

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ
по лабораторной работе №2
по дисциплине «Информатика»
Тема: Введение в архитектуру компьютера

Студент гр. 3341

Костромитин М.М.

Преподаватель

Иванов Д.В.

Санкт-Петербург

2023

Цель работы

Целью лабораторной работы является изучение модуля PIL языка python и его практическое применение для решения трех подзадач лабораторной работы.

Задание

Вариант 2

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 круге.

Функция `pentagram()` принимает на вход:

- Изображение (`img`)
- координаты левого верхнего и правого нижнего угла квадрата, в который вписана окружность (`x0,y0,x1,y1`)
- Толщину линий и окружности (`thickness`)
- Цвет линий и окружности (`color`) - представляет собой список (`list`) из 3-х целых чисел

Функция должна вернуть обработанное изображение.

Примечание:

Вершины пентаграммы высчитывать по формуле:

$$\begin{aligned} \text{phi} &= (\pi/5) * (2*i + 3/2) \\ \text{node_i} &= (\text{int}(x_0 + r * \cos(\text{phi})), \text{int}(y_0 + r * \sin(\text{phi}))) \end{aligned}$$

`x0,y0` - координаты центра окружности, в который вписана пентаграмма

`r` - радиус окружности

`i` - номер вершины от 0 до 4

Подсказка: Округляйте все вычисляемые вами значения (кроме значений углов) до целых чисел.

2) Инвертирование полос

Необходимо реализовать функцию `invert`, которая делит изображение на "полосы" и инвертирует цвет нечетных полос (счёт с нуля).

Функция `invert()` принимает на вход:

- Изображение (`img`)
- Ширину полос в пикселах (`N`)
- Признак того, вертикальные или горизонтальные полосы (`vertical` - если `True`, то вертикальные)

Функция должна разделить изображение на вертикальные или горизонтальные полосы шириной `N` пикселей. И инвертировать цвет в нечетных полосах (счет с нуля). Последняя полоса может быть меньшей ширины, чем `N`.

3) Поменять местами 9 частей изображения

Необходимо реализовать функцию `mix`, которая делит квадратное изображение на 9 равных частей (сторона изображения делится на 3), и по правилам, записанным в словаре, меняет их местами.

Функция `mix()` принимает на вход:

- Изображение (`img`)
- Словарь с описанием того, какие части на какие менять (`rules`)

Пример словаря `rules`:

`{0:1,1:2,2:4,3:4,4:5,5:3,6:8,7:8,8:8}`

Элементы нумеруются слева-направо, сверху-вниз.

В данном случае нулевой элемент заменяется на первый, первый на второй, второй на четвертый, третий на четвертый и так далее.

Функция должна вернуть обработанное изображение.

Пример входной картинки и словаря:



{0:2,1:2,2:2,3:5,4:5,5:5,6:8,7:8,8:8}

Результат:



Можно реализовывать дополнительные функции.

Выполнение работы

Импортируем Image, ImageDraw, ImageOps из модуля PIL, импортируем модель pipru.

Задача 1.

Создадим переменную drawing с помощью которой будем добавлять объекты на картинку, нарисуем круг внутри которого будет находится пентаграмма, создадим словарь с координатами центра нарисованного круга а также найдем радиус данного круга, далее в цикле будем искать координаты каждой точки нашей пентаграммы и записывать их в массив кортежей nodes, и в конце с помощью метода line нарисуем пентаграмму.

Задача 2.

Укажем переменную M, которая будем равняться количеству полос, на которые будет разбита изначальная картинка, далее будем проверять являются ли наши полосы горизонтальными или вертикальными, после этого проверяем делится ли соответствующий размер картинки нацело на ширину полосы, если да то каждую нечетную полосу вырезаем с помощью метода crop и с помощью класса ImageOps инвертируем и добавляем на исходную картинку. Если же соответствующий размер не делится нацело на ширину полосы, то делаем все тоже самое для всех полос кроме последней, а далее последнюю полосу проверяем отдельно.

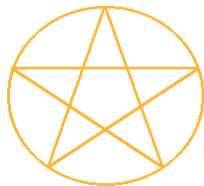
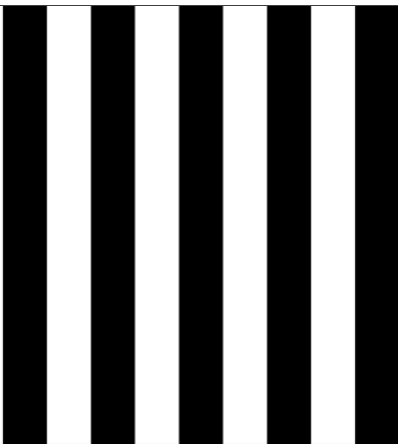
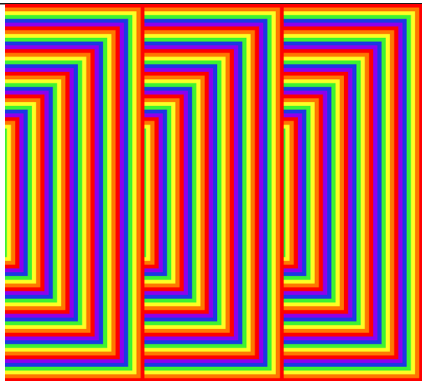
Задача 3.

Сначала в цикле разобьем нашу картинку на 9 частей и добавим их всех в соответствующий по счету элемент массива pieces а также их координаты левого верхнего угла. Далее в цикле проходимся по ключам словаря rules и вставляем на место ключа, значение этого ключа на исходную картинку.

Тестирование

Результаты тестирования представлены в Таблице 1.

Таблица 1.

№ п/п	Входные данные	Выходные данные	Комментарии
1	<code>pentagram(Image.new('RGB', (450, 450), 'white'), 50, 50, 250, 250, 3, (250, 184, 59))</code>		Задача 1
2	<code>invert(Image.new('RGB', (300, 300), 'white'), 30, True)</code>		Задача 2
3	<code>mix(Image.open('example.jpg'), {0:2, 1:2, 3:5, 4:5, 6:8, 7:8})</code>		Задача 3

Выводы

В результате выполнения лабораторной работы были изучены и использованный на практике функции библиотеки PIL языка python, написанна программа, выполняющая три подзадачи.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Исходный файл: main.py

```
from PIL import Image, ImageDraw, ImageOps
import numpy as np
from math import cos, sin, pi

def pentagram(img, x0, y0, x1, y1, thickness, color):
    color = tuple(color)

    drawing = ImageDraw.Draw(img)
    drawing.ellipse((x0, y0, x1, y1), outline=color,
width=thickness)

    centre = {'x': x1 - (abs(x1 - x0) // 2), 'y': y1 - (abs(y1 -
y0) // 2)}
    radius = abs(x1 - x0) // 2

    nodes = [(0, 0)] * 5

    for i in range(5):
        phi = (np.pi / 5) * (2 * i + 3 / 2)
        node_i = (int(centre['x'] + radius * np.cos(phi)),
int(centre['y'] + radius * np.sin(phi)))
        nodes[i] = node_i

    drawing.line((nodes[0], nodes[3], nodes[1], nodes[4], nodes[2],
nodes[0]), color, thickness, None)
    return img
    pass

def invert(img, N, vertical):
    w = img.size[0]
    h = img.size[1]
    M = (w // N) + 1

    if vertical:
        if w % N == 0:
            for i in range(M):
                if i % 2 == 1:
                    strip = img.crop((N * i, 0, N * (i + 1), h))
                    invertedStrip = ImageOps.invert(strip)
                    img.paste(invertedStrip, (N * i, 0))
        else:
            for i in range(M):
                if i % 2 == 1 and i < (M - 1):
                    strip = img.crop((N * i, 0, N * (i + 1), h))
                    invertedStrip = ImageOps.invert(strip)
                    img.paste(invertedStrip, (N * i, 0))
                if i == M - 1 and i % 2 == 1:
                    strip = img.crop((N * (M - 1), 0, w, h))
                    invertedStrip = ImageOps.invert(strip)
                    img.paste(invertedStrip, (N * (M - 1), 0))
```

```

else:
    if w % N == 0:
        for i in range(M):
            if i % 2 == 1:
                strip = img.crop((0, N * i, w, N * (i + 1)))
                invertedStrip = ImageOps.invert(strip)
                img.paste(invertedStrip, (0, N * i))
    else:
        for i in range(M):
            if i % 2 == 1 and i < (M - 1):
                strip = img.crop((0, N * i, w, N * (i + 1)))
                invertedStrip = ImageOps.invert(strip)
                img.paste(invertedStrip, (0, N * i))
            if i == M - 1 and i % 2 == 1:
                strip = img.crop((0, N * (M - 1), w, h))
                invertedStrip = ImageOps.invert(strip)
                img.paste(invertedStrip, (0, N * (M - 1)))

return img
pass

def mix(img, rules):
    pieces = []
    w = img.size[0]
    h = img.size[1]

    for i in range(3):
        for j in range(3):
            piece = img.crop((j * (h // 3), i * (w // 3), (j + 1) *
(h // 3), (i + 1) * (w // 3)))
            pieces += [(piece, (j * (h // 3), i * (w // 3)))]

    for i in rules:
        img.paste(pieces[rules[i]][0], pieces[i][1])

return img
pass

```