Python Imaging Library (PIL)

python

Данная публикация представляет собой перевод самых необходимых модулей пакета для работы с изображениями на языке **Python**.

Перевод **Python Imaging Library (PIL)** делался по официальной документации и содержит ряд самых необходимых модулей:

1. **The Image Module** — Модуль содержит функций, методы и свойства для открытия, сохранения и манипулирования изображениями;
2. **The ImageChops Module** — Модуль содержит много арифметических операций над изображениями;
3. **The ImageColor Module** — Модуль содержит функции для преобразования строки определения цвета в кортеж формата RGB;
4. **The ImageDraw Module** — Модуль для рисования простой 2D-графики. Используется для рисования, создания новых изображений, создания текста и ретуширования существующие изображения;
5. **The ImageGrab Module** – Модуль содержат функции которые помогают сделать снимок экрана.
6. **The ImageFont Module** – Модуль содержит функционал для работы с TrueType и OpenType шрифтами.

Сразу хочу отметить, что это не дословный перевод. Материал направлен на новичков стремящихся разобраться в возможностях данного пакета. За более подробной информации по каждому инструменту рекомендуется обратиться к официальной документации.

Скачать пакет для Python версий 2 и 3 можно от сюда:
http://www.lfd.uci.edu/~gohlke/pythonlibs/

The Image Module

Модуль **Image** содержит функций, методы и свойства для открытия,

сохранения и анализа изображений.

Функции

new
Image.new(mode, size) => изображение
Image.new(mode, size, color) => изображение

Создает новое изображение с данным режимом и размером. Размер задаётся виде котрежа “(100,100)”. Если цвет опущен то изображение будет заполнено черным цветом. Режимы могут быть: **1** (черно-белый), **L** (монохромный, оттенки серого), **RGB**, **RGBA** (RGB с альфа каналом), **CMYK**, **YCbCr**, **I** (32 bit Integer pixels), **F** (32 bit Float pixels).

open
Image.open(infile) => изображение
Image.open(infile, mode) => изображение

Открывает и идентифицирует файл изображения.

```
>>> from PIL import Image
>>> Image.open(‘C:\test\pakmen.jpg’).show()
```

blend
Image.blend(image1, image2, alpha) => изображение

Создает новое изображение путем интерполяции между заданными изображениями, с помощью постоянной альфа. Оба изображения должны иметь одинаковый размер и режим.

```
>>> from PIL import Image
>>> im1 = Image.open(‘C:\test\pakmen.jpg’)
>>> im2 = Image.open(‘C:\test\pakmen2.jpg’)
>>> Image.blend(im1, im2, 50).show()
```

composite
Image.composite(image1, image2, mask) => изображение

Создает новое изображение путем интерполяции между заданными изображениями, используя маску, как альфа. Маска изображение может

иметь режим “1”, “L”, или “RGBA”. Все изображения должны быть одинакового размера.

eval
Image.eval(image, function) => изображение

Применяет функцию для каждого пикселя в данном изображении.

frombuffer
Image.frombuffer(mode, size, data) => изображение

Создает образ памяти из пиксельных данных в строке или буфере, используя стандартный “сырой” декодер.

fromstring
Image.fromstring(mode, size, data) => изображение

Создает образ памяти из пиксельных данных в строке или буфере, используя стандартный “сырой” декодер.

merge
Image.merge(mode, bands) => изображение

Создает новое изображение из нескольких отдельных изображений. Bands задаются как кортеж или список изображений, по одному для каждой Bands описывается режиме. Все Bands должны иметь одинаковые размеры.

Методы

convert
im.convert(mode) => изображение

Возвращает преобразованную копию изображения.

im.convert(mode, matrix) => изображение

Преобразует “RGB” изображение в “L” или “RGB”, используя матрицы преобразования.
Матрица представляет собой 4 — или 16-кортеж.

copy

im.copy() => изображение

Копирует изображение

crop
im.crop(box) => изображение

Возвращает прямоугольную область от текущего изображения. Box это 4-кортеж определяющие левый, верхний, правый и нижний пиксель координат.

```
>>> from PIL import Image
>>> im = Image.open('C:\test\pakmen.jpg')
>>> im.crop((0,10,80,80)).show()
```

draft
im.draft(mode, size)

Настраивает загрузчик файлов изображений, чтобы он возвращал версию изображения, которые как можно более точно соответствует данному режиму и размеру. Например, можно использовать этот метод для преобразования цветов в оттенки серого JPEG при загрузке, или для извлечения 128×192 версии с PCD файла.

filter
im.filter(filter) => изображение

Возвращает копию изображения фильтруя заданным фильтром. Для получения списка доступных фильтров, см. модуль ImageFilte.

fromstring
im.fromstring(data)
im.fromstring(data, decoder, parameters)

То же, что fromstring функции, но загружает данные в текущее изображение.

getbands
im.getbands() => кортеж строк

Возвращает кортеж, содержащий имя каждой группы. Например, getbands на

изображение RGB возвращается (“R”, “G”, “B”).

getbbox
im.getbbox() => 4-кортеж или None

Вычисляет ограничивающий прямоугольник ненулевых областей в изображении. Если изображение пусто то вернёт None.

```
>>> from PIL import Image
>>> im = Image.open(‘C:\test\pakmen.jpg’)
>>> im.getbbox()
(0, 0, 150, 120)
```

getcolors
im.getcolors() => список кортежей (число, цвет) или None
im.getcolors(maxcolors) => а список кортежей (число, цвет) или None

Возвращает несортированный список кортежей (число, цвет), где подсчитывается, сколько раз соответствующий цвет встречается в изображении.

Если значение maxcolors будет превышено, метод прекращает счет и возвращает None. Значение по умолчанию maxcolors 256.

getdata
im.getdata() => последовательность

Возвращает содержимое изображения в виде последовательности, содержащий значения пикселей.

getextrema
im.getextrema() => 2-котеж

Возвращает 2-кортеж, содержащий минимальное и максимальное значения изображение.

getpixel
im.getpixel(xy) => значение или кортеж

Возвращает пиксель в данной позиции.

```
>>> from PIL import Image
>>> im = Image.open('C:\test\pakmen.jpg')
>>> im.getpixel((100,50))
(5, 5, 5)
```

histogram
im.histogram() => список

Возвращает гистограмму изображения. Гистограмма возвращается как список пикселей.

im.histogram(mask) => список

Возвращает гистограммы для тех частей изображения, где изображения маски не равно нулю.

load
im.load()

Загружает пиксели из файла изображения.

```
>>> from PIL import Image
>>> im = Image.open('C:\test\pakmen.jpg')
>>> pix = im.load()
>>> pix[1,1]
(5, 5, 5)
>>> pix[1,1] = (255, 255, 255)
>>> im.show()
```

paste
im.paste(image, box)

Вставляет изображение в другое. Box это либо 2-кортеж определяющий верхний левый угл или 4-кортеж определяющий левый, верхний, правый и нижний пиксель координат или None (то же, (0, 0)). Если 4-кортеж то область должна соответствовать размеру изображения.

im.paste(colour, box)

То же самое, но заполняет область цветом.

im.paste(image, box, mask)

Тоже самое но с маской.

im.paste(colour, box, mask)

Тоже самое но с маской.

```
>>> from PIL import Image
>>> im = Image.open('C:\test\pakmen.jpg')
>>> im.paste('#FF0000', (0,0,100,100))
>>> im.show()
```

point
im.point(table) => изображение
im.point(function) => изображение

Возвращает копию изображения, где каждый пиксель был сопоставлен по данной таблице.

putalpha
im.putalpha(band)

putdata
im.putdata(data)
im.putdata(data, scale, offset)

putpalette
im.putpalette(sequence)

Палитра для "P" или "L" изображений.

putpixel
im.putpixel(xy, colour)

Изменяет цвет пикселя в данной позиции.

```
>>> from PIL import Image
>>> im = Image.open('C:\test\pakmen.jpg')
```

```
>>> im.putpixel((10,10), (255, 255, 255))
>>> im.show()
```

resize
im.resize(size) => изображение
im.resize(size, filter) => изображение

Возвращает копию изображения с измененными размерами. Фильтр может быть: NEAREST, BILINEAR, BICUBIC, ANTIALIAS.

```
>>> from PIL import Image
>>> im = Image.open('C:\test\pakmen.jpg')
>>> im.resize((100,100)).show()
```

rotate
im.rotate(angle) => изображение
im.rotate(angle, filter=NEAREST, expand=0) => изображение

Возвращает копию изображения повернутую на указанное кол-во градусов против часовой стрелки вокруг её центра. Фильтр может быть: NEAREST, BILINEAR, BICUBIC.

save
im.save(outfile, options...)
im.save(outfile, format, options...)

Сохраняет изображение в файл. Если формат не указан, формат определяется по расширению файла, если это возможно. Возвращает None. Можно использовать файловый объект, а не имя файла, в этом случае необходимо определять формат. Объект файла должен реализовывать seek , tell и write методы, должен быть открыт в двоичном режиме.

```
>>> from PIL import Image
>>> im = Image.open('C:\test\pakmen.jpg')
>>> im.save('/test/new_pakmen.png', 'PNG')
```

seek
im.seek(frame)

show
im.show()

Отображение изображения. Этот метод предназначен в основном для отладки.

split
im.split() => последовательность

Возвращает кортеж из отдельных полос изображения с изображениями. Например, расщепление “RGB” создает три новых изображения каждый из которых содержит копию одного из оригинальных полос (красный, зеленый, синий).

tell
im.tell() => число

Возвращает номер текущего кадра.

thumbnail
im.thumbnail(size)
im.thumbnail(size, filter)

Изменяет размер самого изображения. Фильтр может принимать значения NEAREST , BILINEAR , BICUBIC , или ANTIALIAS.

tobitmap
im.tobitmap() => строка

Возвращает изображение преобразованное в X11 растровое изображения.

tostring
im.tostring() => строка

Возвращает строку, содержащую данные пикселей, с использованием стандартного “сырого” энкодера.

im.tostring(encoder, parameters) => строка

Возвращает строку, содержащую данные пикселей, используя данный

кодирования данных.

transform
im.transform(size, method, data) => изображение
im.transform(size, method, data, filter) => изображение

Создает новое изображение заданного размера, и того же режима, что и оригинал. Метод может быть: EXTENT, AFFINE, QUAD, MESH. См. документацию.

transpose
im.transpose(method) => изображение

Возвращает повёрнутую копию изображения. Метод может иметь следующий вид:
FLIP_LEFT_RIGHT, FLIP_TOP_BOTTOM, ROTATE_90, ROTATE_180, или ROTATE_270.
Или число от 1 до 4, в зависимости от версий модуля.

```
>>> from PIL import Image
>>> im = Image.open('C:\test\pakmen.jpg')
>>> im.transpose(2).show()
```

verify
im.verify()

Пытается определить поврежден ли файл, без фактического декодирования данных изображения. Если этот метод находит какие-либо проблемы, то возникает подходящее исключения. Этот метод работает только на вновь открытых изображениях, если изображение уже загружено, результат не определен.

Свойства

format
im.format => строка или None

Формат файла исходного файла.

mode

im.mode => строка

Режим изображения. Типичные значения “1”, “L”, “RGB”, или “CMYK”.

size
im.size => (width, height)

Размер изображения в пикселях. Размер указан в виде 2-кортежа (ширина, высота).

palette
im.palette => палитру или None

Цветовая палитра таблицы, если имеется.

info
im.info => словарь

Словарь данных, связанных с изображением.

The ImageChops Module

Модуль **ImageChops** содержит множество арифметических операций над изображениями. Они могут быть использованы для различных целей, в том числе специальные эффекты, изображение композиции, алгоритмической живописи, и многое другое.

Операции реализованы только для 8-битных изображений (например, “L” и “RGB”).

constant
ImageChops.constant(image, value) => изображение

Вернёт новый слой размером с указанное изображение и заполненный цветом указанным значением value (0–255).

```
>>> from PIL import Image, ImageChops
>>> im = Image.open(“C:\test\pakmen.jpg”)
>>> ImageChops.constant(im, 255).show()
```

duplicate

ImageChops.duplicate(image) => изображение

Копирует изображение.

invert
ImageChops.invert(image) => изображение

Инверсия изображением.

lighter
ImageChops.lighter(image1, image2) => изображение

Сравнивает два изображения, пиксель за пикселем, и возвращает новое изображение, содержащее “лёгкие” цвета.

```
>>> from PIL import Image, ImageChops
>>> im1 = Image.open(“C:\test\pakmen.jpg”)
>>> im2 = Image.open(“C:\test\pakmen2.jpg”)
>>> ImageChops.lighter(im1, im2).show()
```

darker
ImageChops.darker(image1, image2) => изображение

Сравнивает два изображения, пиксель за пикселем, и возвращает новое изображение, содержащее “тёмные” цвета.

difference
ImageChops.difference(image1, image2) => изображение

Возвращает абсолютное значение разницы между двумя изображениями.

```
>>> from PIL import Image, ImageChops
>>> im1 = Image.open(“C:\test\pakmen.jpg”)
>>> im2 = Image.open(“C:\test\pakmen2.jpg”)
>>> ImageChops.difference(im1, im2).show()
```

multiply
ImageChops.multiply(image1, image2) => изображение

Накладывает два изображения друг на друга. Если умножить изображение с

черным изображение, то результат будет черным.

screen
ImageChops.screen(image1, image2) => изображение

Накладывает два инвертированные изображения друг на друга.

add
ImageChops.add(image1, image2, scale, offset) => изображение

Добавляет два изображения, разделив результат на масштаб и добавив смещение. Масштаб по умолчанию 1.0, и смещение 0.0.

subtract
ImageChops.subtract(image1, image2, scale, offset) => изображение

Вычитание двух изображений, разделив результат на масштаб и добавив смещение. Масштаб по умолчанию 1.0, и смещение 0.0.

blend
ImageChops.blend(image1, image2, alpha) => изображение

То же, что функция blend в модуле Image.

composite
ImageChops.composite(image1, image2, mask) => изображение

То же, что функция composite в модуле Image.

The ImageColor Module

Модуль **ImageColor** содержит функции для преобразования строки определяющую цвет в RGB кортеж. Этот модуль используется в Image.new и ImageDraw модуле.

Модуль ImageColor поддерживает следующие форматы строк:

1. Шестнадцатеричные спецификаторы цвета, вида “#RGB” или “#RRGGBB”. Например, “# ff0000”;
2. RGB функции, вида “rgb(красный, зеленый, синий)”. Например,

“rgb(255,0,0)” и “rgb(100%, 0%, 0%)”;

3. HSL функций (Тон-Насыщенность-Яркость) заданных в качестве “HSL (тон, насыщенность%, яркость%)”, где тон цвета дается как угол между 0 и 360, насыщение значения между 0% и 100%, и яркость это значение от 0% до 100. Например, “hsl(0,100%, 50%)”;
4. HTML названия цветов. Модуль ImageColor поддерживает около 140 стандартных названий цветов. Например “red” или “white”.

Функции

getrgb
getrgb(color) => (красный, зеленый, синий)

Преобразует строку обозначающую цвет в RGB кортеж.

```
>>> from PIL import ImageColor
>>> ImageColor.getrgb(‘hsl(0,100%, 50%)’)
(255, 0, 0)
>>> ImageColor.getrgb(’#FF0000’)
(255, 0, 0)
>>> ImageColor.getrgb(‘red’)
(255, 0, 0)
>>>
```

getcolor
getcolor(color, mode) => (красный, зеленый, синий) или целое число

То же что и функция getrgb, но можно задать режим, “RGB”, “RGBA” или “L”

```
>>> from PIL import ImageColor
>>> ImageColor.getcolor(’#FF0000’, ‘RGBA’)
(255, 0, 0, 255)
```

The ImageDraw Module

Модуль содержит инструменты для рисования простой 2D-графики.

Формат цвета такой же как в модуле **ImageColor**.

Шрифты — PIL можете использовать растровые шрифты или OpenType / TrueType шрифты. Для загрузки шрифтов OpenType / TrueType следует использовать TrueType функции из модуля **ImageFont**.

Функции

Draw(image) => объект Draw

Создает объект, который можно использовать для рисования в данном изображении.

Методы

arc
draw.arc(xy, start, end, fill=None)

Рисует дугу между начальными и конечными углами, внутри данной ограничительной рамки.

```
>>> from PIL import Image, ImageDraw
>>> im = Image.open(“C:\test\pakmen.jpg”)
>>> draw = ImageDraw.Draw(im)
>>> draw.arc( (40, 74, 80, 84), 10, 120, ”#FF0000” )
>>> im.show()
```

bitmap
draw.bitmap(xy, bitmap, fill=None)

Рисует изображение (маска) на данной позиции, используя текущий цвет заливки.

chord
draw.chord(xy, start, end, fill=None, outline=Nonez)

То же, что дуга , но соединяет концы прямой линией.

ellipse
draw.ellipse(xy, fill=None, outline=None)

Рисует элипс.

```
>>> from PIL import Image, ImageDraw
>>> im = Image.open(“C:\test\pakmen.jpg”)
>>> draw = ImageDraw.Draw(im)
>>> draw.ellipse((10,10,50,50), fill=“red”, outline=“red”)
>>> im.show()
```

line
draw.line(xy, fill=None, width=0)

Рисует линию

```
>>> from PIL import Image, ImageDraw
>>> im = Image.open(“C:\test\pakmen.jpg”)
>>> draw.line((0, 30, 30, 0), fill=”#FF0000”, width=5)
>>> im.show()
```

Аргумент может быть списком, кортежем или списком кортежей “[(0, 30), (30, 0)]”

pieslice
draw.pieslice(xy, start, end, fill=None, outline=None)

То же, что дуга, но и рисует прямые линии между конечными точками и в центре ограничительной рамки.

point
draw.point(xy, fill=None)

Рисует точку (отдельных пикселей) на заданной координате.

polygon
draw.polygon(xy, fill=None, outline=None)

Рисует многоугольник

rectangle
draw.rectangle(box, fill=None, outline=None)

Рисует прямоугольник

```
>>> from PIL import Image, ImageDraw
>>> im = Image.open(“C:\test\pakmen.jpg”)
>>> draw = ImageDraw.Draw(im)
>>> draw.rectangle((10,10,150,150), fill=”#FF0000”, outline=
”#FFFFFF”)
>>> im.show()
```

text
draw.text(position, string, fill=None, font=None)

Рисует строку в заданную позицию.

```
>>> from PIL import Image, ImageDraw
>>> im = Image.open(“C:\test\pakmen.jpg”)
>>> draw = ImageDraw.Draw(im)
>>> draw.text((10,10), “hello”, fill=“black”)
>>> im.show()
```

textsize
draw.textsize(string, font=None) => (width, height)

Возвращает размер строки в пикселях.

The ImageGrab Mobule

Модуль I**mageGrab** может быть использован для копирования содержимого экрана (скриншот). Текущая версия работает только в Windows.

grab
ImageGrab.grab() => изображение
ImageGrab.grab(BBOX) => изображение

Сделает снимок экрана, и вернёт “RGB” изображение. При использование с аргументом можно копирования только часть экрана. BBOX это кортеж определяющий координаты левого верхнего пикселя и ширина с высотой.

```
>>> from PIL import ImageGrab, Image
>>> ImageGrab.grab().show()
```

```
>>> ImageGrab.grab((100,100,200,200)).show()
```

grabclipboard
ImageGrab.grabclipboard () => изображений или список строк или None

Сделает снимок содержимого буфера обмена и вернёт изображение или список имен файлов. Если буфер обмена не содержит данных, то функция возвратит None.

The ImageFont Module

Модуль используется для работы с **TrueType** и **OpenType** шрифтами. Можно использовать совместно с модулем **ImageDraw**.

```
import ImageFont, ImageDraw
draw = ImageDraw.Draw(image)
```

# use a bitmap font
font = ImageFont.load(“arial.pil”)
draw.text((10, 10), “hello”, font=font)

# use a truetype font
font = ImageFont.truetype(“arial.ttf”, 15)
draw.text((10, 25), “world”, font=font)

Функции

load
ImageFont.load(file) => объект шрифта

Загружает шрифт из указанного файла, и возвращает соответствующий объект шрифта.

load_path
ImageFont.load_path(file) => объект шрифта

То же, что функция load, но ищет файл вместе с sys.path, если он не найден в текущем каталоге.

truetype

ImageFont.truetype(file, size) => объект шрифта

Загружает TrueType или OpenType файл шрифта и создаёт объект со шрифтом заданного размера.

ImageFont.truetype(file, size, encoding=value) => объект шрифта

Загружает TrueType или OpenType файл шрифта и создаёт объект со шрифтом заданного размера. И задаёт кодировку: “UNIC” (Unicode), “Symb” (Microsoft Symbol), “ADOB” (Adobe Standard), “ADBE” (Adobe Expert), и “armn” (Apple Roman).

load_default
ImageFont.load_default() => объект шрифта

Загружает шрифт по умолчанию.

Методы

getsize
font.getsize(text) => (width, height)

Возвращает кортеж с размером шрифта.

getmask
font.getmask(text, mode=””) => изображение

Возвращает изображение для текста.

#python
