Задачи курса "Работа с файлами и изображениями на Python"

Задачи в виде ответов на вопросы

1.

img - открытое с помощью Image.open() Pillow-изображение. Каким образом можно узнать размеры изображения?

Выберите один или несколько ответов:

1. img.getsize
2. img.size
3. img.getsize()
4. img.size()

2.

Что выведет данный код?

```
from PIL import Image
import numpy as np
color = (0,0,0)
img = Image.new("RGB",(200,230),color)
img_arr = np.array(img)
count = 0
for row in img_arr:
    for pixel in row:
        if pixel[0] == 0 and pixel[1] == 0 and pixel[2] == 0:
            count+=1
print(count)
```

Выберите один ответ:

1. 600
2. 13
3. 100
4. 40000
5. 230
6. 200
7. 0
8. 46000

3.

Какие методы и поля есть у Pillow изображения? (PIL.Image.Image)

Выберите один или несколько ответов:

1. width()
2. show()
3. height
4. format
5. save()
6. addpixel()
7. getpixel()
8. removepixel()

4.

Каким образом можно создать изображение, с котором можно работать, используя библиотеку OpenCV?

Выберите один или несколько ответов:

1. numpy.full((300, 400, 1), 150, dtype=numpy.uint8)
2. cv2.new("path.jpg",image)
3. cv2.createImage("path.jpg",image)
4. numpy.full((300, 400, 2), [150,100], dtype=numpy.uint8)
5. numpy.full((20,200,3), [255,255,255], dtype = numpy.uint8)

5.

Что делает данный код?

```
import cv2
img = cv2.imread("img4.jpg")
for row in range(img.shape[0]):
    for col in range(img.shape[1]):
        pix = img[row,col]
        if pix[0] > pix[2]:
            img[row,col] = [0,0,0]
cv2.imwrite("new.jpg",img)
```

Выберите один ответ:

1. Заменяет все пиксели, где красная компонента преобладает на синей, на белый цвет.
2. Заменяет все пиксели, где синяя компонента преобладает над красной, на белый цвет.
3. Заменяет все пиксели, где красная компонента преобладает над синей, на черный цвет.
4. Заменяет все пиксели, где синяя компонента преобладает над красной, на черный цвет.
5. Данный код завершится с ошибкой

6.

Какими способами можно узнать высоту изображения, работая с библиотекой OpenCV?

Выберите один или несколько ответов:

1. img.height()
2. cv2.height(img)
3. img.height
4. img.shape[0]
5. img.shape[1]
6. cv2.shape(img)[1]
7. numpy.size(img,0)
8. img.getHeight()
9. cv2.shape(img)[0]

7.

С помощью каких методов можно считать содержимое файла?

Выберите один или несколько ответов:

1. read()
2. readlines()
3. getlines()
4. printline()
5. getline()
6. readall()
7. getall()
8. getcontent()
9. readline()
10. get()
11. readcontent()

8.

Сопоставьте названия режимов открытия файлов с их назначением.

Режимы : "a", "+", "b", "w", "r", "t", "rw", "rb".

Значения:

1. Для записи текста с перезаписыванием файла
2. Так открыть файл нельзя
3. Открытие файла в бинарном виде
4. Для чтения в бинарном виде
5. Для чтения и записи
6. Для чтения текста
7. Для добавления текста в конец файла
8. Открытие файла в текстовом виде

9.

Какие методы из перечисленных позволяют создать директорию?

Выберите один или несколько ответов:

1. os.rmdir()
2. os.create()
3. os.newdir()
4. os.mkdir()
5. os.makedir()

10.

Что выведется в консоль в операционной системе Windows?

print(os.path.join(".","..","..","dir1","dir2"))

11.

Соотнесите первые 2 байта файла изображения и его формат.

Форматы:

1. JPEG
2. PNG
3. BMP

Байты:

1. 42 4D
2. 42 30
3. FF E5

4. 4A 2B
5. 89 50
6. FF D8

12.

BMP–файле, открытом в бинарном виде, байты, отвечающие за высоту изображения, записаны следующим образом: BA 02 00 00. Какова высота изображения в пикселях?

Выберите один ответ:

1. 8378
2. 8363
3. 698
4. 3120693248
5. 700
6. 512
7. 47618

13.

Соотнесите байт-идентификатор и название маркера.

Названия маркеров:

1. EOI
2. SOF1
3. SOI
4. SOS
5. SOF0
6. DHT

Байты идентификаторы:

1. D9
2. FF
3. C1
4. C0
5. DA
6. F1
7. D8
8. C2
9. C4

14.

Выберите обязательные чанки (chunk), которые должны присутствовать в PNG-файле.

Выберите один или несколько ответов:

1. IDAT
2. PLTE
3. iTXt
4. zTXt
5. gAMA
6. hIST
7. IEND
8. tIME
9. IHDR
10. bkGD

15.

Напишите чему равен размер чанка без содержимого в PNG-файле в байтах.

16.

Первые 16 байт чанка в PNG-изображении выглядят следующим образом: «00 00 00 81 50 4C 54 45 00 00 00 FF FF FF EB EB EB». Какой это чанк (имя) и каков его размер в байтах?

Напишите ответ в следующем формате : «<Имя чанка>-<Количество байт>». Например, если это чанк iTXt и его размер 20 байт, то ответ будет выглядеть следующим образом: «iTXt-20».

17.

Как называется маркер в JPEG-изображении и каков его размер в байтах, если первые его 13 байт следующие: “FF E0 00 10 4A 46 49 46 00 01 01 00 00»?

Ответ запишите в формате «<Краткое название маркера>-<Количество байт>». Например, если это маркер SOF1 (Start of Frame 1) с размером 10 байт, то ответ будет выглядеть следующим образом: «SOF1-10».

18.

По какому смещению (в байтах) от начала PNG-файла находится ASCII-коды строки ‘PNG’?

19.

По какому смещению (в байтах) от начала BMP-файла находится размер массива пикселей в байтах (Image Size)?

Задачи с написанием кода

1.

(Pillow)

Необходимо написать функцию solve(), которая принимает на вход путь к изображению (img_path), открывает его с помощью средств Pillow и возвращает открытое изображение.

2.

(Pillow)

Необходимо написать функцию solve(), которая принимает на вход путь к изображению (img_path), открывает его с помощью средств Pillow и сохраняет изображение по пути "img.bmp"

3.

Изменение пикселя изображения (Pillow)

Необходимо написать функцию solve(), которая принимает на вход:

- путь к RGB изображению img_path
- координаты пикселя (x,y)
- цвет пикселя (color)

Функция должна открыть файл изображения с помощью средств Pillow и изменить в нем пиксель по координатам (x,y) на пиксель цвета color. Измененное изображение должно быть сохранено по пути 'img.bmp'.

4.

Необходимо написать функцию solve(), которая принимает на вход размеры (width, height) и цвет (color) изображения. Функция должна с помощью средств Pillow создать RGB-изображение заданных размеров и заданного цвета, и сохранить изображение по пути "img.png".

5.

(Pillow)

Необходимо написать функцию solve(), вход Numpy массив (array). Функция должна преобразовать массив в Pillow изображение (PIL.Image.Image) и сохранить это изображение по пути "img.jpg".

6.

Необходимо написать функцию solve(), которая принимает на вход путь к RGB изображению (img_path). Функция должна прочитать изображение с помощью средств

Pillow и вернуть информацию о ширине, высоте и формате изображение в формате списка : [width,height,format].

7.

(Pillow)

Необходимо реализовать функцию solve(), рисующую на картинке отрезок

Функция solve() принимает на вход:

- Путь к RGB изображению (img_path);
- координаты начала (x0, y0);
- координаты конца (x1, y1);
- цвет (color);
- толщину (width).

Функция должна сохранить измененное изображение по пути "img.png"

9.

(Pillow)

Необходимо реализовать функцию solve, которая рисует на изображении квадрат с определенными параметрами. Функция принимает на вход:

- Путь к RGB изображению img_path
- Координату левого верхнего угла (x1, y1)
- Сторону квадрата (side)
- Толщину линий (lineThickness)
- Цвет линий (lineColor)
- Квадрат может быть залит или нет (filled)
- Цвет которым он залит, если пользователем выбран залитый (если квадрат не залит, то значение цвета - None) (fillColor)

Функция должна нарисовать заданный квадрат на изображении и сохранить изображение по пути "img.png".

10.

Необходимо написать функцию solve(), которая рисует на изображении треугольник

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Координаты вершин (x0,y0,x1,y1,x2,y2)
- Толщину линий (thickness)
- Цвет линий (color)
- Цвет, которым залит (fill_color - если значение None, значит треугольник не залит)
- Функция должна сохранить обработанное изображение по пути "img.png".

Примечание: Используйте функцию polygon

11.

Необходимо реализовать функцию solve(), которая принимает на вход:

1) Путь к изображению - img_path

2) Цвет линий - color - (r,g,b) - tuple

3) Ширину линий - width

4) Координаты левого верхнего угла квадрата, в который вписан крест -x,y

5) Длину стороны квадрата - size

Функция должна нарисовать крест (диагонали) квадрата. Сам квадрат рисовать не нужно.

Функция должна сохранить измененное изображение по пути "img.png".

12.

Необходимо реализовать функцию solve(), которая принимает на вход:

1) Путь к изображению - img_path

2) Цвет линий - color - (r,g,b) - tuple

3) Ширину линий - width

4) Координаты левого верхнего угла прямоугольника (x1,y1)

5) Координаты нижнего правого угла прямоугольника (x2,y2)

6) Флаг того нужно ли рисовать сам прямоугольник (flag)

Функция должна нарисовать диагонали прямоугольника с заданными параметрами линии.

Если прямоугольник нужно нарисовать, то рисовать его без заливки, с шириной и цветом линии как у диагоналей.

Функция должна сохранить измененное изображение по пути "img.png".

13.

Необходимо реализовать функцию solve(), которая принимает на вход:

1) Путь к изображению - img_path

2) Цвет обводки - color - (r,g,b) - tuple

3) Цвет заливки - fill - (r,g,b) - tuple

4) Ширину линий - width

5) Координаты левого верхнего угла квадрата, в который вписана окружность - (x,y)

6) Длину стороны квадрата - size

7) Флаг того, нужно ли рисовать сам квадрат (flag)

Функция должна нарисовать окружность, которая вписана в квадрат с переданными параметрами.

Если flag = True, то необходимо нарисовать сам квадрат. Квадрат рисовать без заливки, с цветом и шириной линий как у окружности.

Функция должна сохранить измененное изображение по пути "img.png".

14.

Необходимо написать функцию solve(), которая рисует на изображении пентаграмму в окружности.

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- координаты центра окружность (x,y)
- радиус окружности
- Толщину линий и окружности (thickness)
- Цвет линий и окружности (color)

Функция должна сохранить обработанное изображение по пути "img.png".

Примечание:

Вершины пентаграммы высчитывать по формуле:

phi = (pi/5)*(2*i+3/2)

node_i = (int(x0+r*cos(phi)),int(y0+r*sin(phi)))

x0,y0 - координаты центра окружности, в который вписана пентаграмма

r - радиус окружности

i - номер вершины от 0 до 4

Линии нужно рисовать по отдельности, используя метод line.

15.

(Pillow)

Необходимо написать функцию solve(), которая реализует инверсию цвета в заданной области. Функция принимает на вход:

- Путь к RGB-изображение (img_path)
- Координаты левого верхнего угла области (x1, y1)
- Координаты правого нижнего угла области (x2, y2)

Функция должна сохранить измененное изображение по пути "img.png"

16.

(Pillow)

Необходимо написать функцию solve(), которая в заданной прямоугольной области (через левую верхнюю и правую нижнюю точки) меняет компоненты цвета местами. В цвете точки rgb, r меняется на g, g меняется на b, b меняется на r (то есть происходит смещение влево). Функция принимает на вход:

- Путь к исходному RGB изображению (img_path)
- Координаты левой верхней точки области (x1, y1)
- Координаты правой нижней точки области (x2, y2)

Функция должна сохранить измененное изображение по пути "img.png"

17.

(Pillow)

Необходимо написать функцию solve(), которая заменяет переданный цвет old_color, на другой цвет new_color.

Функция принимает на вход:

- Путь к RGB-изображению (img_path)
- Цвет, который требуется заменить (old_color)
- Цвет на который требуется заменить (new_color)

Функция должна в изображении преобразовать все пиксели цвета old_color в пиксели с цветом new_color. Функция должна сохранить измененное изображение по пути "img.png".

18.

(pillow)

Необходимо реализовать функцию solve(), которая преобразовывает изображение в черно-белый формат.

Функция принимает на вход:

- Путь к RGB-изображению (img)

- Функция должна преобразовать изображение в черно-белый формат и сохранить измененное изображение по пути "img.png".

Для преобразования пикселей изображения в черно-белый формат нужно использовать формулу: Y = int(0.36*R* + *0.53*G +0.11*B). Пиксель (R,G,B) заменяется на (Y, Y, Y).

19.

(Pillow)

Необходимо написать функцию solve(), которая закрашивает пиксели, которые расположены по определенным координатам, в переданный цвет.

Функция принимает на вход:

- Путь к файлу с RGB-изображением (img_path)
- Список с координатами пикселей (pixels)
- Цвет в которых их нужны закрасить (color)
- Координата пикселя - список из 2-х элементов - [x,y].

Функция должна заменить каждый переданный пиксель из pixels на цвет color и сохранить изображение по пути "img.png".

20.

(Pillow)

Необходимо реализовать функцию solve, которая заменяет цвет каждого пикселя в области на средний цвет пикселей вокруг (не считая сам этот писксель).

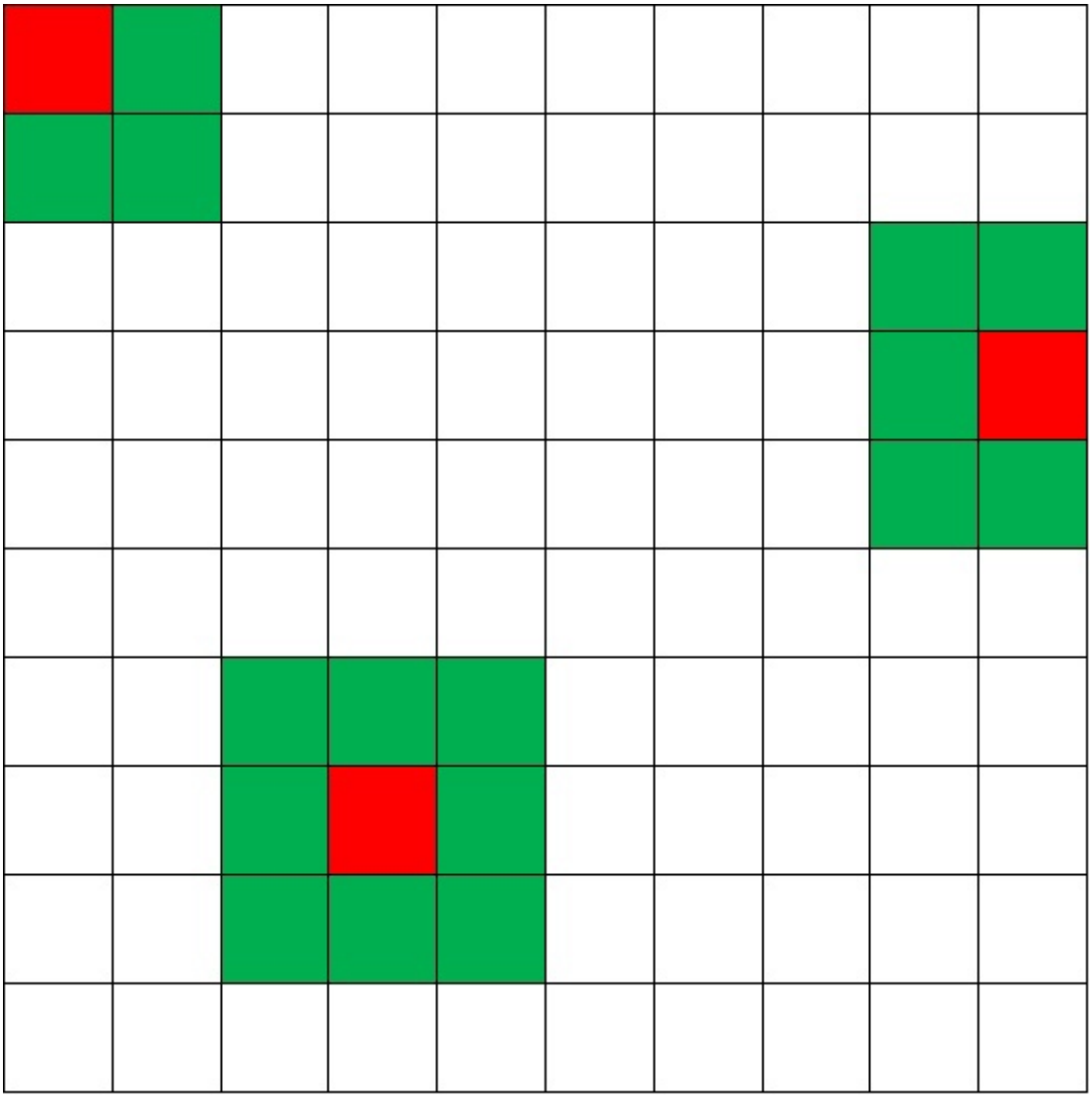
Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Координаты левого верхнего угла области (x0,y0)
- Координаты правого верхнего угла области (x1,y1)

Функция должна заменить цвета каждого пикселя в этой области на средний цвет пикселей вокруг.

Пиксели вокруг :

- 8 самых близких пикселей, если пиксель находится в центре изображения
- 5 самых близких пикселей, если пиксель находится у стенки
- 3 самых близких пикселя, если пиксель находится в угле



Средний цвет - берется целая часть от среднего каждой компоненты из rgb. (int(sum(r)/count),int(sum(g)/count),int(sum(b)/count))

Функция должна сохранить изображение по пути "img.png"

21.

(Pillow)

Необходимо написать функцию solve(), которая заменяет наиболее часто встречаемый цвет на переданный.

Функция solve() принимает на вход:

- Путь к RGB-Изображение (img_path)
- Цвет (color - представляет собой список из трех целых чисел)

Функция должна найти в изображении самый частый цвет и заменить его на переданный, затем сохранить измененное изображение по пути "img.png".

Подсказка: используйте функцию getcolors().

22.

Необходимо реализовать функцию solve, которая делит изображение на "полосы" и инвертирует цвет нечетных полос (счёт с нуля).

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Ширину полос в пикселах (N)
- Признак того, вертикальные или горизонтальные полосы(vertical - если True, то вертикальные)

Функция должна разделить изображение на вертикальные или горизонтальные полосы шириной N пикселей. И инвертировать цвет в нечетных полосах (счет с нуля). Последняя полоса может быть меньшей ширины, чем N.

Функция должна сохранить измененное изображение по пути "img.png".

23.

(pillow)

Необходимо реализовать функцию solve(), которая принимает путь к RGB изображению img_path и сортирует пиксели в нем по возрастанию суммы компонент (r+g+b). Функция должна сохранить изображение с отсортированными пикселями по пути "img.png"

24.

(Pillow)

Необходимо реализовать функцию solve, которая изменяет цвет в палитре цветов.

Функция solve() принимает на вход:

- Путь к изображению, которое использует палитру (img_path)
- Индекс цвета в палитре цветов - index
- Цвет на который нужно изменить - color - (list из 3х элементов [r,g,b])

Функция должна изменить цвет в палитре по индексу index на color и сохранить изображение по пути "img.png"

25.

(Pillow)

Необходимо реализовать функцию solve, которая переворачивает палитру цветов. То есть цвет, который имел наибольший индекс становится с индексом = 0 и так далее.

Функция solve() принимает на вход:

- Путь к изображению, которое использует палитру (img_path)
- Функция должна сохранить изображение по пути "img.png"

пример:

изначальная палитра - [0,0,10,255,100,50,40,30,20]

перевернутая палитра - [40,30,20,100,50,40,0,0,10]

26.

(Pillow)

Необходимо реализовать функцию solve, которая инвертирует первые n цветов в палитре.

Функция solve() принимает на вход:

- Путь к изображению, которое использует палитру (img_path)
- Число n - количество цветов, которые нужно инвертировать (с начала палитры)

Функция должна сохранить изображение по пути "img.png"

27.

(pillow)

Необходимо реализовать функцию solve(), которая копирует область одного изображения и вставляет эту область в другое изображение.

Функция solve() принимает на вход:

- Путь к RGB изображению 1 (img_from_path)
- Путь к RGB изображению 2 (img_to_path)
- Координаты левого верхнего угла области (x1,y1)
- Координаты нижнего правого угла области (x2,y2)

Функция должна скопировать область изображения img_from, где (x1,y1) - координаты верхнего левого угла области копирования, (x2,y2) - координаты нижнего правого угла области копирования.

Функция должна вставить скопированную область в изображение img_to, по координате (0,0).

Функция должна сохранить измененное изображение по пути "img.png".

28.

(Pillow)

Необходимо реализовать функцию solve(), которая вставляет одно изображение в другое.

Функция solve() принимает на вход:

- Путь к RGB изображению 1 (img_from_path)
- Путь к RGB изображению 2 (img_to_path)
- Координаты левого верхнего угла области вставки(x,y)

Функция должна вставить изображение 1 в изображение 2, начиная с координат (x,y).

Функция должна сохранить измененное изображение 2 по пути "img.png".

29.

(pillow)

Необходимо написать функцию solve(), которая реализует поворот квадратной части изображения на 90/180/270 градусов против часовой стрелки. Функция принимает на вход:

Путь к исходному RGB изображению (img_path) Координаты левого верхнего угла области (x, y) Ширину квадратной области (width) Угол поворота (angle - 90 или 180 или 270) Функция должна сохранить изображение по пути "img.png"

30.

(Pillow)

Необходимо реализовать функцию solve, которая меняет местами два одинаковых по размеру участка изображения. Участки не пересекаются.

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Координаты левого верхнего угла первого участка(x0,y0)
- Координаты левого верхнего угла второго участка(x1,y1)
- Размеры области (width,height)

Функция должна поменять участки изображения и сохранить изображение по пути "img.png".

31.

Необходимо реализовать функцию solve, которая меняет местами два квадратных, одинаковых по размеру, участка изображений и поворачивает эти участки на 90 градусов по часовой стрелке, а затем поворачивает изображение на 90 градусов по часовой стрелке.

Функция solve() принимает на вход:

- Путь к квадратному RGB-изображению (img_path)
- Координаты левого верхнего угла первого квадратного участка(x0,y0)
- Координаты левого верхнего угла второго квадратного участка(x1,y1)
- Длину стороны квадратных участков (width)

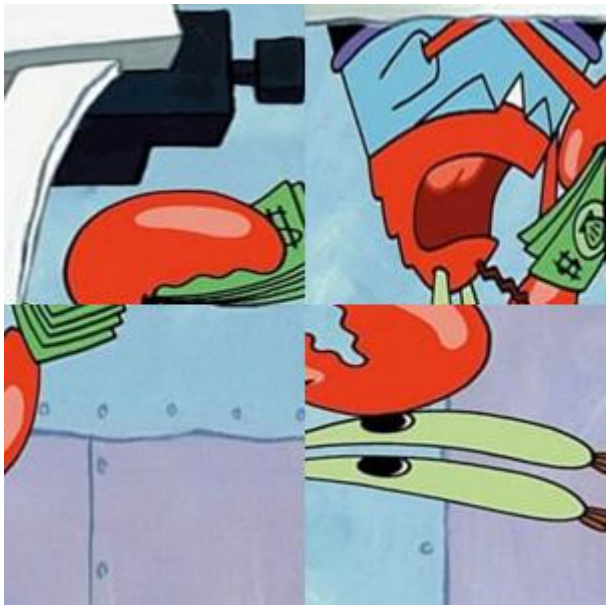
Функция должна сначала поменять местами переданные участки изображений [данные участки не пересекаются]. Затем повернуть каждый участок на 90 градусов по часовой стрелке. Затем повернуть всё изображение на 90 градусов по часовой стрелке.

Функция должна сохранить обработанное изображение по пути "img.png".

Пример входной картинки (300x300) и переданных параметров:

x0 = 0 ; y0 = 0; x1 = 150; y1 = 150; width = 150;

Изображение после первого этапа (замена участков и поворот этих участков на 90 градусов):

Итоговое изображение: 

32.

(Pillow)

Необходимо написать функцию solve().

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Строку (text)
- Цвет (color - str или tuple)
- Размер шрифта (size)
- Координату начала текста (x,y)

Функция должна написать текст text на изображении с заданными параметрами (цвет, размер, координата) и сохранить изображение по пути (img.png).

Файл шрифта находится по пути "arial.ttf".

33.

(Pillow)

Необходимо написать функцию solve().

Функция solve() принимает на вход:

- Размер изображения (width,height)
- Первый цвет изображения (img_color1 - tuple)
- Второй цвет изображения (img_color2 - tuple)
- Строку (text)
- Цвет текста (text_color - str или tuple)
- Размер шрифта (size)
- Координату начала текста (x,y)

Функция должна создать RGB-изображение заданных размеров (width,height), цвет строк изображения должен чередоваться: четные строки (0,2,4,...,2n) должны быть закрашены в цвет img_color1, нечетные строки (1,3,5,...,2n+1) - должны быть закрашены в цвет img_color2.

Затем на созданном изображении должен быть написан текст (text) с заданными параметрами (цвет, размер, координата). Изображение должно быть сохранено по пути "img.png".

Файл шрифта находится по пути "arial.ttf".

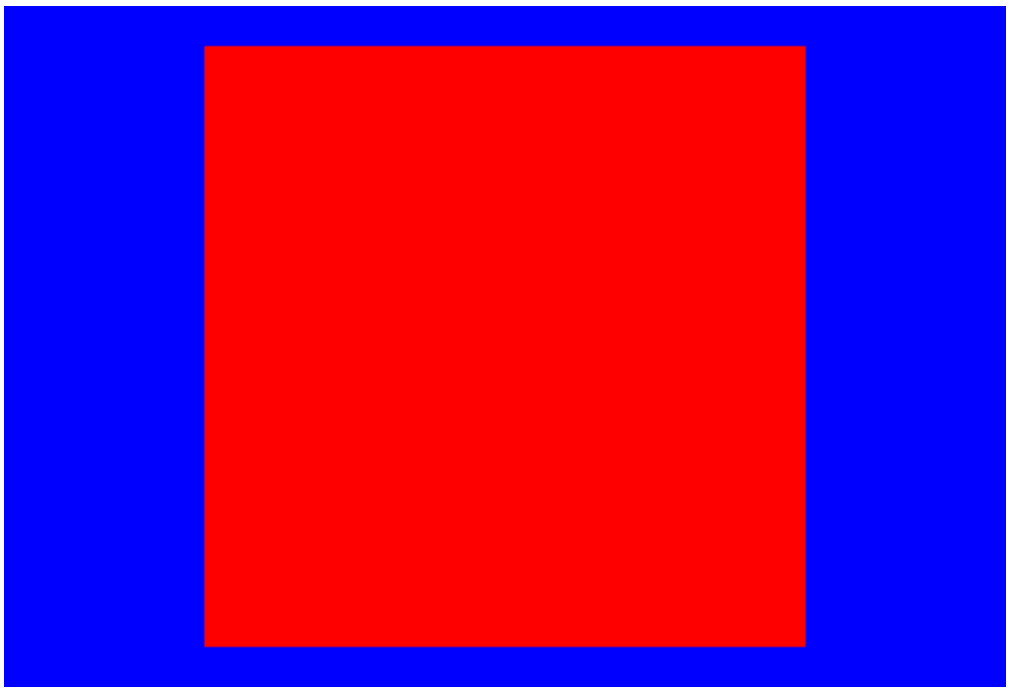
34.

Необходимо написать функцию solve(), которая принимает на вход:

- Путь к RGB-изображению (img_path)
- Цвет (color - tuple)
- Прирост высоты и ширины с каждой стороны (width_gain, height_gain)

Функция должна расширить изображение, "нарастив" вокруг него фон заданного цвета color. Новое изображение должно быть сохранено по пути "img.png"

Пример: Изначальное изображение: 
После работы функции с параметрами: color = (0,0,255),width_gain = 100, height_gain =

20: 

35.

(pillow)

Необходимо реализовать функцию solve(), которая принимает на вход:

- Путь к RGB изображению (img_path)
- Координаты левого верхнего угла области - (x0,y0)
- Координаты правого нижнего угла области - (x1,y1)
- Ось отражения (vertical - если True, то вертикальная. Иначе - горизонтальная)

Функция должна отразить изображение в переданном участке по одной из осей. Измененное изображение нужно сохранить по пути "img.png".

36.

(Pillow)

Необходимо реализовать функцию solve, которая делит квадратное изображение на 9 равных частей (сторона изображения делится на 3), и по правилам, записанным в словаре, меняет их местами.

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Словарь с описанием того, какие части на какие менять (rules)

Пример словаря rules:

{0:1,1:2,2:4,3:4,4:5,5:3,6:8,7:8,8:8}

Элементы нумеруются слева-направо, сверху-вниз.

В данном случае нулевой элемент заменяется на первый, первый на второй, второй на четвертый, третий на четвертый и так далее.

Функция должна сохранить измененное изображение по пути "img.png"

Пример входной картинки и словаря: 
{0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8}

Результат: 

37.

(Pillow)

Необходимо написать функцию solve().

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Количество изображений по "оси" Y (N - натуральное)
- Количество изображений по "оси" X (M - натуральное)

Функция должна создать коллаж изображений (это же изображение, повторяющееся NxM раз. (N раз по высоте, M раз по ширине) и сохранить его по пути "img.png".

38.

(OpenCV)

Необходимо написать функцию solve(), которая принимает на вход путь к изображению (img_path), открывает его с помощью средств OpenCV и возвращает открытое изображение.

39.

(OpenCV)

Необходимо написать функцию solve(), которая принимает на вход путь к изображению (img_path), открывает его с помощью средств OpenCV и сохраняет изображение по пути "img.bmp".

40.

(OpenCV)

Необходимо написать функцию solve(), которая принимает на вход:

- путь к RGB изображению img_path
- координаты пикселя (x,y)
- цвет пикселя (color - в виде списка list)

Функция должна открыть файл изображения с помощью средств OpenCV и изменить в нем пиксель по координатам (x,y) на пиксель цвета color. Измененное изображение должно быть сохранено по пути 'img.bmp'.

41.

(OpenCV)

Необходимо написать функцию solve(), которая принимает на вход путь к изображению (img_path) и число (k), открывает изображение с помощью средств OpenCV и возвращает представление k-ой строки пикселей в виде numpy-массива.

42.

(OpenCV)

Необходимо написать функцию solve(), которая принимает на вход размеры (width, height) и цвет (color - list) изображения. Функция должна с помощью средств OpenCV и numpy создать RGB-изображение заданных размеров и заданного цвета, и сохранить изображение по пути "img.png".

43.

(OpenCV)

Необходимо реализовать функцию solve(), рисующую на картинке 2 отрезка.

Функция solve() принимает на вход:

- Путь к RGB изображению (img_path);
- координаты начала (x0, y0);
- координаты конца (x1, y1);
- цвет (color - list);
- толщину (width).

Функция должна нарисовать отрезок с переданными координатами. А затем параллельный ему отрезок ниже на 30 пикселей с инвертированным цветом.

Функция должна сохранить измененное изображение по пути "img.png"

44.

(OpenCV)

Необходимо реализовать функцию solve, которая рисует на изображении квадрат с определенными параметрами. Функция принимает на вход:

- Путь к RGB изображению img_path
- Координату левого верхнего угла (x1, y1)
- Сторону квадрата (side)
- Толщину линий (lineThickness)
- Цвет линий (lineColor - list)
- Квадрат может быть залит или нет (filled)
- Цвет которым он залит, если пользователем выбран залитый (fillColor - list)

Если квадрат залит, то сначала нужно нарисовать залитую часть квадрата, а затем его "бортик".

Функция должна нарисовать заданный квадрат на изображении и сохранить изображение по пути "img.png".

45.

(OpenCV)

Необходимо написать функцию solve(), которая рисует на изображении треугольник

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Координаты вершин (x0,y0,x1,y1,x2,y2)
- Толщину линий (thickness)
- Цвет линий (color)
- Цвет, которым залит (fill_color - если значение None, значит треугольник не залит)

Для залитого треугольника сначала надо отобразить залитый прямоугольник, а затем линии его обводки.

Подсказка: используйте функции polylines и fillPoly библиотеки OpenCV

Функция должна сохранить обработанное изображение по пути "img.png".

46.

(OpenCV)

Необходимо реализовать функцию solve(), которая принимает на вход:

1) Путь к изображению - img_path

2) Цвет линий - color - [b,g,r] - list

3) Ширину линий - width

4) Координаты левого верхнего угла квадрата, в который вписан крест -x,y

5) Длину стороны квадрата - size

Функция должна нарисовать крест (диагонали) квадрата и сам квадрат (та же ширина линий и цвет).

Функция должна сохранить измененное изображение по пути "img.png".

47.

(OpenCV)

Необходимо реализовать функцию solve(), которая принимает на вход:

1) Путь к изображению -(img_path)

2) Цвет линий - (color - [b,g,r] - list)

3) Ширину линий - (width)

4) Координаты левого верхнего угла прямоугольника (x1,y1)

5) Координаты нижнего правого угла прямоугольника (x2,y2)

6) Флаг того нужно ли рисовать сам прямоугольник (flag)

Функция должна нарисовать диагонали прямоугольника с заданными параметрами линии.

Если прямоугольник нужно нарисовать, то рисовать его без заливки, с шириной и цветом линии как у диагоналей.

Функция должна сохранить измененное изображение по пути "img.png".

48.

(OpenCV)

Необходимо реализовать функцию solve(), которая принимает на вход:

1) Путь к изображению - (img_path)

2) Цвет - color - ([b,g,r] - list)

3) Ширину линий - (width)

4) Координаты левого верхнего угла квадрата, в который вписана окружность - (x,y)

5) Длину стороны квадрата - (size)

6) Флаг того, нужно ли рисовать сам квадрат - (flag)

Функция должна нарисовать окружность, которая вписана в квадрат с переданными параметрами.

Если flag = True, то необходимо нарисовать сам квадрат. Квадрат рисовать без заливки, с цветом и шириной линий как у окружности.

Функция должна сохранить измененное изображение по пути "img.png".

49.

(OpenCV)

Необходимо реализовать функцию solve, которая рисует на изображении многоугольник. Функция принимает на вход:

- Путь к RGB изображению img_path
- Координаты точек вершин в виде списка coords (x_1, y_1, x_2, y_2, ..., x_n,y_n)
- Цвет линий - color [b,g,r] - list
- Толщину линий - width
- Цвет заливки - fill (b,g,r) - list (Если None, то без заливки)

Функция должна нарисовать многоугольник с заданными параметрами на изображении, а затем сохранить измененное изображение по пути "img.png".

50.

(OpenCV)

Необходимо написать функцию solve(), которая рисует на изображении пентаграмму в окружности.

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)

- Координаты левого верхнего угла квадрата, в который вписана окружность (x,y)
- Длину стороны квадрата (size)
- Толщину линий и окружности (thickness)
- Цвет линий и окружности (color)

Функция должна сохранить обработанное изображение по пути "img.png".

Примечание: Вершины пентаграммы высчитывать по формуле:

phi = (pi/5)*(2*i+3/2)

node_i = (int(x0+r*cos(phi)),int(y0+r*sin(phi)))

x0,y0 - координаты центра окружности, в который вписана пентаграмма

r - радиус окружности

i - номер вершины от 0 до 4

Линии нужно рисовать по отдельности, используя метод line.

51.

(OpenCV)

Необходимо написать функцию solve(), которая реализует инверсию красной и зеленой компоненты в заданной области. Функция принимает на вход:

- Путь к RGB-изображение (img_path)
- Координаты левого верхнего угла области (x1, y1)
- Координаты правого нижнего угла области (x2, y2)

Функция должна сохранить измененное изображение по пути "img.png"

52.

(OpenCV)

Необходимо написать функцию solve(), которая в заданной прямоугольной области (через левую верхнюю и правую нижнюю точки) меняет компоненты цвета местами. В цвете точки rgb, r меняется на b, g меняется на r, b меняется на g. Функция принимает на вход:

- Путь к исходному RGB изображению (img_path)
- Координаты левой верхней точки области (x1, y1)
- Координаты правой нижней точки области (x2, y2)

Функция должна сохранить измененное изображение по пути "img.png"

53.

(OpenCV)

Необходимо написать функцию solve(), которая заменяет все цвета в изображении, хоть одна компонента которого подходит под условие, на новый цвет.

Условие: Любая компонента пикселя (k) - cond1<k<cond2

Функция принимает на вход:

- Путь к RGB-изображению (img_path)
- Число, больше которого должна быть компонента для выполнения условия (cond1)
- Число, меньше которого должна быть компонента для выполнения условия (cond2)
- Цвет на который требуется заменить (new_color)

54.

(OpenCV)

Необходимо реализовать функцию solve(), которая преобразовывает изображение в черно-белый формат.

Функция принимает на вход:

- Путь к RGB-изображению (img)

Функция должна преобразовать изображение в черно-белый формат и сохранить измененное изображение по пути "img.png".

Для преобразования пикселей изображения в черно-белый формат нужно использовать формулу: Y = int(0.3*R* + *0.45*G +0.25*B). Пиксель (R,G,B) заменяется на (Y, Y, Y).

55.

(OpenCV)

Необходимо написать функцию solve(), которая закрашивает пиксели, которые расположены в переданных строках изображения.

Функция принимает на вход:

- Путь к файлу с RGB-изображением (img_path)
- Список с номерами строк (lines)
- Цвет в которых их нужны закрасить (color)

Функция должна сохранить изображение по пути "img.png".

56.

(OpenCV)

Необходимо реализовать функцию solve, которая изменяет цвет пикселей, входящих в переданную область (окружность) изображения.

Цвет должен изменяться следующим образом - красная компонента должна заменяться на переданное значение.

Функция принимает на вход:

- Путь к RGB-изображению (img_path)
- Центр окружности (x_center,y_center)
- Радиус окружности (radius)
- Значение красной компоненты пикселя (red)

Функция должна сохранить изображение по пути "img.png"

57.

(OpenCV)

Необходимо реализовать функцию solve(), которая принимает на вход путь к RGB-изображению (img_path) и заменяет цвет каждого пикселя следующим образом:

-берется средний цвет пикселей в столбце, где находится пиксель

-берется средний цвет пикселей в строке, где находится пиксель

-затем берется средний цвет из этих двух

Функция должна сохранить измененное изображение по пути "img.png"

58.

(OpenCV)

Необходимо реализовать функцию solve(), которая принимает путь к RGB-изображению img_path и сортирует пиксели в нем по возрастанию суммы синей и зеленой компонент (g+b). Функция должна сохранить изображение с отсортированными пикселями по пути "img.png"

59.

(OpenCV)

Необходимо реализовать функцию solve, которая делит изображение на "полосы" и инвертирует красную и зеленую компоненты нечетных полос (счёт с нуля).

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Ширину полос в пикселах (N)

- Признак того, вертикальные или горизонтальные полосы(vertical - если True, то вертикальные)

Функция должна разделить изображение на вертикальные или горизонтальные полосы шириной N пикселей. И инвертировать цвет в нечетных полосах (счет с нуля). Последняя полоса может быть меньшей ширины, чем N.

Функция должна сохранить измененное изображение по пути "img.png".

60.

(OpenCV)

Необходимо написать функцию solve().

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Строку (text)
- Масштабирование шрифта (font_scale)
- Толщину текста (thickness)
- Координату начала текста (x_text,y_text)
- Координату пикселя (x_pixel,y_pixel)
- Шрифт (font)

Функция должна написать текст text на изображении с заданными параметрами (размер, координата, шрифт). Цвет текста нужно взять из цвета пикселя переданными координатами (x_pixel,y_pixel). Функция должна сохранить изображение по пути (img.png).

61.

(OpenCV)

Необходимо реализовать функцию solve(), которая вставляет одно изображение в другое.

Функция solve() принимает на вход:

Путь к RGB изображению 1 (img_from_path) Путь к RGB изображению 2 (img_to_path) Координаты левого верхнего угла области вставки(x,y) Функция должна вставить изображение 1 в изображение 2, начиная с координат (x,y).

Функция должна сохранить измененное изображение 2 по пути "img.png".

62.

(OpenCV)

Необходимо написать функцию solve(), которая реализует поворот квадратной части изображения на 90/180/270 градусов против часовой стрелки. Функция принимает на

вход:

- Путь к исходному RGB изображению (img_path)
- Координаты левого верхнего угла области (x, y)
- Ширину квадратной области (width)
- Угол поворота (angle - 90 или 180 или 270)

Функция должна сохранить изображение по пути "img.png"

63.

(OpenCV)

Необходимо реализовать функцию solve, которая меняет местами два одинаковых по размеру участка изображения. Участки не пересекаются. Затем функция должна перевернуть изображение на определенное количество градусов по часовой стрелке.

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Координаты левого верхнего угла первого участка(x0,y0)
- Координаты левого верхнего угла второго участка(x1,y1)
- Размеры области (width,height)
- Угол поворота (angle)
- Нужно ли обрезать изображение при повороте (crop - True или False)

Функция должна поменять участки изображения и сохранить изображение по пути "img.png".

64.

(OpenCV)

Необходимо реализовать функцию solve(), которая принимает на вход:

- Путь к RGB-изображению (img_path)
- Ширину изображения (width)
- Высоту изображения (height)
- Флаг - нужно ли сохранить пропорции (flag - True, False)

Функция должна изменить размер изображения. Если flag = False то нужно изменить размер изображения на (width,height). Иначе - нужно изменить размер изображения с сохранением пропорций, использовав только аргумент width (без height).

Функция должна сохранить изображение по пути "img.png".

65.

(OpenCV)

Необходимо написать функцию solve().

Функция solve() принимает на вход:

- Размер изображения (width,height)
- Первый цвет изображения (img_color1 - list)
- Второй цвет изображения (img_color2 - list)
- Строку (text)
- Цвет текста (text_color - list)
- Шрифт текста (font)
- Масштабирование шрифта (font_scale)
- Толщину текста (thickness)
- Координату начала текста (x,y)

Функция должна создать RGB-изображение заданных размеров (width,height), цвет столбцов изображения должен чередоваться: четные столбцы (0,2,4,...,2n) должны быть закрашены в цвет img_color1, нечетные столбцы (1,3,5,...,2n+1) - должны быть закрашены в цвет img_color2.

Затем на созданном изображении должен быть написан текст (text) с заданными параметрами (цвет, размер, координата). Изображение должно быть сохранено по пути "img.png".

66.

(OpenCV)

Необходимо реализовать функцию solve(), которая принимает на вход:

- Путь к RGB изображению (img_path)
- Координаты левого верхнего угла области (включительно) - (x0,y0)
- Координаты правого нижнего угла области (включительно) - (x1,y1)
- Ось отражения (vertical - если True, то вертикальная. Иначе - горизонтальная)

Функция должна отразить изображение в переданном участке по одной из осей. Измененное изображение нужно сохранить по пути "img.png".

67.

(OpenCV)

Необходимо реализовать функцию solve(), которая принимает на вход:

- Путь к RGB-изображению (img_path)
- Сдвиг по горизонтали (x_shift)
- Сдвиг по вертикали (y_shift)
- Цвет (color)

Функция должна сдвинуть изображение на указанные значения по обеим осям и изменить цвет части, которая не относится к изначальному изображению, на переданный. Измененное изображение должно быть сохранено по пути "img.png"

68.

(OpenCV)

Необходимо реализовать функцию solve, которая делит квадратное изображение на 9 равных частей (сторона изображения делится на 3), и по правилам, записанным в словаре, меняет их местами.

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Словарь с описанием того, какие части на какие менять (rules)
- Пример словаря rules:

{0:1,1:2,2:4,3:4,4:5,5:3,6:8,7:8,8:8}

Элементы нумеруются слева-направо, сверху-вниз.

В данном случае нулевой элемент заменяется на первый, первый на второй, второй на четвертый, третий на четвертый и так далее.

Функция должна сохранить измененное изображение по пути "img.png"

Пример входной картинки и словаря:

{0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8}

Результат: 

69.

(OpenCV)

Необходимо написать функцию solve().

Функция solve() принимает на вход:

- Путь к RGB-изображению (img_path)
- Количество изображений по "оси" Y (N - натуральное)
- Количество изображений по "оси" X (M - натуральное)

Функция должна создать коллаж изображений (это же изображение, повторяющееся NxM раз. (N раз по высоте, M раз по ширине) и сохранить его по пути "img.png".

Подсказка: numpy.tile

70.

Необходимо реализовать функцию solve(), которая принимает на вход 2 пути к файлам (path1, path2).

Функция должна скопировать содержимое по пути path1 и вставить это содержимое в файл по пути path2. Затем функция должна стереть содержимое файла по пути path1.

Если хоть один из файлов не существует в момент начала работы функции, то необходимо совершить выход из функции.

71.

Необходимо реализовать функцию solve(), которая принимает на вход путь к директории (path). Функция должна вернуть список из двух чисел [n,k], где n - количество папок в директории, k - количество файлов в директории (файлы и папки только первого уровня, без вложений).

Если директории нет, функция должна вернуть список [-1,-1].

72.

Необходимо реализовать функцию solve(), которая принимает на вход путь к директории (path) (точно существующей). Функция должна найти в этой директории файл с расширением ".txt" и с самым большим содержимым (метод read() вернет наибольшее количество символов) и вернуть название этого файла.

73.

Необходимо реализовать функцию solve(), которая принимает на вход путь к директории (path), название файла (name) и текст (text).

Функция должна создать в директории по пути path файл с именем name и записать в этот файл текст text.

Если такой директории не существует, то нужно в текущей директории создать файл с названием "not_exists.txt" и записать в него текст text.

74.

Необходимо реализовать функцию solve(), которая принимает на вход 2 числа - n, m.

Функция должна создать n вложенных папок с именами "dir_1","dir_2",...,"dir_m". В каждой папке должно быть создано m файлов, с названиями "file_1.txt","file_2.txt",...,"file_m.txt". В каждом файле должен быть соответствующий текст : "text_n_m". Например, для 2-го файла в 3-ей папке "text_3_2".

То есть при n=2, m=2 должна получить следующая структура:

-.

---dir_1

---------file_1.txt #text = "text_1_1"

---------file_2.txt #text = "text_1_2"

---------dir_2

----------file_1.txt #text = "text_2_1"

----------file_2.txt #text = "text_2_2"

75.

Необходимо реализовать функцию solve(), которая принимает на вход путь к директории (path), строку (subname).

Функция должна пройтись по всей директории с путем (path) (включая все поддиректории) и найти количество файлов, которые в названии содержат подстроку subname.

Функция должна вернуть количество таких файлов.

76.

Необходимо реализовать функцию solve(), которая принимает на вход путь к директории (path), строку (name).

Функция должна пройтись по всей директории с путем (path) (включая все поддиректории) и найти файл с именем name (он будет единственный). Функция должна прочитать и вернуть содержимое файла.

77.

Необходимо реализовать функцию solve, которая определяет высоту и ширину изображения из заголовка.

На вход функция принимает путь к изображению формата BMP (path).

Нужно открыть изображение с помощью стандартной функции open и определить размер изображения. Необходимо вернуть строку следующего формата: "x"(без треугольных скобок).

Пользоваться дополнительными библиотеками нельзя.

78.

Необходимо реализовать функцию solve, которая определяет высоту и ширину изображения из заголовка.

На вход функция принимает путь к изображению формата JPEG (path).

Нужно открыть изображение с помощью стандартной функции open и определить размер изображения. Необходимо вернуть строку следующего формата: "x"(без треугольных скобок).

Пользоваться дополнительными библиотеками нельзя.

79.

Необходимо реализовать функцию solve, которая определяет высоту и ширину изображения из заголовка.

На вход функция принимает путь к изображению формата PNG (path).

Нужно открыть изображение с помощью стандартной функции open и определить размер изображения. Необходимо вернуть строку следующего формата: "x"(без

треугольных скобок).

Пользоваться дополнительными библиотеками нельзя.

80.

Необходимо реализовать функцию solve, которая определяет формат изображения (PNG, JPEG или BMP).

На вход функция принимает файл изображения, открытый с помощью функции open: open(file_name,'rb').

Нужно определить формат изображения из заголовка. В зависимости от формата, нужно вернуть одну из строк: 'BMP', 'JPEG' или 'PNG'.

Пользоваться дополнительными библиотеками и полем name нельзя.