


ХОЧУ ПОМОЧЬ  
ПРОЕКТУ

Модуль qrcode в Python, генератор QR кодов



gb.ru

РЕКЛАМА • 16+

Бесплатный практикум для детей: Python и анимация

Живой практикум для детей по 2D-анимации и Python. Количество мест ограничено.

Узнать больше

Страница 3. / Модуль qrcode в Python, генератор QR кодов

Советы

Советы

ВСТРЕЧАЙТЕ НОВЫЕ АВТОМОБИЛИ SOLLERS

Многообразие модификаций

Оставить заявку

Python

виртуальное окружение;

п;

Создание QR-кода в SVG-формате;

Создание QR-кода в текстовом виде;

Стилизация изображения с QR-кодом;

Использование модуля qrcode из командной строки.

## Что такое QR-код?

QR-код - это двумерный пиктографический код, который стал широко использоваться из-за его быстрой читаемости и сравнительно большой емкости памяти. Код состоит из черных модулей, расположенных в виде квадрата на белом фоне. Закодированная информация может состоять из данных любого типа (например, двоичных, буквенно-цифровых символов или символов Кандзи).

## Установка модуля qrcode в виртуальное окружение.

```
# создаем виртуальное окружение, если нет
$ python3 -m venv .venv --prompt VirtualEnv
# активируем виртуальное окружение
$ source .venv/bin/activate
# ставим модуль qrcode
(VirtualEnv):~$ python3 -m pip install -U qrcode
```

## Создание QR-кода на Python.

Простой пример [создания QR-кода на Python](#), средствами стороннего модуля qrcode.

```
import qrcode

img = qrcode.make('https://docs-python.ru/packages/generator-qr-kodov/')
type(img) # qrcode.image.pil.PilImage
img.save("some_file.png")
```

Для большего контроля над генерацией QR-кода, можно использовать класс qrcode.QRCode().

Пример использования класса qrcode.QRCode():

```
import qrcode

qr = qrcode.QRCode(
    version=1,
    error_correction=qrcode.constants.ERROR_CORRECT_L,
    box_size=10,
    border=4,
)

qr.add_data('Какой-то текст или URL-адрес')
qr.make_image(fit=True)
```


https://docs-python.ru/packages/generator-qr-kodov/


1/4

```
img = qr.make_image(fill_color="black", back_color="white")
img.save("some_file.png")
```

Метод `QRCode.add_data()` добавит данные к текущему объекту `qr`. Чтобы добавить новые данные путем замены предыдущего содержимого в том же объекте, сначала используйте метод `QRCode.clear()`:

РЕКЛАМА





**ВСТРЕЧАЙТЕ НОВЫЕ  
АВТОМОБИЛИ SOLLERS**

Многообразие модификаций

Оставить заявку

```
qr.add_data('или URL-адрес')
qr.add_data('URL-адрес')
qr.save('some_file.png')
```

Аргумент `version` — это число от 1 до 40, которое контролирует размер QR-кода (самый маленький `version=1` и самый большой `version=40` (21). Чтобы размер определялся автоматически, нужно установить `version=None` и использовать метод `qr.make_image()`.  
Аргументы `fill_color` и `back_color` могут изменять фон и цвет отрисовки QR-кода при использовании фабрики изображений по умолчанию. Оба параметра принимают кортежи цветов RGB.

```
img = qr.make_image(back_color=(255, 195, 235), fill_color=(55, 95, 35))
```

Аргумент `error_correction` управляет исправлением ошибок, используемым для QR-кода. В [модуле qrcode](#) доступны следующие четыре константы:

- `ERROR_CORRECT_L`: можно исправить около 7% или меньше ошибок.
- `ERROR_CORRECT_M`: можно исправить около 15% или меньше ошибок.
- `ERROR_CORRECT_Q`: можно исправить около 25% или меньше ошибок.
- `ERROR_CORRECT_H`: можно исправить около 30% или меньше ошибок.

Аргумент `box_size` определяет количество пикселей в каждом "квдрате" QR-кода. Другими словами - это разрешение картинки.

Аргумент `border` определяет, сколько прямоугольников должно быть у границы (по умолчанию - 4, что является минимумом в соответствии со спецификациями).

## Создание QR-кода в SVG-формате.

Можно создать SVG целиком или фрагмент SVG. При построении всего изображения SVG, можно использовать фабрику, которая объединяет как путь (рекомендуется и по умолчанию для скрипта), или фабрику, которая создает простой набор прямоугольников.

Смотрим пример:

```
import qrcode
import qrcode.image.svg

if method == 'basic':
    # Простая фабрика, просто набор прямоугольников.
    factory = qrcode.image.svg.SvgImage
elif method == 'fragment':
    # Фабрика фрагментов (тоже просто набор прямоугольников)
    factory = qrcode.image.svg.SvgFragmentImage
else:
    # Фабрика комбинированных путей, исправляет пробелы,
    # которые могут возникнуть при масштабировании
    factory = qrcode.image.svg.SvgPathImage

img = qrcode.make('Some data here', image_factory=factory)
```

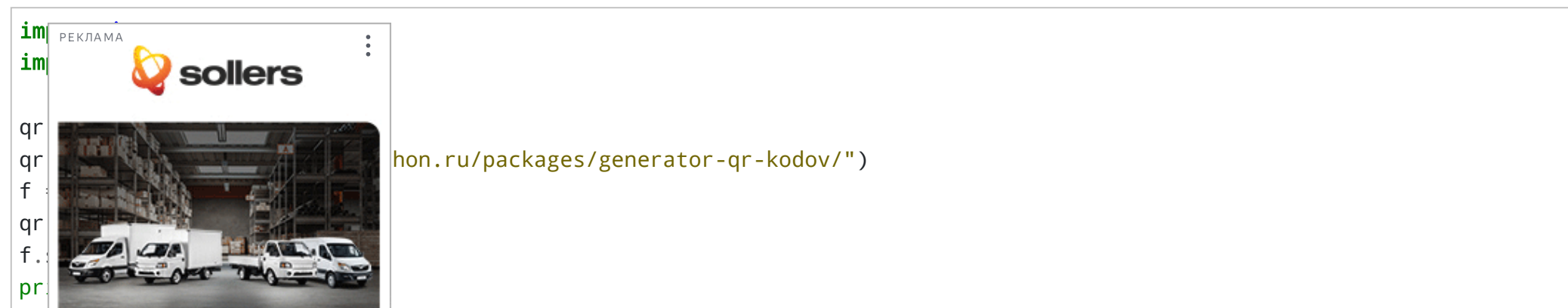
Доступны еще две связанные фабрики, которые работают так же, при этом заполняют фон SVG белым:

- `qrcode.image.svg.SvgFillImage;`
- `qrcode.image.svg.SvgPathFillImage;`

Вверх

## Создание QR-кода в текстовом виде.

Получаем текстовое представление методом `QRCode.print_ascii`:



Страница 1 из 1

Чтобы получить изображение QR-кода, нужно использовать фабрику изображений `StyledPilImage`. Дополнительно, для изменения модули для управления формой QR-кода, дополнительная цветовая маска для изменения цвета изображения для встраивания в центр.

Эти функции позволяют работать со всеми читалками QR, поэтому поэкспериментируйте и установите высокий уровень коррекции ошибок (особенно при встраивании изображения).

Примеры рисования QR-кода с закругленными углами, радиальным градиентом и встроенным изображением:

```
import qrcode
from qrcode.image.styledpil import StyledPilImage
from qrcode.image.styles.moduledrawers import RoundedModuleDrawer
from qrcode.image.styles.colormasks import RadialGradientColorMask

qr = qrcode.QRCode(error_correction=qrcode.constants.ERROR_CORRECT_L)
qr.add_data('Какие-то данные или URL-адрес')

# закругленные углы
img_1 = qr.make_image(image_factory=StyledPilImage, module_drawer=RoundedModuleDrawer())
# радиальный градиент
img_2 = qr.make_image(image_factory=StyledPilImage, color_mask=RadialGradientColorMask())
# встроенное изображение `path/to/image.png`
img_3 = qr.make_image(image_factory=StyledPilImage, embedded_image_path="/path/to/image.png")
```

Список доступных классов для использования с аргументом `module_drawer`:

У всех ниже представленных классов (за исключением ImageColorMask()), аргументы означают цвет и принимают кортеж RGB, например back\_color=(255,255,255).

- `SolidFillColorMask(back_color, front_color)`: Просто заливает фон одним цветом, а передний план - другим.
- `RadialGradientColorMask(back_color, center_color, edge_color)`: Заполняет передний план радиальным градиентом от центра к краю.
- `SquareGradientColorMask(back_color, center_color, edge_color)`: Заполняет передний план квадратным градиентом от центра к краю.
- `HorizontalGradientColorMask(back_color, left_color, right_color)`: Заполняет передний план плавным переходом слева направо.
- `VerticalGradientColorMask(back_color, top_color, bottom_color)`: Заполняет передний план плавным переходом сверху вниз.
- `ImageColorMask(back_color, color_mask_path, color_mask_image)`: Заполняет передний план пикселями с другого изображения, переданными либо по пути `color_mask_path`, либо по объекту изображения `color_mask_image`.

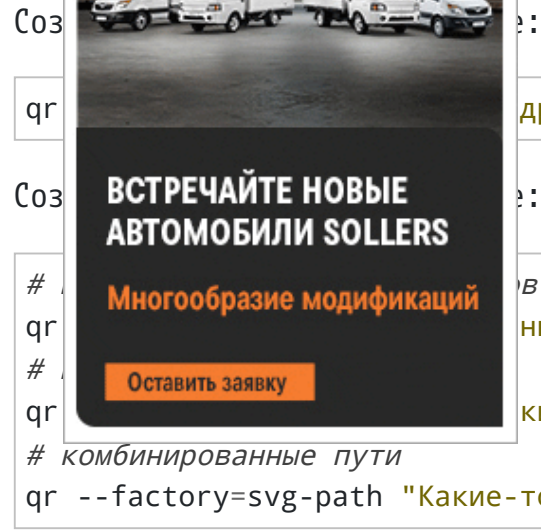
Список доступных цветных масок для использования с аргументом `color_mask`:

- `SquareModuleDrawer()`: Рисует модули в виде простых квадратов.
- `GappedSquareModuleDrawer(size_ratio=0.8)`: Отображает модули в виде простых квадратов, которые не являются смежными. Аргумент `size_ratio` определяет ширину квадратов относительно ширины пространства, в котором они печатаются.
- `CircleModuleDrawer()`: Рисует модули в виде кругов.
- `RoundedModuleDrawer(radius_ratio=1)`: Рисует модули, при этом все углы 90 градусов заменяются закругленными краями. Аргумент `radius_ratio` определяет радиус закругленных краев - значение 1 означает, что изолированный модуль будет нарисован в виде круга, в то время как значение 0 означает, что радиус закругленного края будет равен 0 (и, таким образом, снова вернется к 90 градусам).

- `VerticalBarsDrawer(horizontal_shrink=0.8)`: Рисует смежные по вертикали группы модулей в виде длинных закругленных прямоугольников с промежутками между соседними полосами (размер этих промежутков обратно пропорционален `horizontal_shrink`).
- `HorizontalBarsDrawer(vertical_shrink=0.8)`: Рисует смежные по горизонтали группы модулей в виде длинных закругленных прямоугольников с промежутками между соседними полосами (размер этих промежутков обратно пропорционален `vertical_shrink`).

## Использование модуля qrcode из командной строки.

Создание QR-кода из командной строки, для этого используйте установленный скрипт с названием `qr`.



```
qr --factory=svg-path "Какие-то данные или URL-адрес" > test.png
qr --factory=svg-path "Какие-то данные или URL-адрес" > test.svg
qr --factory=svg-path "Какие-то данные или URL-адрес" > test.svg
qr --factory=svg-path "Какие-то данные или URL-адрес" > test.svg
```

Создание QR-кода в текстовом виде:

```
qr --ascii "Какие-то данные или URL-адрес" > "test.txt"
cat test.txt
```

Для Windows!!! Альтернатива передаче вывода в файл, чтобы избежать проблем с PowerShell:

```
# qr "Some data" > test.png
qr --output=test.png "Some data"
```