

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

Кузнецова С.Е.

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, 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`.

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:

Объявляем функцию `def user_func(image, x0, y0, x1, y1, fill, width)`. Она принимает на вход изображение, координаты начала и конца отрезка, цвет отрезка и его толщину. С помощью метода `Draw` рисуем линию с использованием всех этих параметров и возвращаем полученное измененное изображение.

2 задача:

Объявляем функцию `check_coords(image, x0, y0, x1, y1)`, которая проверяет корректность координат, с которыми мы будем работать. Если координаты больше или равны нулю и разность координат по каждой оси не превышает размер изображения по этой оси, то значение `k` равно единице. Функция выводит результат выполнения условия `k==1`.

Объявляем функцию `set_black_white(image, x0, y0, x1, y1)`. Если координаты корректны (функция `check_coords` вернула `True`), то создаем новое изображение, состоящее из нужной обрезанной части введенного изображения от левого верхнего до правого нижнего угла, переводим это изображение в ч/б с помощью функции `convert` со значением “1” и вставляем его на место вырезанной части. Функция возвращает полученное изображение.

3 задача:

Объявляем функцию `find_rect_and_recolor(image, old_color, new_color)`. Создаем копию изображения для работы с ним и пользуемся функцией `load()` для перевода в пиксели. Определяем длину и ширину изображения с помощью функции `size`, максимальная площадь изначально равна нулю. Пробегаемся по пикселям копии и если находим пиксель старого цвета, в переменную `coords` записываем кортеж из координат левого верхнего и правого нижнего угла прямоугольника, найденных с помощью вспомогательной функции `fill`. Далее определяем площадь найденного прямоугольника и если она оказывается больше площади максимальной — запоминаем координаты углов прямоугольника и максимальную площадь приравниваем к найденной. Далее

пробегаемся по пикселям наибольшего прямоугольника в заданном изображении и меняем их цвет на новый. Функция возвращает измененное изображение.

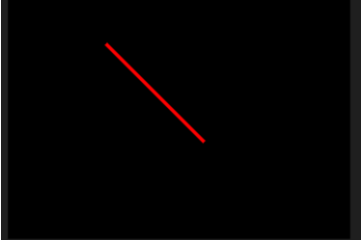
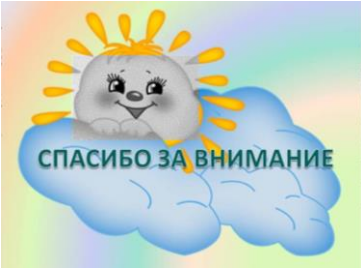
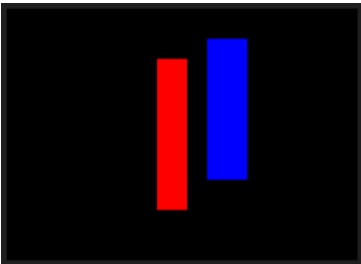
Объявляем функцию `fill(x, y, width, height, pixdata, old_color)`. Первоначальные значения координат левого верхнего угла ставим равными (0,0), нижнего правого – (длина изображения, высота изображения), в массив с текущими проверяемыми пикселями записываем переданные значения координат (x,y). Далее, пока массив не пустой, берем из него первые координаты, удаляем их и работаем с ними дальше: Если они находятся в пределах изображения и цвет данного пикселя равен старому цвету, тогда в массив с координатами левого верхнего угла записываются минимальные найденные координаты, с координатами правого нижнего – максимальные. Далее в массив с текущими проверяемыми координатами добавляем все смежные координаты для следующей проверки и цвет проверенного пикселя делаем черным, чтобы избежать множественной проверки. Функция выводит кортеж с координатами левого верхнего угла и правого нижнего.

См. приложение А.

Тестирование

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

п/п	Входные данные	Выходные данные	Комментарии
	<code>user_func(Image.new("RGB", (350, 250), 'black'), 100, 50, 200, 150, 'red', 3)</code>		№1
	<code>set_black_white(Image.open("input.png"), 250, 200, 700, 500)</code>		№2
	<code>image = Image.new("RGB", (350, 250), 'black') image.paste(Image.new("RGB", (30, 150), 'red'), (150, 50)) image.paste(Image.new("RGB", (40, 140), 'red'), (200, 30)) find_rect_and_recolor(image, (255, 0, 0), (0, 0, 255))</code>		№3

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_lb2.py

```
# Задача 1
def user_func(image, x0, y0, x1, y1, fill, width):
    drawing = ImageDraw.Draw(image)
    drawing.line(((x0,y0),(x1,y1)), fill, width)
    return image

# Задача 2
def check_coords(image, x0, y0, x1, y1):
    x,y=image.size
    k=0
    if (x0>=0 and x1>=0 and y0>=0 and y1>=0):
        if ((x1-x0)>=0 and (x1-x0)<=x):
            if ((y1-y0)>=0 and (y1-y0)<=y):
                k=1
    return k==1

def set_black_white(image,x0,y0,x1,y1):
    if check_coords(image,x0,y0,x1,y1)==True:
        img1 = image.crop((x0,y0,x1,y1))
        gray_img = img1.convert("1")
        image.paste(gray_img, (x0,y0))
    return image

# Задача 3
def find_rect_and_recolor(image, old_color, new_color):
    img_copy=image.copy()
    pixdata = img_copy.load()
    width, height = image.size[0], image.size[1]
    s_max = 0

    for x in range(width):
        for y in range(height):
            if pixdata[x, y] == old_color:
                coords = fill(x, y, width, height, pixdata, old_color)
                s_rect = (coords[3]+1 - coords[1]) * (coords[2]+1 -
coords[0])
                if s_rect > s_max:
                    s_max = s_rect
                    coords_max = coords

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

    return image

def fill(x, y, width, height, pixdata, 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 pixdata[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)]
        pixdata[x1, y1] = (0, 0, 0)

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