

**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. 3344

Волков А.А.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

## **Цель работы**

Ознакомиться с использованием библиотеки Pillow в сочетании с библиотекой NumPy для обработки изображений.

## 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 `pentagram()`, которая рисует на изображении пентаграмму в окружности.

Функция `pentagram()` 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 вычислять по формуле:

$$\phi_i = (\pi/5) * (2*i + 3/2)$$

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

$x_0, y_0$  - 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 `swar()`, 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 `swar()` 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 `avg_color()`, которая заменяет цвет каждого пикселя в области на средний цвет пикселей вокруг (не считая сам этот пиксель).

Функция `avg_color()` 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 угле

Функция должна вернуть обработанное изображение, не изменяя исходное.

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

## Выполнение работы

Подключается библиотека *Numpy* и требуемые модули библиотеки *Pillow*(*Image*, *ImageDraw*).

Функции:

1. Функция *pentagram(img, x, y, r, thickness, color)*:

Получает на вход изображение, координаты центра окружности, радиус окружности, толщину и цвет линий окружности. Вычисляет координаты левого верхнего и правого нижнего углов квадрата, в который вписана окружность, для того, чтобы её нарисовать. С помощью цикла *for* и формул вычисляет координаты вершин пентаграммы. Затем с помощью цикла *for* и метода *line* рисует линии между её вершинами. Возвращает изменённое исходное изображение с нарисованной пентаграммой.

2. Функция *swap(img, x0, y0, x1, y1, width)*:

Получает на вход квадратное изображение, координаты левых верхних углов первого и второго квадратных участков, а также длину стороны квадратных участков. В переменную *img\_copy* копирует исходное изображение с помощью метода *copy*. С помощью метода *crop* выделяет два участка и поворачивает их на 90 градусов по часовой стрелке с помощью *transpose*. После заменяет местами эти участки на изображении с помощью метода *paste*. В конце поворачивает всё изображение на 90 градусов по часовой стрелке с использованием метода *transpose*. Возвращает изображение, которое получено из исходного, повернутое на 90 градусов, где данные участки поменяны местами и тоже повернуты на 90 градусов по часовой стрелке.

3. Функция *avg\_color(img, x0, y0, x1, y1)*

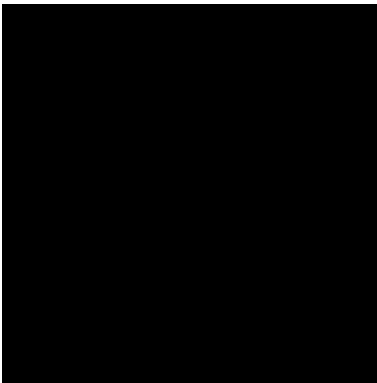



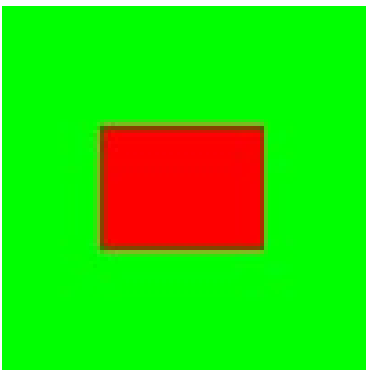
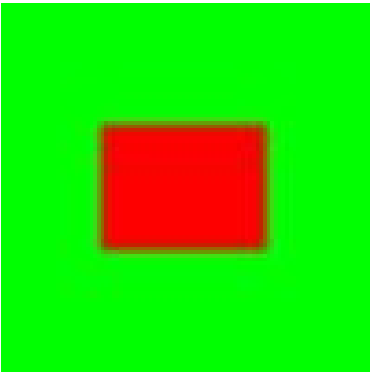
Получает на вход изображение и координаты левого верхнего и правого нижнего углов области, в которой нужно произвести замену цвета пикселей. В переменную *img\_copy* копирует исходное изображение с помощью метода *copy*. Для каждого пикселя в заданной области создаётся кортеж из 8 соседей, затем кортеж фильтруется при помощи функции *filter*, оставляя в нём только подходящие пиксели. С помощью цикла *for* проходит по всем нужным

пикселям, которые остались в кортеже, и при помощи *getpixel* выделяет их цвет. Далее каждая компонента цвета суммируется в нужной ячейке списка *rgb*. Средний цвет получается взятием целой части от среднего каждой компоненты из *rgb*. Возвращает изображение, которое получено из исходного, где цвета всех пикселей в определенной области были заменены на средний цвет близлежащих пикселей вокруг.

## Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.			<code>pentagram(img, 150, 150, 100, 5, (255, 0 , 0))</code>
2.			<code>swap(img, 0, 0, 150, 150, 150)</code> <code>size = (300, 300)</code>
3.			<code>avg_color(img, 0, 0, 99, 99)</code> <code>size = (100, 100)</code>



## **Выводы**

Были освоены основы работы с библиотекой Pillow, которые применялись при решении заданий.

Была разработана программа, состоящая из трёх функций, каждая из которых по-своему изменяет изображение: рисует пентаграмму внутри окружности на изображении; меняет местами два квадратных участка изображения и поворачивает их на 90 градусов по часовой стрелке, затем поворачивает всё изображение на 90 градусов по часовой стрелке; заменяет цвет каждого пикселя в области на средний цвет пикселя вокруг.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: lb\_2.py

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

```
def swap(img, x0, y0, x1, y1, width):
    img_copy = img.copy()
    first = img_copy.crop((x0, y0, x0 + width, y0 +
width)).transpose(Image.Transpose.ROTATE_270)
    second = img_copy.crop((x1, y1, x1 + width, y1 +
width)).transpose(Image.Transpose.ROTATE_270)
    img_copy.paste(first, (x1, y1))
    img_copy.paste(second, (x0, y0))
    img_copy = img_copy.transpose(Image.Transpose.ROTATE_270)
    return img_copy
```

```
def avg_color(img, x0, y0, x1, y1):
    img_copy = img.copy()
    size = img.size
    for x in range(x0, x1 + 1):
        for y in range(y0, y1 + 1):
            neighbor_pixels = (
                (x, y + 1),
                (x, y - 1),
                (x + 1, y),
                (x - 1, y),
                (x + 1, y + 1),
                (x + 1, y - 1),
                (x - 1, y - 1),
                (x - 1, y + 1),
            )
            neighbor_pixels = tuple(
                filter(lambda n: (0 <= n[0] <= size[0] - 1) and (0 <=
n[1] <= size[1] - 1), neighbor_pixels))
            len_neighbor_pixels = len(neighbor_pixels)

            rgb = [0, 0, 0]
            for i in neighbor_pixels:
                pixel_color = img.getpixel(i)
                rgb[0] += pixel_color[0]
                rgb[1] += pixel_color[1]
                rgb[2] += pixel_color[2]
            new_color = (
                int(rgb[0] / len_neighbor_pixels),
                int(rgb[1] / len_neighbor_pixels),
                int(rgb[2] / len_neighbor_pixels),
            )
            img_copy.putpixel((x, y), new_color)
    return img_copy
```

```

def pentagram(img, x, y, r, thickness, color):
    drawing = ImageDraw.Draw(img)
    color = tuple(color)
    coordinates = ((x - r, y - r), (x + r, y + r))
    drawing.ellipse(coordinates, outline=color, width=thickness)
    nodes = []
    for i in range(5):
        phi = (np.pi / 5) * (2 * i + 3 / 2)
        node_i = (int(x + r * np.cos(phi)), int(y + r * np.sin(phi)))
        nodes.append(node_i)
    for i in range(5):
        drawing.line((nodes[i], nodes[(i + 2) % 5]), fill=color,
width=thickness)
    return img

```