

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

**ОТЧЕТ**  
**по лабораторной работе №1**  
**по дисциплине «Информатика»**  
**Тема: Введение в архитектуру компьютера**

Студентка гр. 3343

Лобова Е. И.

Преподаватель

Иванов Д. В.

Санкт-Петербург

2023

## **Цель работы**

Целью работы является освоение работы с изображениями в языке Python, используя методы библиотеки Pillow.

## Задание

### 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.

Примечание:

Вершины пентаграммы высчитывать по формуле:

$$\text{phi} = (\pi/5) * (2 * i + 3/2)$$

$$\text{node\_i} = (\text{int}(x_0 + r * \cos(\text{phi})), \text{int}(y_0 + r * \sin(\text{phi})))$$

$x_0, y_0$  - координаты центра окружности, в который вписана пентаграмма

$r$  - радиус окружности

$i$  - номер вершины от 0 до 4

#### 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( $x_0, y_0$ )
- Координаты левого верхнего угла второго квадратного участка( $x_1, y_1$ )
- 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.

Пример входной картинки (300x300) и переданных параметров:



Рисунок 1 – Пример входной картинки (300x300) с переданными параметрами  $x_0 = 0$  ;  $y_0 = 0$ ;  $x_1 = 150$ ;  $y_1 = 150$ ; width = 150

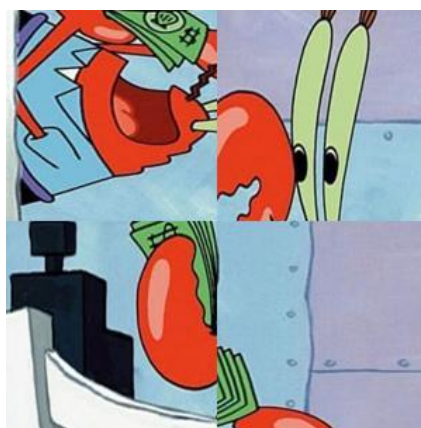


Рисунок 2 - Изображение после первого этапа (замена участков и поворот этих участков на 90 градусов)

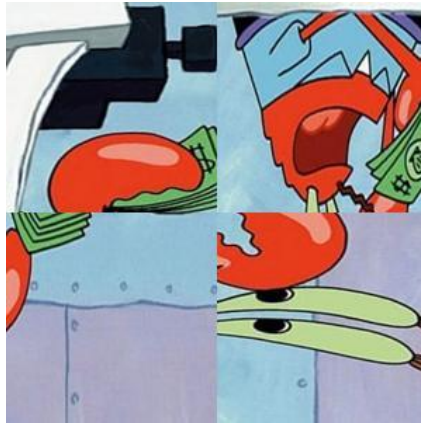


Рисунок 3 – Итоговое изображение

### 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.

Средний цвет - берется целая часть от среднего каждой компоненты из rgb.

$(\text{int}(\text{sum}(r)/\text{count}), \text{int}(\text{sum}(g)/\text{count}), \text{int}(\text{sum}(b)/\text{count}))$

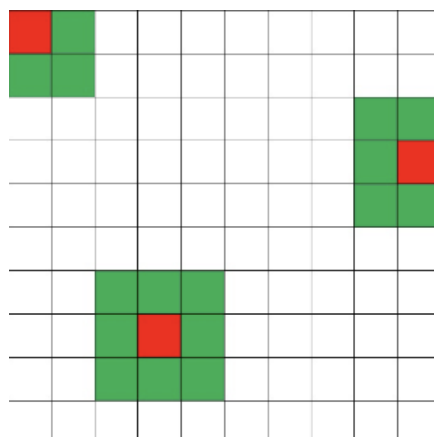


Рисунок 4 – Изображения для примера получения среднего цвета  
Для красных пикселей, берется средний цвет зеленых пикселей вокруг него.

## Выполнение работы

Функции, реализованные в программе:

- Функция *pentagram(img, x, y, r, thickness, color)* принимает на вход изображение, на котором рисует пентаграмму по заданным параметрам для окружности, с заданными толщиной и цветом для контура. В первую очередь рисуется окружность с помощью метода *ImageDraw.ellipse(frame\_coordinates, fill=None, outline=None, width=1)*. Далее в массив *peeks* записываются координаты вершин звезды, после чего в этот же массив его элементы записываются в том порядке, в котором их нужно соединить для получения нужной фигуры. Линии рисуются с помощью метода *ImageDraw.line(coordinates, fill=None, width=0, joint=None)*. Функция возвращает преобразованное изображение.
- Функция *swap(img, x0,y0,x1,y1, width)* принимает на вход изображение, координаты левых верхних углов участков, которые следует поменять местами и их длину стороны. В первую очередь создается копия изображения. Далее вырезаются два участка с помощью метода *Image.crop(box)*, поворачиваются на 90 градусов по часовой с помощью метода *Image.transpose(direction)* и помещаются в переменные *img1* и *img2*, после чего вставляются на копию изображения с помощью метода *Image.paste(other\_image, coordinates)*. Затем вся копия изображения поворачивается на 90 градусов по часовой, она же и возвращается.
- Функция *avg\_color(image, x0, y0, x1, y1)* принимает на вход изображение и координаты области, в которой необходимо заменить цвет каждого пикселя. В первую очередь создается копия изображения. Далее два цикла проходятся по каждому пикселю нужной области и еще один цикл проходится по 8 соседним координатам, в котором с помощью метода *Image.getpixel(box)* у подходящих пикселей берутся компоненты из RGB и суммируются с

предыдущими компонентами. После обработки соседних пикселей высчитывается новый цвет и меняется пиксель с помощью метода *Image.putpixel(box, color)*. Функция возвращает измененную копию изображения.

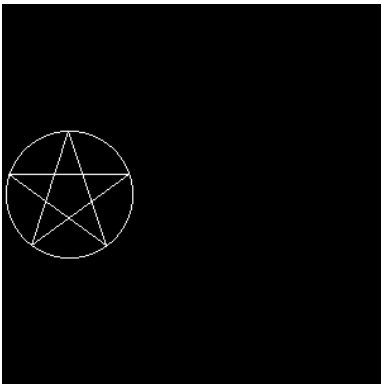
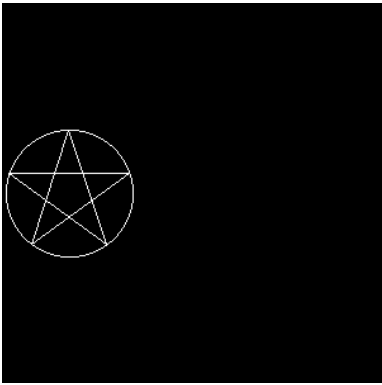
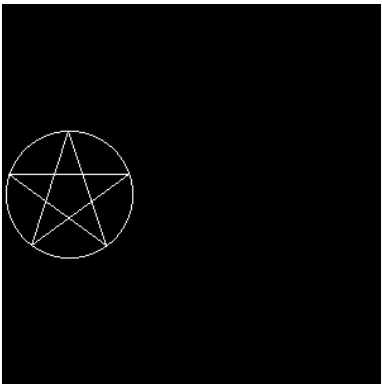
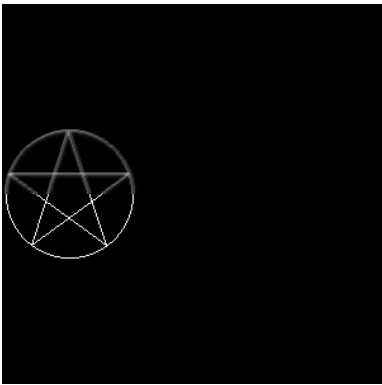
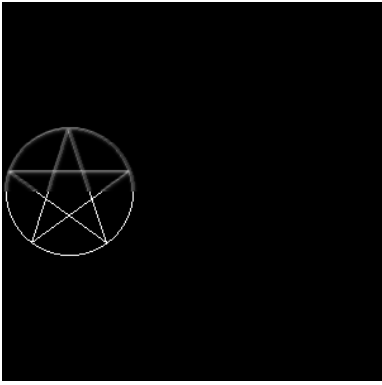
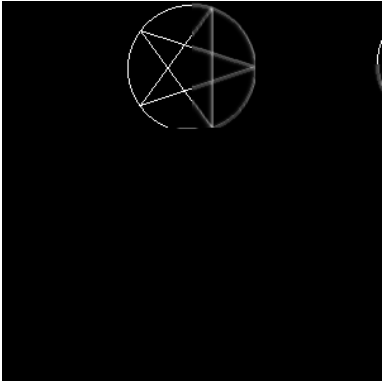
Разработанный программный код см. в приложении А.



## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	 <pre>img = Image.new("RGB", (300, 300), [0, 0, 0]) ; x = 53; y = 150; r = 50; thickness = 1; color = [256, 256, 256]</pre>		Функция <i>pentagram(img, x, y, r, thickness, color)</i> работает корректно.
2.	 <pre>x0 = 0; y0 = 0; x1 = 150; y1 = 150</pre>		Функция <i>avg_color(image, x0, y0, x1, y1)</i> работает корректно.
3.	 <pre>x0 = 0; y0 = 0; x1 = 100; y1 = 100, width = 100</pre>		Функция <i>swap(img, x0, y0, x1, y1, width)</i> работает корректно.

## Выводы

Были изучены основные принципы работы с изображениями в Python, особое внимание было уделено работе с модулем Pillow и его пакетам Image, ImageDraw.

Разработана часть программы, содержащая в себе 3 функции. Для реализации работы с изображениями использовались методы *ImageDraw.ellipse(frame\_coordinates, fill=None, outline=None, width=1)*, *ImageDraw.line(coordinates, fill=None, width=0, joint=None)*, *Image.crop(box)*, *Image.transpose(direction)*, *Image.paste(other\_image, coordinates)*, *Image.getpixel(box)*, *Image.putpixel(box, color)*. Для перебора пар элементов массива использовались циклы *for*. Для проверки выполнения заданного условия использовался условный оператор *if – else*.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from PIL import Image, ImageDraw
import numpy as np
from math import pi, cos, sin

def pentagram(img, x, y, r, thickness, color):
    color=tuple(color)
    coordinates=((x-r,y-r),(x+r,y+r))
    drawing = ImageDraw.Draw(img)
    drawing.ellipse(coordinates, None, color, thickness)
    peeks=[]
    for i in range(5):
        phi = (pi / 5) * (2 * i + 3 / 2)
        peek = (int(x + r * cos(phi)), int(y + r * sin(phi)))
        peeks.append(peek)
    peeks = [peeks[0], peeks[2], peeks[4], peeks[1], peeks[3],
peeks[0]]
    drawing.line(peeks, fill=tuple(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)).transpose(Image.Transpose.ROTATE
_270)

img2=img.crop((x1,y1,x1+width,y1+width)).transpose(Image.Transpose.ROTATE
_270)
    new_image.paste(img1,(x1,y1))
    new_image.paste(img2, (x0, y0))
    new_image = new_image.transpose(Image.Transpose.ROTATE_270)
    return new_image

def avg_color(image, x0, y0, x1, y1):
    new_image = image.copy()
    for x in range(x0, x1 + 1):
        for y in range(y0, y1 + 1):
            r_sum, g_sum, b_sum = 0, 0, 0
            count = 0
            for i in range(x-1, x+2):
                for j in range(y-1, y+2):
                    if i < 0 or j < 0 or i >= image.width or j >=
image.height or (x == i and y == j):
                        continue
                    pixel_r, pixel_g, pixel_b = image.getpixel((i,
j))
                    r_sum += pixel_r
                    g_sum += pixel_g
                    b_sum += pixel_b
                    count += 1
            new_color = (int(r_sum / count), int(g_sum / count),
int(b_sum / count))
            new_image.putpixel((x, y), new_color)
```

```
return new_image
```