

**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: Основные управляющие конструкции языка Python

Студен гр. 3343

Преподаватель

---

---

Кербель Д. А.

Иванов Д. В.

Санкт-Петербург

2023

## **Цель работы**

Изучить и научиться применять библиотеку языка Python Pillow (PIL) и Numpy.

## **Задание**

Вариант лабораторной работы состоит из 3 задач, необходимо оформить каждую задачу в виде отдельной функции согласно условиям задач. Вы можете реализовывать вспомогательные функции, главное — использовать те же названия основных функций, что требуются в задании. Сами функции вызывать не надо, это делает за вас проверяющая система.

### **Задача 1.**

Рисование отрезка. Отрезок определяется: координатами начала, координатами конца, цветом толщиной.

Необходимо реализовать функцию `user_func()`, рисующую на картинке отрезок

Функция `user_func()` принимает на вход: изображение; координаты начала ( $x_0, y_0$ ); координаты конца ( $x_1, y_1$ ); цвет; толщину.

Функция должна вернуть обработанное изображение.

### **Задача 2.**

Преобразовать в Ч/Б изображение (любым простым способом).

Функционал определяется: Координатами левого верхнего угла области; Координатами правого нижнего угла области; Алгоритмом, если реализовано несколько алгоритмов преобразования изображения (по желанию студента).

Нужно реализовать 2 функции: `check_coords(image, x0, y0, x1, y1)` - проверяет координаты области ( $x_0, y_0, x_1, y_1$ ) на корректность (они должны быть неотрицательными, не превышать размеров изображения, поскольку  $x_0$ ,

$y_0$  - координаты левого верхнего угла,  $x_1$ ,  $y_1$  - координаты правого нижнего угла, то  $x_1$  должен быть больше  $x_0$ , а  $y_1$  должен быть больше  $y_0$ ); `set_black_white(image, x0, y0, x1, y1)` - преобразовывает заданную область изображения в черно-белый (используйте для конвертации параметр '1'). В этой функции должна вызываться функция проверки, и, если область некорректна, то должно быть возвращено исходное изображение без изменений. Примечание: поскольку черно-белый формат изображения (greyscale) является самостоятельным форматом, а не вариацией RGB-формата, для его получения необходимо использовать метод `Image.convert`.

### **Задача 3. Содержательная часть задачи**

Найти самый большой прямоугольник заданного цвета и перекрасить его в другой цвет. Функционал определяется: Цветом, прямоугольник которого надо найти Цветом, в который надо его перекрасить.

Написать функцию `find_rect_and_recolor(image, old_color, new_color)`, принимающую на вход изображение и кортежи rgb-компонент старого и нового цветов. Она выполняет задачу и возвращает изображение. При необходимости можно писать дополнительные функции.

## Выполнение работы

Для выполнения поставленных задач, мною была написана программа на языке Python, в которой описываются функции для решения поставленных задач.

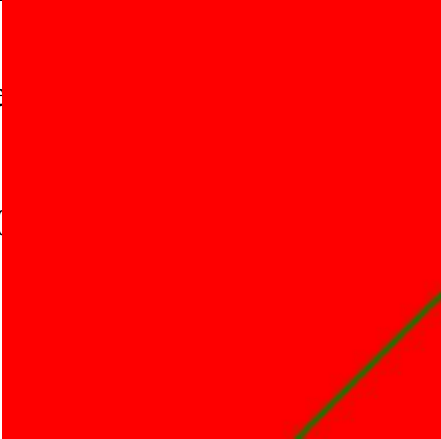
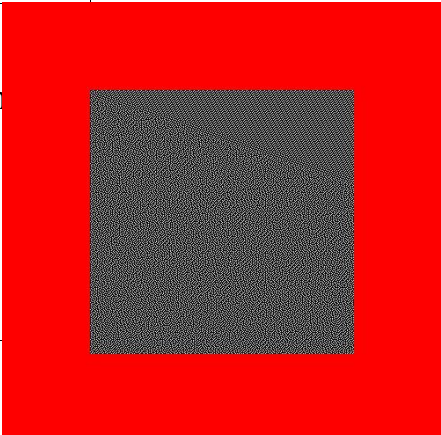
Функция `user_func` принимает изображение `image` и координаты двух точек  $(x_0, y_0)$  и  $(x_1, y_1)$ , а также цвет `fill` и ширину линии `width`. Функция использует библиотеку PIL (Python Imaging Library) для рисования линии между заданными точками на изображении. Функция возвращает измененное изображение. Функция `check_coords` принимает изображение `image` и координаты двух точек  $(x_0, y_0)$  и  $(x_1, y_1)$ . Функция проверяет, что все координаты положительны (больше нуля), а также что  $(x_1, y_1)$  больше  $(x_0, y_0)$  и находятся в пределах размеров изображения. Если все условия выполняются, функция возвращает `True`, иначе — `False`. Функция `set_black_white` принимает изображение `image` и координаты двух точек  $(x_0, y_0)$  и  $(x_1, y_1)$ . Функция использует функцию `check_coords` для проверки правильности координат. Если координаты верны, функция обрезает изображение по данным координатам, конвертирует его в черно-белый формат и заменяет обрезанную часть исходного изображения на новое черно-белое изображение. Возвращается измененное изображение. Функция `find_rect_and_recolor` принимает изображение `image`, старый цвет `old_color` и новый цвет `new_color`. Функция преобразует изображение в массив `numpy`, затем проверяет каждый пиксель на соответствие старому цвету и заменяет его на 1, если цвет соответствует, или на 0 в противном случае. Затем функция выполняет поиск прямоугольника из пикселей со значением 1 с наибольшей площадью. Затем функция заменяет цвет пикселей в найденном прямоугольнике на новый цвет и возвращает измененное изображение.

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.	<code>image user_func(Image.new("RGB", (300, 300), 0, 0)), 10, 490, 490 'green', 3)</code>		Выходные данные соответствуют ожиданиям.
2.	<code>image set_black_white(Image.new("RGB", (500, (255, 0, 0)), 100, 400, 400)</code>		Выходные данные соответствуют ожиданиям.



3.	<pre>image = Image.new("RGB", (500, 500), (255, 0, 0)) image.paste(Image.new("RGB", (100, 15), (0, 255, 0)), (10, 5)) image.paste(Image.new("RGB", (200, 20), (0, 255, 0)), (30, 20))</pre>	Выходные данные соответствуют ожиданиям.
----	---	--

## **Выводы**

В ходе выполнения лабораторной работы были изучены методы работы с модулем *Pillow*.



## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
from PIL import Image, ImageDraw
```

```
import numpy as np
```

```
# Функция 1
```

```
# Функция user_func принимает изображение image и координаты двух точек (x0, y0) и (x1, y1), а также цвет fill и ширину линии width
```

```
def user_func(image, x0, y0, x1, y1, fill, width):
```

```
    draw = ImageDraw.Draw(image)
```

```
# Функция использует библиотеку PIL (Python Imaging Library) для рисования линии между заданными точками на изображении
```

```
    draw.line(((x0, y0), (x1, y1)), fill, width)
```

```
# Функция возвращает измененное изображение
```

```
    return image
```

```
# Функция 2
```

```
# Функция check_coords принимает изображение image и координаты двух точек (x0, y0) и (x1, y1)
```

```
def check_coords(image, x0, y0, x1, y1):
```

```
    size_y = image.size[1]
```

```
    size_x = image.size[0]
```

```
# Функция проверяет, что все координаты положительны (больше нуля), а также что (x1, y1) больше (x0, y0) и находятся в пределах размеров изображения
```

```
    if (x0 > 0 and y0 > 0 and x1 > 0 and y1 > 0) and (x1 > x0 and y1 > y0) and (x1 < size_x and y1 < size_y):
```

```
# Если все условия выполняются, функция возвращает True, иначе - False
```

```
    return True
```

else:

return False

# Функция set\_black\_white принимает изображение image и координаты двух точек (x0, y0) и (x1, y1)

def set\_black\_white(image, x0, y0, x1, y1):

# Функция использует функцию check\_coords для проверки правильности координат

if check\_coords(image, x0, y0, x1, y1):

# если координаты верны, функция обрезает изображение по данным координатам, конвертирует его в черно-белый формат и заменяет обрезанную часть исходного изображения на новое черно-белое изображение

img\_convertd = image.crop((x0, y0, x1, y1))

img\_convertd = img\_convertd.convert("1")

image.paste(img\_convertd, (x0, y0))

# Возвращается измененное изображение.

return image

# Функция 3

# Функция find\_rect\_and\_recolor принимает изображение image, старый цвет old\_color и новый цвет new\_color

def find\_rect\_and\_recolor(image, old\_color, new\_color):

# Функция преобразует изображение в массив numpy, затем проверяет каждый пиксель на соответствие старому цвету и заменяет его на 1, если цвет соответствует, или на 0 в противном случае

color = list(old\_color)

array = np.array(image).tolist()

for i in range(len(array)):

for j in range(len(array[i])):

array[i][j] = int(array[i][j] == color)

array = np.array(array)

```

for i in range(1, len(array)):

    for j in range(len(array[i])):

        if array[i][j] == 1:
            array[i][j] += array[i - 1][j]

max_size = 0

coords = (0, 0, 0, 0)

for i in range(len(array)):

    curr_size = 0

    last_j = 0

    for j in range(len(array[i]) - 1):

        curr_size += array[i][j]

        if curr_size > max_size:
            max_size = curr_size

        coords = (j - (max_size // array[i][j]) + 1, i - array[i][j] + 1, j, i)

        if array[i][j] != array[i][j + 1]:
            curr_size = 0

    last_j = j

    if array[i][last_j] == array[i][last_j + 1]:

        curr_size += array[i][last_j + 1]

        if curr_size > max_size:
            max_size = curr_size

        coords = (last_j - (max_size // array[i][last_j + 1]) + 1, i -
array[i][last_j + 1] + 1, last_j + 1, i)

array = np.array(image).tolist()

```

```
for i in range(coords[1], coords[3] + 1):  
  
    for j in range(coords[0], coords[2] + 1):  
        array[i][j] = new_color  
  
image = Image.fromarray(np.uint8(array))  
  
# Затем функция заменяет цвет пикселей в найденном  
прямоугольнике на новый цвет и возвращает измененное изображение  
return image
```