

## Практическое задание.

### Теоретическая часть.

Библиотека PIL (Python Imaging Library) - это библиотека для работы с изображениями в Python. Она предоставляет множество функций для открытия, обработки и сохранения изображений различных форматов. В настоящее время основной развивающийся форк PIL - это Pillow, который представляет собой продолжение разработки PIL с улучшенной поддержкой и обновленным API.

Разберём модули PIL: Image и ImageDraw.

**Image** - это основной класс для работы с изображениями в Pillow. Он предоставляет методы для открытия, создания, сохранения и манипуляций с изображениями.

Метод	Описание	Пример
Image.open()	Этот метод открывает изображение из файла.	<code>img = Image.open("example.jpg")</code>
Image.save()	Сохраняет изображение в файл.	<code>img.save("output.jpg")</code>
Image.show()	Отображает изображение во внешнем просмотрщике.	<code>img.show()</code>
Image.convert()	Преобразует изображение в другой режим (например, в черно-белый)	<code>bw_img = img.convert("L")</code>
Image.resize()	Изменяет размер изображения.	<code>resized_img = img.resize((width, height))</code>
Image.rotate()	Поворачивает изображение на указанный угол.	<code>rotated_img = img.rotate(90)</code>
Image.crop()	Обрезает изображение до указанных размеров.	<code>cropped_img = img.crop((left, top, right, bottom))</code>

### Пример использования.

#### Пример 1.

В качестве примера откроем изображение, находящееся в C:\Users\users\Desktop и имеющее имя, и расширение image.jpg

```
from PIL import Image

image = Image.open("C:\\Users\\users\\Desktop\\image.jpg")
image.show()
```

- `from PIL import Image`: Эта строка импортирует класс `Image` из модуля `PIL` (или `Pillow`), который используется для работы с изображениями.
- `image = Image.open("C:\\Users\\users\\Desktop\\image.jpg")`: Эта строка открывает изображение с указанным путем ("`C:\\Users\\users\\Desktop\\image.jpg`") с помощью метода `open()` класса `Image` и присваивает его переменной `image`. Это открывает изображение для последующей обработки или отображения.
- `image.show()`: Эта строка отображает открытое изображение с помощью метода `show()`. Обычно это открывает стандартное приложение просмотра изображений операционной системы (например, `Windows Photo Viewer`) и показывает изображение пользователю.

Важно отметить, что для успешного выполнения кода необходимо, чтобы файл "`image.jpg`" находился по указанному пути, и у вас были права доступа для его чтения.

## Пример 2.

Сделаем картинку меньше, передадим её значения в другую переменную и сохраним.

```
from PIL import Image

image = Image.open("input_image.jpg")

smaller_image = image.resize((image.width // 2, image.height // 2))
smaller_image.save("smaller_image.jpg")
```

- `image = Image.open("input_image.jpg")`: Эта строка открывает изображение с именем "`input_image.jpg`". Метод `open()` класса `Image` используется для открытия изображения из файла и возвращает объект изображения. Этот объект присваивается переменной `image`, чтобы мы могли обращаться к нему позже.
- `smaller_image = image.resize((image.width // 2, image.height // 2))`: Эта строка создает новое изображение, уменьшая размер изображения `image` до половины его исходного размера по ширине и высоте. Метод `resize()` принимает кортеж (`new_width`, `new_height`) в качестве аргумента, где `new_width` и `new_height` - это новые размеры изображения. Мы используем оператор целочисленного деления `//` для получения целых чисел, так как размеры должны быть целыми числами.
- `smaller_image.save("smaller_image.jpg")`: Эта строка сохраняет уменьшенное изображение в файл "`smaller_image.jpg`". Метод `save()`

используется для сохранения изображения в файле. В данном случае, мы сохраняем уменьшенное изображение, которое мы получили после вызова метода `resize()`.

**ImageDraw** - это модуль, который предоставляет класс `ImageDraw.Draw`, который используется для рисования на изображениях.

Метод	Описание
<code>ImageDraw.Draw()</code>	Метод для создания объекта <code>ImageDraw.Draw</code> , связанного с конкретным изображением. Этот объект используется для выполнения операций рисования, таких как рисование линий, прямоугольников, окружностей, текста и т. д.
<code>ImageDraw.Draw.line()</code>	Метод для рисования линии на изображении.
<code>Image.show()</code>	Отображает изображение во внешнем просмотрщике.
<code>ImageDraw.Draw.rectangle()</code>	Метод для рисования прямоугольника на изображении.
<code>ImageDraw.Draw.text()</code>	Метод для добавления текста на изображение.
<code>ImageDraw.ellipse()</code>	Метод для рисования окружности на изображении.

### Пример рисования: квадрат и текст.

```
from PIL import Image, ImageDraw, ImageFont

# Создаем новое изображение
image = Image.new("RGB", (200, 200), "white")
draw = ImageDraw.Draw(image)

# Рисуем синий прямоугольник с красным контуром
draw.rectangle((50, 50, 150, 150), outline="red", fill="blue")

# Загружаем шрифт
font = ImageFont.truetype("arial.ttf", 20)

# Добавляем текст на изображение
draw.text((50, 50), "Hello, PIL!", fill="black", font=font)

# Отображаем изображение
image.show()
```

### Пример рисования: круг и линия.

```
from PIL import Image, ImageDraw

# Создаем новое изображение размером 200x200 пикселей, белого цвета
image = Image.new("RGB", (200, 200), "white")

# Создаем объект ImageDraw для рисования на изображении
draw = ImageDraw.Draw(image)

# Рисуем эллипс с контуром и заливкой
draw.ellipse((50, 50, 150, 150), outline="black", fill="blue")
```

```
# Рисуем красную линию
draw.line((20, 30, 180, 170), fill="red", width=2)

# Отображаем изображение
image.show()
```

## Практическая часть.

### Задание 1.

В папке с практическим заданием находится изображение screenshot.jpg переделайте его в ч/б и сохраните в свою папку с названием screenshot\_bw.jpg.

### Задание 2.

Снимок с камеры наблюдения screen\_camera.png оказался перевёрнут в процессе сохранения. Используйте метод библиотеки и переверните его. Далее, сохраните к себе в папку с тем же именем.

### Задание 3.

Откройте изображение figures.png и обрежьте её так, чтобы в итоге остался только квадрат с надписью. Красным выделено, что должно остаться после обрезки.



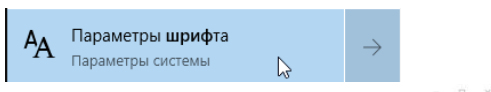
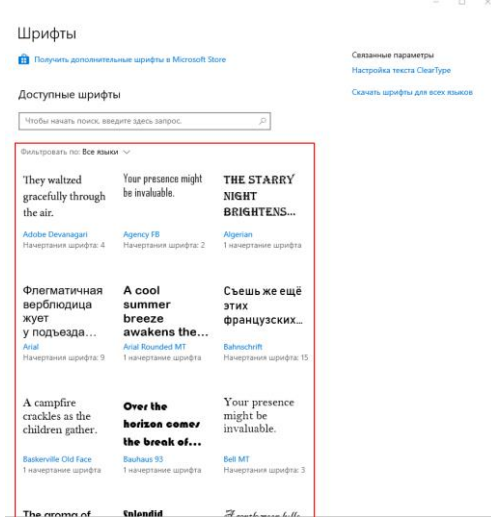
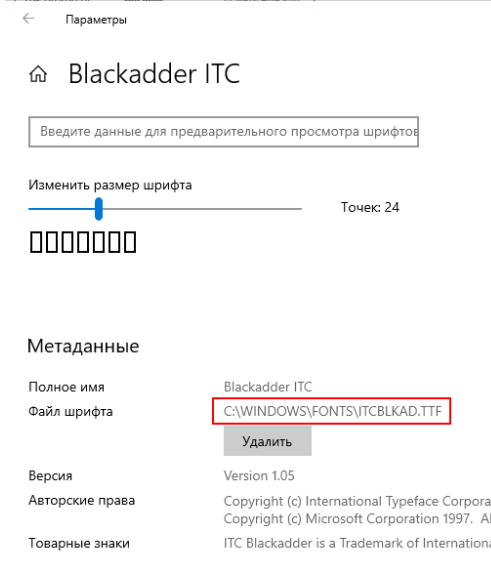
Сохраните результат в вашу папку и назовите cube.png.

## Задание 4.

Создайте изображение 200x200 раскрашенное в любой цвет и надпись: «Ваше\_имя был здесь» Используйте любой шрифт. Сохраните результат в папку с названием «Ваше\_имя был здесь.png».



Для начала выберите шрифт. Перейдите в настройки:

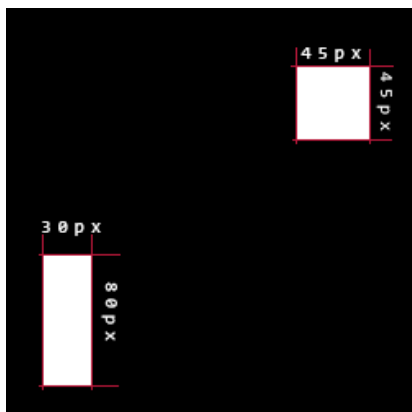
1. 
2.   

- 3.

После выбора, пропишите полный путь в переменной и свойстве выбора:

```
font = ImageFont.truetype("C:\\WINDOWS\\FONTS\\ITCBLKAD.ttf", size)
```

## Задание 5.

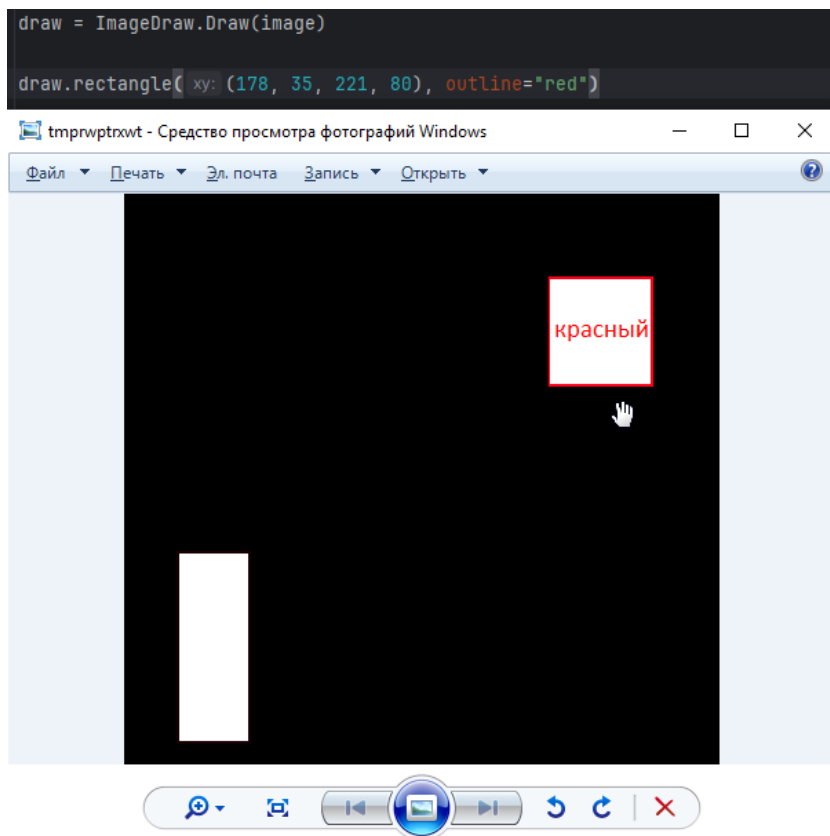
В изображении pixels.png присутствуют лишние 2 отверстия.



Необходимо загрузить готовую картинку и нарисовать на ней. Закройте верхнее отверстие закрашенным красным прямоугольником, а нижнее синим. Учитывайте размер исходного изображения.

**Важно!** Прямоугольники не должны перекрывать слишком много пространства вокруг отверстия. Используйте outline, чтобы отследить событие.

Рассмотрите пример ниже, координаты для верхнего прямоугольника уже даны.



Результат сохраните к себе в папку с названием pixels\_ready.png.

## Задание 6.

Есть некоторая карта пикселей:

```
pixels = [[1, 1, 1, 1, 1, 1, 1, 1, 1],
           [1, 0, 1, 1, 1, 1, 1, 0, 1],
           [1, 1, 1, 0, 0, 0, 1, 1, 1],
           [1, 1, 0, 1, 0, 1, 0, 1, 1],
           [1, 1, 0, 0, 1, 0, 0, 1, 1],
           [1, 1, 0, 1, 0, 1, 0, 1, 1],
           [1, 1, 1, 0, 0, 0, 1, 1, 1],
           [1, 0, 1, 1, 1, 1, 1, 0, 1],
           [1, 1, 1, 1, 1, 1, 1, 1, 1]]
```

В ней 1 – это чёрный цвет, а 0 – белый. Необходимо создать изображение по этой карте.

### Построчный алгоритм выполнения.

**Импорт библиотек:** Вы импортируете два модуля из библиотеки PIL (Python Imaging Library): Image и ImageDraw. Эти модули предоставляют инструменты для работы с изображениями и рисования на них.

**Определение списка пикселей:** Вы определяете список pixels, который представляет собой изображение, представленное в виде матрицы битов. В этом списке каждый элемент - это строка из нулей (0) и единиц (1), где 1 обозначает черный пиксель, а 0 - белый пиксель.

**Создание нового изображения с белым фоном:** Вы используете метод Image.new(), чтобы создать новый объект изображения. Аргументы этого метода включают режим изображения ("RGB" в данном случае), размер изображения (ширина и высота) и цвет фона (в данном случае, "white").

**Получение объекта "рисование":** Вы получаете объект "рисование" изображения, который позволяет вам рисовать на созданном изображении с помощью метода ImageDraw.Draw().

**Определение размера пикселя:** Вы задаете размер пикселя в переменной, например, pixel\_size. В данном случае он равен 1 пикселю.

Далее необходимо построить цикл. Речь идёт о двумерном массиве, поэтому и цикл будет выстроен соответствующе. Рассмотрим структуру цикла с примерами переменных:

```
for y, row in enumerate(pixels):
```

Этот цикл `for` проходит по каждой строке в списке `pixels`. Переменная `y` получает индекс строки, а `row` - саму строку (список значений пикселей в этой строке).

```
for x, pixel in enumerate(row):
```

Внутренний цикл `for` перебирает каждый пиксель в текущей строке `row`. Переменная `x` получает индекс пикселя в строке, а `pixel` - значение этого пикселя (1 - черный, 0 - белый).

```
top_left = (x * pixel_size, y * pixel_size):
```

Вычисляются координаты верхнего левого угла прямоугольника, представляющего текущий пиксель. Каждая координата вычисляется как произведение индекса пикселя (по горизонтали `x` и по вертикали `y`) на размер пикселя (`pixel_size`).

```
bottom_right = ((x + 1) * pixel_size, (y + 1) * pixel_size):
```

Вычисляются координаты нижнего правого угла прямоугольника. Для этого добавляется размер пикселя к координатам верхнего левого угла.

```
color = "black" if pixel == 1 else "white":
```

Определяется цвет для текущего пикселя. Если значение пикселя равно 1, то цвет устанавливается как "black" (черный), иначе - "white" (белый).

```
draw.rectangle([top_left, bottom_right], fill=color):
```

Нарисуйте прямоугольник на изображении, представляющий текущий пиксель. Координаты верхнего левого и нижнего правого углов используются для определения местоположения и размера прямоугольника, а цвет определяется переменной `color`.

После того как цикл закончен, можно показывать изображение. Сохраните его в свою папку с названием "my pixel pict.png"