**Лабораторная работа №4**

Содержание

[Урок 25. Задачи: 3](#_Toc21809711)

[25.1. Генератор визуально различимых паролей (A) 3](#_Toc21809712)

[25.2. Генератор визуально различимых паролей (B) 4](#_Toc21809713)

[25.3. Найти приближённое значение Пи 5](#_Toc21809714)

[25.4. Генератор визуально различимых паролей (A + B) 7](#_Toc21809715)

[Урок 26. Задачи: 8](#_Toc21809716)

[26.1. Градиент 8](#_Toc21809717)

[26.2. Шахматная доска 10](#_Toc21809718)

[26.3. Стереопара 12](#_Toc21809719)

[26.4. Красивое имя 15](#_Toc21809720)

[Урок 27. Задачи: 16](#_Toc21809721)

[27.1. Миниатюра для сайта 16](#_Toc21809722)

[27.2. «Минифотошоп» 18](#_Toc21809723)

**Урок 25. Задачи:**

# 25.1. Генератор визуально различимых паролей (A)

Требуется по запросу выдавать N различных паролей длиной M символов, состоящих из строчных и прописных латинских букв и цифр, кроме тех, которые легко перепутать между собой: «l» (L маленькое), «I» (i большое), «1» (цифра), «o» и «O» (большая и маленькая буквы) и «0» (цифра).   
  
Дополнительное условие: в каждом из паролей не должно быть повторяющихся символов.   
  
Решение должно содержать две функции: вспомогательную generate\_password(m), возвращающую случайный пароль длиной m символов, и основную main(n, m), возвращающую список из n различных паролей, каждый длиной m символов.   
  
Будем считать, что параметры n и m всегда таковы, что требуемые пароли возможно сгенерировать.

Формат ввода

Пример вызова ваших функций для тестирования:

print("Случайный пароль из 7 символов:", generate\_password(7))    
print("10 случайных паролей длиной 15 символов:")    
print(\*main(10, 15), sep="\n")

Формат вывода

Пример результатов:

Случайный пароль из 7 символов: ZdmtPhz    
10 случайных паролей длиной 15 символов:    
H2YuebGj9mwgXUp    
9P4WRMeu8GmpZa3    
Bpje6wgvC72kxJn    
7U5SbRMrkgYNGuT    
GMBWavKAdHi5jr7    
CXTYJNwcdazDtLP    
g9MjduQWhivCwVH    
H9vats8jF6A3xr4    
mcvxtXNujkKiz68    
mTCQ4Wi5yqPBtgv

Примечания

Возможно, вам пригодятся константы string.ascii\_letters и string.digits из [модуля string](https://docs.python.org/3/library/string.html).

# 25.2. Генератор визуально различимых паролей (B)

Требуется по запросу выдавать N различных паролей длиной M символов, состоящих из строчных и прописных латинских букв и цифр, кроме тех, которые легко перепутать между собой: «l» (L маленькое), «I» (i большое), «1» (цифра), «o» и «O» (большая и маленькая буквы) и «0» (цифра).   
  
Дополнительное условие: в каждом пароле обязательно должна присутствовать хотя бы одна цифра и как минимум по одной букве в верхнем и нижнем регистре.   
  
Решение должно содержать две функции: вспомогательную generate\_password(m), возвращающую случайный пароль длиной m символов, и основную main(n, m), возвращающую список из n различных паролей, каждый длиной m символов.   
  
Будем считать, что параметры n и m всегда таковы, что требуемые пароли возможно сгенерировать.

Формат ввода

Пример вызова ваших функций для тестирования:

print("Случайный пароль из 7 символов:" , generate\_password(7))    
print("10 случайных паролей длиной 15 символов:")    
print(\*main(10, 15), sep="\n")

Формат вывода

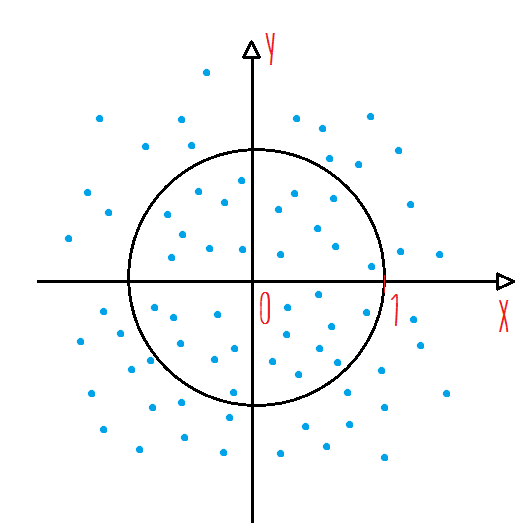
Пример результатов:

Случайный пароль из 7 символов: V77LMgk    
10 случайных паролей длиной 15 символов:    
nsHVGJH4B9q3n7B    
KgBmJKP9XLqHKcp    
LuydsM9ZgBUJvVj    
a3jxMpQjmtjs79g    
nvU8SMLfGFq4erU    
sfcmQmR6dLKQeNt    
WNtFQd6Bd7wADNP    
fDtMmhxA9eN4yJg    
Uau2sRtWL5qH81R    
G5hfFkHKxSBZxsu

Примечания

Возможно, вам пригодятся константы string.ascii\_uppercase, string.ascii\_lowercase и string.digits из [модуля string](https://docs.python.org/3/library/string.html).

# 25.3. Найти приближённое значение Пи



При помощи метода Монте-Карло определите приближённое значение числа π.   
Метод Монте-Карло — общее название группы численных методов, основанных на получении большого числа реализаций стохастического (случайного) процесса, который формируется таким образом, чтобы его вероятностные характеристики совпадали с аналогичными величинами решаемой задачи.   
  
Например, если мы будем случайным образом «бросать» на плоскость точки (случайно выбирать координаты x и y), то чем больше площадь фигуры, тем больше на нее попадёт точек. Если при этом число «бросков» стремится к бесконечности, то отношение количества точек, попавших на две фигуры, будет стремиться к отношению площадей этих фигур.   
  
Из курса математики мы знаем, что площадь круга равна πR2. Значит, чтобы вычислить число π, достаточно узнать площадь круга радиусом 1. Для простоты вычисления возьмём четверть такого круга, лежащую в I квадранте Декартовой плоскости, а потом умножим результат на 4. Таким образом, координаты x и у будут меняться в диапазоне [0, 1].

# 25.4. Генератор визуально различимых паролей (A + B)

Требуется по запросу выдавать N различных паролей длиной M символов, состоящих из строчных и прописных латинских букв и цифр, кроме тех, которые легко перепутать между собой: «l» (L маленькое), «I» (i большое), «1» (цифра), «o» и «O» (большая и маленькая буквы) и «0» (цифра).   
  
Дополнительное условие: в каждом пароле обязательно должна присутствовать хотя бы одна цифра и как минимум по одной букве в верхнем и нижнем регистре, причём все символы должны быть различны.   
  
Решение должно содержать две функции: вспомогательную generate\_password(m), возвращающую случайный пароль длиной m символов, и основную main(n, m), возвращающую список из n различных паролей, каждый длиной m символов.   
  
Будем считать, что параметры n и m всегда таковы, что требуемые пароли возможно сгенерировать.

Формат ввода

Пример вызова ваших функций для тестирования:

print("Случайный пароль из 7 символов:" , generate\_password(7))    
print("10 случайных паролей длиной 15 символов:")    
print(\*main(10, 15), sep="\n")

Формат вывода

Пример результатов:

Случайный пароль из 7 символов: A6fXxBh    
10 случайных паролей длиной 15 символов:    
JebgLW4Ndzkt6rU    
Q3nbTS9ERguDy2e    
E98heZH32fFGVTN    
hJZ5bvCrny2RFEA    
rumkV5WzSHKRA9q    
3Yd7exfi6HBZ98y    
aUWZgjDutswnHv7    
vU2x7wNQHps4nmi    
a8zn23vNGipJqk7    
nkQ5hCetWJ3jHRF

Примечания

Возможно, вам пригодятся константы string.ascii\_uppercase, string.ascii\_lowercase и string.digits из [модуля string](https://docs.python.org/3/library/string.html).

**Урок 26. Задачи:**

# 26.1. Градиент

Напишите функцию **gradient(color)**, создающую графический файл с плавным переходом цвета. Файл должен содержать прямоугольник длиной 512 пикселей (по 2 пикселя на каждый оттенок) и высотой 200 пикселей.

В функцию передается обозначение цвета, градиент которого надо построить. Варианты цветов: R, G и B. Обозначение может быть введено в любом регистре.

Формат ввода

gradient(’R’)

Формат вывода

Файл **res.png**:



Примечания

Для того чтобы создать новое изображение необходимо воспользоваться функцией **Image.new()**:

new\_color = (0, 0, 0)    
new\_image = Image.new("RGB", (size\_x, size\_y), new\_color)

Для рисования на изображении используется объект **Draw**из библиотеки **ImageDraw**. У этого объекта есть много инструментов для создания **графических примитивов**: прямых, кривых, точек, прямоугольников, дуг и т.д.

Следующий пример создает новое черное изображение размером 100 на 200 и нарисует на нем линию красного цвета, толщиной в 1 пиксель из левого верхнего в правый нижний угол.

# создание изображения    
new\_image = Image.new("RGB", (100, 200), (0, 0, 0))    
# на изображении создаем рисунок для рисования    
draw = ImageDraw.Draw(new\_image)    
# рисуем линию    
draw.line((0, 0, 100, 200), fill=(255, 0, 0), width=1)    
# сохраним изображением в файл формата PNG    
new\_image.save(’line.png’, "PNG")

Подробно про библиотеку **ImageDraw**можно почитать тут: <https://pillow.readthedocs.io/en/stable/reference/ImageDraw.html#functions>

# 26.2. Шахматная доска

Напишите функцию **board(num, size)**, создающую графический файл в формате **PNG**с изображением квадратного чёрно-белого клетчатого поля.

В функцию поступают два целых числа — количество клеток **n**и размер клетки в пикселях **s**.

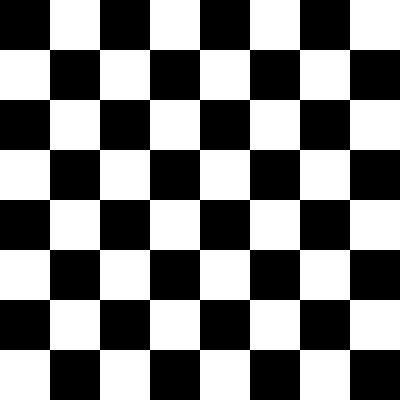
(в примере создаётся доска 8x8 клеток, размер каждой клетки — 50x50 пикселей).

Левая верхняя клетка должна быть чёрной.

Формат ввода

board(8, 50)

Формат вывода

Графический файл **res.png**:   


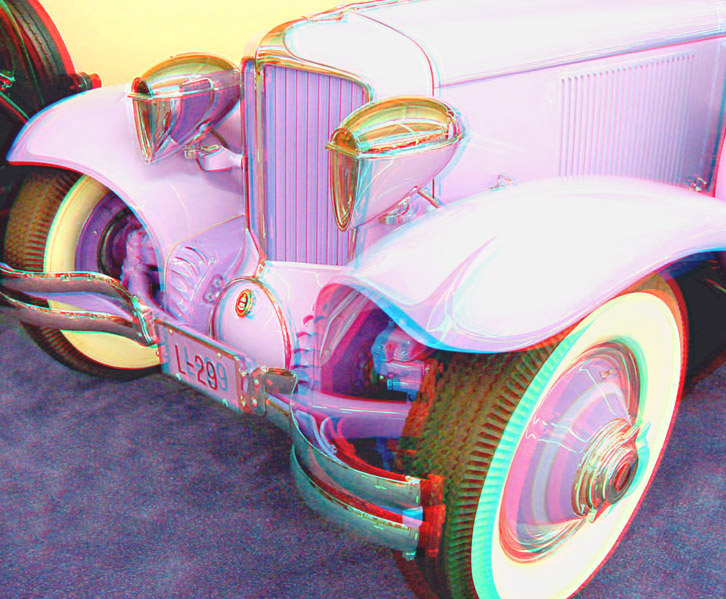
Примечания

Чтобы создать новое изображение, необходимо воспользоваться функцией **Image.new()**:

new\_color = (0, 0, 0)    
newImage = Image.new("RGB", (size\_x, size\_y), new\_color)

Для рисования прямоугольников воспользуйтесь функцией **rectangle()** из **ImageDraw**. Напомним, что подробная информация тут: <https://pillow.readthedocs.io/en/stable/reference/ImageDraw.html#functions>

# 26.3. Стереопара

  
  
Стереопара — это пара плоских изображений одного и того же объекта, в которой различия между изображениями создают эффект объёма.   
  
Один из самых простых способов получить из стереопары трёхмерное изображение — анаглиф. Для просмотра такого изображения нужны очки с двумя линзами разных цветов — например, с синей и красной.   
  
Чтобы построить итоговое изображение, нужно брать канал R из одной копии изображения (левый глаз), а каналы B и G — из другой (правый глаз).   
  
Напишите функцию **makeanagliph(ﬁlename, delta)**, которая получает в себя имя файла с картинкой и расстояние, на которое надо сдвинуть изображение c красной составляющей вправо, чтобы построить анаглиф.

Кусочек картинки, который «вылезает» вправо сохранять не надо, таким образом, размеры исходной и получаемой картинки должны совпадать.

Результатом работы является файл **res.jpg**. Функция возвращает None.

Формат ввода

Тестирующая система вызовет вашу функцию, поэтому ничего с клавиатуры вводить не надо.   
Пример вызова:

 makeanagliph("image.jpg", 10)



Формат вывода

Результат сохраните в файл **res.jpg**в той же папке, где лежит ваша программа.



Примечания

Для получения копии изображения вам может помочь функция Image.copy(), а чтобы создать новое изображение необходимо воспользоваться функцией Image.new():

 newImage = Image.new(’RGB’, (x, y), (0,0,0))

А может, вы придумаете другой способ.

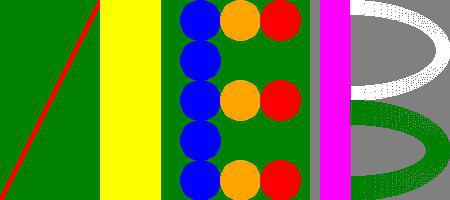
# 26.4. Красивое имя

Напишите программу, которая **красиво** рисует имя своего автора. Каждую букву надо нарисовать с помощью графических примитивов (линий, дуг и прочее), для заливки так же необходимо использовать функции библиотеки PIL.

Главное правило – **не использовать встроенные шрифты**.

Сохраните полученный рисунок в файле **name.png**.

Например, для имени Лев «красивое имя» может выглядеть так:



**Урок 27. Задачи:**

# 27.1. Миниатюра для сайта

В файле **image.jpg**лежит изображение. Почитайте документацию на модуль **Image**библиотеки **PIL**<https://pillow.readthedocs.io/en/3.1.x/index.html> и напишите функцию **make\_preview(size, n\_colors)**которая:

1. открывает изображение **image.jpg**
2. уменьшает его до размера **size**
3. уменьшает число цветов в изображении до **n\_colors**(такая процедура называется квантование или **quantize**)
4. сохраняет результат в файле **res.bmp**

Формат ввода

Изображение **image.jpg**в текущей папке.



вызов функции:

make\_preview((400, 200), 64)

Формат вывода



Примечания

Обратите внимание, что результат сохраняется в формате **bmp**.

# 27.2. «Минифотошоп»

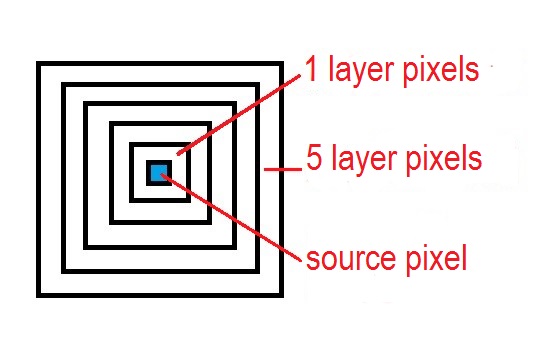
Сейчас существует огромное количество приложений для обработки фотографий, многие из них могут похвастаться интересными и довольно сложными фильтрами, в результате применения которых изображение может измениться до неузнаваемости.

Подавляющее большинство сложных фильтров для преобразования используют информацию не только о цвете пикселя, к которому применяется преобразование, но и о его «соседях». Зачастую в преобразованиях используется не один ряд соседних пикселей.

В общем случае сигнатуру функции, которая реализовывает такое преобразование, можно описать следующим образом:

def image\_filter(pixels, i, j):    
    ...    
    return r, g, b

То есть функция принимает на вход некоторый массив пикселей, а также координаты того пикселя, к которому применяется преобразование (потому что он не всегда будет в центре – например, при обработке краёв изображения). В результате преобразования функция возвращает кортеж цветов целевого пикселя после преобразования. При преобразовании можно (но не обязательно) учитывать соседние пиксели.



Например, вот такой фильтр ничего не делает (изображение остаётся прежним):

def image\_filter(pixels, i, j):    
    return pixels[i][j]

А вот пример фильтра, который заменяет цвет исходного пикселя на средний относительно его окружения:

def image\_filter(pixels, i, j):    
    """Заменяет цвет исходного пикселя на средний относительно его окружения"""    
    r, g, b = 0, 0, 0    
    for row in range(11):    
        for col in range(11):    
            if row != i or col != j:    
                r += pixels[row, col][0]    
                g += pixels[row, col][1]    
                b += pixels[row, col][2]    
    return int(r / 120), int(g / 120), int(b / 120)

Напишите свою функцию **image\_ﬁlter(pixels, i, j)**(не обязательно с таким названием, главное – чтобы с такой сигнатурой), в которой реализован какой-нибудь фильтр для изображения. Она может, например, увеличивать контраст, делать сглаживание и т. д.

Напишите к своей функции краткое описание в **docstring**как в примере выше. Посмотрите, как теперь будет выглядеть результат функции **help()**для вашей функции.

Если у вас будет несколько фильтров, которые придерживаются такой же сигнатуры, из них можно будет собирать целые конвееры для получения сложных эффектов.

Примечания

Если идея для фильтра не придумывается, ее можно «подсмотреть» в Instagram или в любом другом приложении для обработки фото. Кроме того, можете почитать документацию на модуль **ImageFilter**библиотеки **PIL**и поискать вдохновения там.