

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Информатика»
Тема: Введение в архитектуру компьютера

Студент гр. 2384

Кузьминых Е.М.

Преподаватель

Шевская Н.В.

Санкт-Петербург

2022

Цель работы.

Написать программу на языке программирования Python, содержащую в себе 3 функции, которые будут работать с модулем *Pillow (PIL)* и редактировать изображение.

Задачи.

Вариант №3.

Предстоит решить 3 подзадачи, используя библиотеку *Pillow (PIL)*. Для реализации требуемых функций студент должен использовать *numpy* и *PIL*. Аргумент *image* в функциях подразумевает объект типа `<class 'PIL.Image.Image'>`

1) Рисование пентаграммы в круге

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

Функция *solve()* принимает на вход:

Изображение (*img*)

координаты центра окружности (*x,y*)

радиус окружности

Толщину линий и окружности (*thickness*)

Цвет линий и окружности (*color*) - представляет собой список (*list*) из 3-х целых чисел

Функция должна изменить исходное изображение и вернуть его изображение.

2) Поменять местами участки изображения и поворот

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:

Квадратное изображение (*img*)

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. Затем повернуть каждый участок на 90 градусов по часовой стрелке. Затем повернуть всё изображение на 90 градусов по часовой стрелке.

Функция должна вернуть обработанное изображение, не изменяя исходное.

3) Средний цвет

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

Функция *solve()* принимает на вход:

Изображение (*img*)

Координаты левого верхнего угла области ($x0, y0$)

Координаты правого нижнего 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 углу

Функция должна вернуть обработанное изображение, не изменяя исходное.

Выполнение работы.

В начале программы мы подключим необходимые для работы модули – *Pillow* и *NumPy*.

В функции *pentagram* мы принимаем изображение, координаты центра и радиус круга, в который вписана пентаграмма, цвет и толщина линий, которыми будет нарисована пентаграмма. Мы создаем *line_color*, в котором будем хранить цвет для линий в формате *rgb*, находим координаты четырехугольника, в который вписан круг и с помощью *ImageDraw.Draw(img).ellipse(coordinates, outline = tuple(color), width=thickness)* рисуем его. После с помощью формулы для расчета координат вершин пентаграммы, которую дали в условии задачи, находим координаты и добавляем их в список *cords*, а в цикле рисуем линии с противоположными координатами. Возвращаем измененное изображение.

В функции *swap* мы принимаем изображение, координаты 2 квадратов и длину их стороны. Сначала мы создаем копию изображения. С помощью функций *crop()* и *rotate()* мы вырезаем и поворачиваем 2 нужных участка изображения, а с помощью *paste()* вставляем эти участки в копию изображения и возвращаем его.

В функции *avg_color()* мы принимаем изображение и координаты области, с которой будем работать. В начале создаем копию изображения и в *coord_x* и *coord_y* сохраняем крайние координаты по Ох и Оу. В вложенном цикле мы перебираем каждый пиксель в области. Мы создаем список из возможных координат пикселей вокруг пикселя, для которого мы считаем средний цвет, проверяем, не выходит ли он за пределы области изображения, после берем цвет этого изображения с помощью функции *getpixel()* и сохраняем его в *old_pixel*. Затем мы считаем средний цвет по формуле, данной в условии

задачи и с помощью функции *putpixel()* изменяем цвет итерируемого пикселя. Функция возвращает измененную копию изображения.



Тестирование.

В качестве фонового изображения возьмем следующее фото.



Рисунок 1 -изображение для редактирования

№	Входные данные	Выходные данные		Комментарий
1	pentagram(img,150,150,150,15, (78,79,248))			Ответ верный

2	<code>swap(img,0,0,150,150,150)</code>		Ответ верный
3	<code>avg_color(img,0,0,150,150)</code>		Ответ верный

Вывод.

В ходе выполнения лабораторной работы была изучена работа с модулем Pillow. Была создана программа, содержащая в себе 3 функции, которые изменяют поданное на вход изображение.

Приложение 1. Исходный код программы.

```
import PIL
import numpy as np
import math
from PIL import Image, ImageDraw

def pentagram(img, x, y, r, thickness, color):
    line_color = (color[0], color[1], color[2])
    draw = ImageDraw.Draw(img)
    coordinates = ((x-r,y-r), (x+r,y+r))
    draw.ellipse(coordinates, outline = tuple(color), width=thickness)
    coords = []
    for i in range(0,6):
        phi = (math.pi/5)*(2*i+3/2)

        node_i = (int(x+r*math.cos(phi)),int(y+r*math.sin(phi)))

        coords.append(node_i)
    for i in range(0, 5):
        draw.line((coords[i], coords[(i + 2) % 5]), fill=line_color,
width=thickness)
    return img

def swap(img, x0,y0,x1,y1,width):
    new_image = img.copy()
    img1= img.crop((x0, y0, x0+width, y0+width))
    img1=img1.rotate(270)
    img2= img.crop((x1, y1, x1+width, y1+width))
    img2=img2.rotate(270)
    new_image.paste(img2, (x0,y0))
    new_image.paste(img1, (x1,y1))
    new_image = new_image.rotate(270)
    return new_image

def avg_color(img, x0, y0, x1, y1):
    new_image = img.copy()
    coord_x= img.size[0]
    coord_y=img.size[1]
    for x in range(x0,x1+1):
        for y in range(y0,y1+1):
            colors=[0,0,0]
            #перебор всех точек и проверка их на вхождение в изображение
            coordinats=[(x+1,y), (x-1,y), (x+1,y+1), (x+1,y-1), (x-1,y+1), (x-
1,y-1), (x,y+1), (x,y-1)]
            for i in range(len(coordinats)):
                if (not(0 <= coordinats[i][0] <= coord_x)) or (not(0 <=
coordinats[i][1] <= coord_y)):
                    coordinats.pop(i)
                    i-=1
            count_pixels=len(coordinats)
            for i in range(len(coordinats)):
                old_pixel = img.getpixel(coordinats[i])
                colors[0]+=old_pixel[0]
```

```
        colors[1]+=old_pixel[1]
        colors[2]+=old_pixel[2]
        new_color      =      (int(colors[0]/count_pixels),
int(colors[1]/count_pixels),int(colors[2]/count_pixels))
        new_image.putpixel((x,y),new_color)

    return new_image
```