brightness
        return min(255, int(r * k ** 2)), \
               min(255, int(g * k ** 2)),\
               min(255, int(b * k ** 2))
    else:
        return r, g, b
```

```
for i in range(x):
    for j in range(y):
        pixels[i, j] = curve(pixels[i, j])
```

Результат:

Готовые функции

Как мы уже говорили раньше, в PIL есть большое число встроенных инструментов для изменения изображений. Продолжим эксперименты над нашим изображением совы.

Например, мы можем изменить размер изображения с помощью функции `resize`, в которую кортежем передается новый размер изображения. Обратите внимание: все подобные функции не изменяют исходное изображение, а возвращают его измененную копию.

```
from PIL import Image


im = Image.open("2.jpg")
```

```
im2 = im.resize((100, 100))

im2.save('6.jpg')
```

С помощью функции `crop` вырезать прямоугольный кусочек из изображения. В функцию передаются координаты верхнего левого и правого нижнего угла вырезаемого прямоугольника одним кортежем.

```
from PIL import Image


im = Image.open("2.jpg")

im2 = im.crop((200, 200, 500, 500))

im2.save('7.jpg')
```

С помощью функции `paste` вырезанный кусочек можно вставить во вновь созданный файл. Если не передавать координаты вставки, то он вставится в левый верхний угол, а если передать, то в указанное место:

```
from PIL import Image


im = Image.open("riana.jpg")

x, y = im.size

im1 = im.crop((0, 0, x // 2, y // 2))
```

```
im2 = Image.new("RGB", (x, y))

im2.paste(im1)

im2.paste(im1, (x // 2, y // 2))

im2.save("part.png")
```

Функция `quantize` используется для сокращения цветов в палитре изображения и используется для создания миниатюр для предпросмотра. Принимает на вход число меньшее 256 — количество цветов. Обратите внимание: эта функция также преобразовывает изображение в формат bmp.

```
from PIL import Image


im = Image.open("2.jpg")

im2 = im.quantize(16)

im2.save('8.bmp')
```

Вращать и отражать изображения можно, манипулируя пикселями, например, меняя местами пиксели справа от вертикальной оси симметрии на пиксели слева и наоборот. Однако PIL содержит уже готовые реализации данных алгоритмов. Повороты и отражения изображения можно выполнить с помощью функции `transpose`, в которую передается тип преобразования. Это может быть отражение слева направо, или сверху вниз, или повороты на 90, 180 или 270 градусов.

```
from PIL import Image



im = Image.open("2.jpg")

im2 =
im.transpose(Image.FLIP_LEFT_RIGHT).transpose(Image.ROTAT
E_90)

# Image.FLIP_LEFT_RIGHT,

# Image.FLIP_TOP_BOTTOM,

# Image.ROTATE_90,

# Image.ROTATE_180,

# Image.ROTATE_270
```

```
im2.save('9.jpg')
```

Благодаря тому, что преобразование возвращает измененное изображение, можно создавать цепочки преобразований.

Библиотека PIL содержит в себе много любопытного: например, в модуле `ImageOps` есть встроенные реализации превращения изображения в негатив (функция `invert`) и в черно-белое изображение (функция `grayscale`), а в модуле `ImageFilter` находятся встроенные интересные фильтры изображений, один из которых мы рассмотрим. Этот фильтр называется размытие Гаусса и принимает на вход радиус размытия.

```
from PIL import Image, ImageFilter



im = Image.open("2.jpg")

im2 = im.filter(ImageFilter.GaussianBlur(radius=5))
```

```
im2.save('10.jpg')
```

Все фильтры данного модуля можно передавать
в функцию `filter` изображения. А еще можно делать цепочки
преобразований для получения сложных эффектов.

Обязательно загляните в документацию к библиотеке PIL, мы уверены,
что вы найдете там много интересных функций, которые не поместились
в наш рассказ об этом модуле.