**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 |  | Кербель Д. А. |
| Преподаватель |  | 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 (x0, y0); координаты конца (x1, y1); 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 (x0, y0, x1, y1) на корректность (они должны быть неотрицательными, не превышать размеров изображения, поскольку x0, y0 - координаты левого верхнего угла, x1, y1 - координаты правого нижнего угла, то x1 должен быть больше x0, а y1 должен быть больше y0); 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 (x0, y0) и (x1, y1), 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 (x0, y0) и (x1, y1). Функция проверяет, что все координаты положительны (больше нуля), а также что (x1, y1) больше (x0, y0) 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 (x0, y0) и (x1, y1). 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 |
|  | image = user\_func(Image.new("RGB", (300, 300), (500, 0, 0)), 10, 490, 490, 10, 'green', 3) |  | Выходные данные соответствуют ожиданиям. |
|  | image = set\_black\_white(Image.new("RGB", (500, 500), (255, 0, 0)), 100, 100, 400, 400) |  | Выходные данные соответствуют ожиданиям. |
|  | 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)) |  | 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):

# сли 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