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

Студентка гр. 3341 _____ Шуменков А.П.

Преподаватель _____ Иванов Д.В.

Санкт-Петербург
2023

Цель работы

Целью работы является изучение библиотеки Pillow, решение 3 подзадач с использованием библиотеки Pillow (PIL), работа с объектом типа <class 'PIL.Image.Image'>

Задание

Вариант 4

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

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.

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.

3) 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

1 задача:

Функция user_func принимает изображение, координаты начальной и конечной точек линии, цвет и ширину линии. Она создает объект ImageDraw для рисования на изображении, затем рисует линию с заданными координатами, цветом и шириной. В конце, функция возвращает измененное изображение.

Функция check_coords принимает изображение и координаты начальной и конечной точек линии. Она также получает ширину и высоту изображения. Функция проверяет следующие условия:

Если какая-либо из координат меньше 0, то функция возвращает False, означая что координаты выходят за границы изображения.

Если какая-либо из координат больше или равна ширине или высоте изображения, то функция также возвращает False, означая что координаты выходят за границы изображения.

Если конечная точка линии находится перед начальной точкой линии (т.е. x1 <= x0 или y1 <= y0), то функция возвращает False, означая что координаты заданы некорректно. Если ни одно из условий не выполняется, то функция возвращает True, означая что координаты корректны.

2 задача:

Функция set_black_white(image, x0, y0, x1, y1) принимает изображение image и координаты прямоугольной области на изображении (x0, y0, x1, y1).

Сначала функция проверяет, что координаты x0, y0, x1, y1 находятся внутри границ изображения с помощью функции check_coords(image, x0, y0, x1, y1). Если проверка прошла успешно, то функция вырезает часть изображения с помощью метода crop() и сохраняет в переменную img_one. Затем создается черно-белая копия вырезанной части изображения с помощью метода convert("1").

Далее, черно-белая копия изображения g_img вставляется в исходное изображение image в исходные координаты (x0, y0) с помощью метода paste().

В конце, функция возвращает измененное изображение image.

Функция check_coords(image, x0, y0, x1, y1) принимает изображение image и координаты прямоугольной области на изображении (x0, y0, x1, y1).

Функция сначала определяет размеры изображения image с помощью метода size(), и сохраняет их в переменные x и y. Затем функция проверяет, что координаты x0, y0, x1, y1 удовлетворяют следующим условиям: ширина и высота прямоугольной области больше либо равны нулю и меньше или равны соответствующим размерам изображения, а также все четыре координаты больше или равны нулю.

Функция возвращает результат проверки - True, если все условия выполняются, и False, если хотя бы одно из условий не выполняется.

3 задача:

Функция helperpr принимает следующие параметры:

x, y: координаты точки в изображении, width, height: ширина и высота изображения, datapx: массив пикселей изображения ,old_color: текущий цвет пикселя.

Функция выполняет следующие действия:

Инициализирует переменные top и bottom как массивы с начальными значениями [0, 0] и [width, height] соответственно.

Инициализирует переменную curr как массив с начальными значениями [(x, y)].

Пока массив curr не пуст:

Извлекает первый элемент из curr и присваивает его значения x1, y1.

Проверяет, что x1 и y1 находятся в пределах изображения и являются пикселями с цветом old_color. Обновляет значения переменных bottom и top, чтобы обрамить найденную область с пикселями old_color. Добавляет координаты соседних пикселей (x1-1, y1), (x1+1, y1), (x1, y1-1), (x1, y1+1) в curr. Устанавливает цвет пикселя (x1, y1) в (0, 0, 0) (черный цвет). Возвращает

координаты области с пикселями old_color в виде кортежа (bottom[0], bottom[1], top[0], top[1]).

Функция find_rect_and_recolor принимает следующие параметры:

image: изображение, old_color: текущий цвет пикселей, которые нужно заменить, new_color: новый цвет пикселей, которыми нужно заменить, Функция выполняет следующие действия:

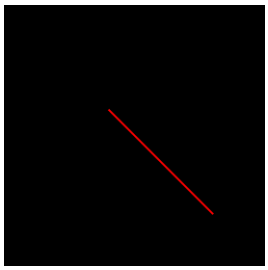
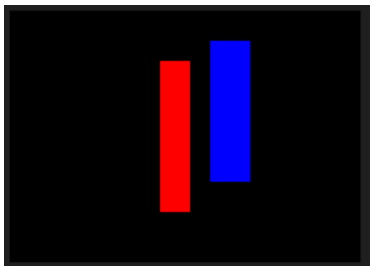
Инициализирует переменную max со значением 0. Создает копию изображения image и называет ее cimg. Загружает пиксели изображения cimg в переменную datapx. Получает ширину и высоту изображения и присваивает их переменным height и width соответственно. Проходит по всем пикселям изображения в двойном цикле for x in range(width) и for y in range(height). Если пиксель имеет цвет old_color, вызывает функцию helperpr с передачей текущих координат пикселя и других параметров. Вычисляет площадь прямоугольника, ограничивающего найденную область, как (crds[3]+1 - crds[1]) * (crds[2]+1 - crds[0]). Если площадь прямоугольника больше значения max, обновляет max и сохраняет координаты прямоугольника в переменной crds_max. Загружает пиксели изображения image в переменную image_r. Проходит по всем пикселям внутри координат прямоугольника crds_max и устанавливает их цвет в new_color. Возвращает измененное изображение image.

См. приложение А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

п/п	Входные данные	Выходные данные	Комментарии
			
	image=Image.new("RGB", (500, 500), (0,0,0)) img=user_func(image, 400, 400, 200, 200, (255,0,0,0), 3) Image._show(img)		№1
	set_black_white(Image.open("input.png"), 250, 200, 700, 500)		№2
	image = Image.new("RGB", (350, 250), 'black') image.paste(Image.new("RGB", (30,150), 'red'), (150, 50)) image.paste(Image.new("RGB", (40,140),		№3

	'red'), (200, 30)) find_rect_and_recolor(image, (255, 0, 0), (0, 0, 255))		

Выводы

Изучена библиотека Pillow, решены 3 подзадачи с использованием библиотеки Pillow (PIL).

Реализована программа, состоящая из трех задач, под каждую из которых выделена отдельная функция.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
import PIL
from PIL import Image, ImageDraw

#1

def user_func(image, x0, y0, x1, y1, fill, width):
    idr = ImageDraw.Draw(image)
    idr.line((x0,y0,x1,y1),fill,width)
    return image

def check_coords(image, x0, y0, x1, y1):
    width, height = image.size
    if x0 < 0 or y0 < 0 or x1 < 0 or y1 < 0:
        return False
    if x0 >= width or x1 >= width or y0 >= height or y1 >= height:
        return False
    if x1 <= x0 or y1 <= y0:
        return False
    return True

#2

def set_black_white(image,x0,y0,x1,y1):
    if check_coords(image,x0,y0,x1,y1)==True:
        img_one = image.crop((x0,y0,x1,y1))
        g_img = img_one.convert("1")
        image.paste(g_img, (x0,y0))
    return image


def check_coords(image, x0, y0, x1, y1):
    x,y=image.size
    return (((y1-y0)>=0 and (y1-y0)<=y) and (((x1-x0)>=0 and (x1-
x0)<=x)) and  (x0>=0 and x1>=0) and (y0>=0 and y1>=0))


#3

def helperpr(x, y, width, height, datapx, old_color):
    top = [0, 0]
    bottom = [width, height]
    curr = [(x, y)]

    while len(curr)>0:
        x1, y1 = curr.pop()
        if (0 <= x1 < width and 0 <= y1 < height and datapx[x1, y1] ==
old_color):
            bottom = [min(bottom[0], x1), min(bottom[1], y1)]
            top = [max(top[0], x1), max(top[1], y1)]
            curr+=[(x1-1,y1),(x1+1,y1),(x1,y1-1),(x1,y1+1)]
            datapx[x1, y1] = (0, 0, 0)
```

```
    return (bottom[0], bottom[1], top[0], top[1])


def find_rect_and_recolor(image, old_color, new_color):
    max = 0
    cimg=image.copy()
    datapx = cimg.load()
    height, width = image.size[1], image.size[0]
    for x in range(width):
        for y in range(height):
            if datapx[x, y] == old_color:
                crds = helperpr(x, y, width, height, datapx,
old_color)
                s_rect = (crds[3]+1 - crds[1]) * (crds[2]+1 - crds[0])
                if s_rect > max:
                    max = s_rect
                    crds_max = crds

    image_r = image.load()
    for x in range(crds_max[0], crds_max[2]+1):
        for y in range(crds_max[1], crds_max[3]+1):
            image_r[x, y] = new_color

    return image
```