Масштабирование и обрезка изображений. Сохранение в базу данных

Эффекты для изображений. Сохранение в базу SQLite и MySQL.

[Масштабирование изображений](#_u1o9f6frpxpk)

[Обрезка изображений](#_wvftzffleig7)

[Сохранение изображений в SQLite](#_s2kkcwhssanw)

[Сохранение изображений с помощью SQLAlchemy](#_xv97gkxiw0u0)

[Передача изображений на сервер и сохранение ссылки на них в MySQL](#_h1kx9vmf5mee)

[Практика](#_a7fiora8azhg)

[Практическое задание](#_1t3h5sf)

[Дополнительные материалы](#_4d34og8)

[Используемая литература](#_2s8eyo1)

# 

# Масштабирование изображений

Чтобы изменить размер изображения, будем использовать функцию **resize**.

В качестве одного из элементов она принимает значение фильтра. Это может быть **NEAREST**, **BILINEAR**, **BICUBIC** и **ANTIALIAS**.

Указываем подключаемые библиотеки:

|  |
| --- |
| from PIL import Image, ImageDraw  from PIL.ImageQt import ImageQt |

После создаем базовый каркас программы:

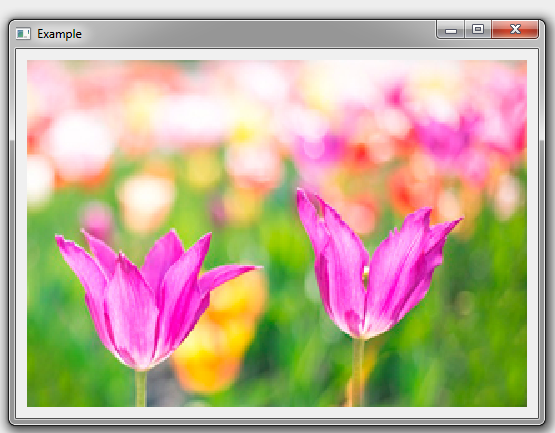
|  |
| --- |
| imageFile = "image.jpg"   image = Image.open(imageFile)   width = 500  height = 347    draw = ImageDraw.Draw(image)   img\_tmp = ImageQt(image.convert('RGBA'))   hbox = QHBoxLayout(self)  pixmap = QPixmap.fromImage(img\_tmp)   lbl = QLabel(self)  lbl.setPixmap(pixmap)   hbox.addWidget(lbl)  self.setLayout(hbox)   self.move(300, 200)  self.setWindowTitle('Example')  self.show() |

В каркасе загружаем изображение и выводим его в **QLabel**.

Сделаем функцию масштабирования с фильтром **NEAREST** — самым примитивным и быстрым. Для каждого конечного пикселя изображения выбирается наиболее близкий пиксель с учетом масштабирования. Этот метод дает пикселизированное изображение при увеличении и зернистое при уменьшении.

|  |
| --- |
| image = image.resize((width, height), Image.NEAREST) |

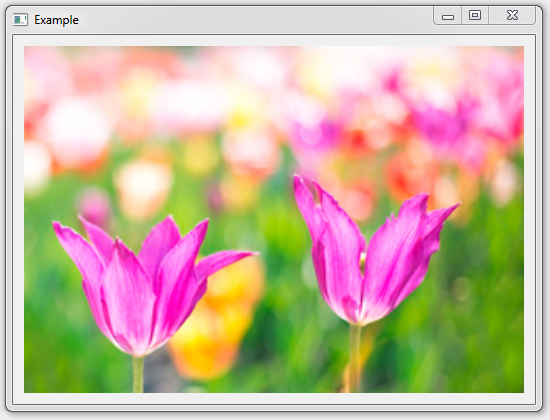
Все это можно увидеть в получившейся программе:



Применим фильтр **BILINEAR**. Немного изменим код:

|  |
| --- |
| image = image.resize((width, height), Image.BILINEAR) |

Результат:



Следующий фильтр — **BICUBIC**. Немного изменим программу:

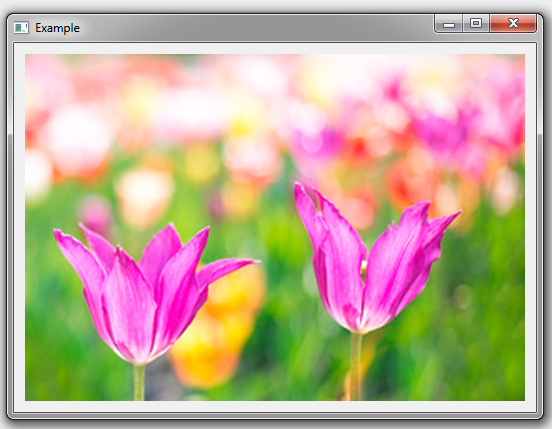
|  |
| --- |
| image = image.resize((width, height), Image.BICUBIC) |

Получится:



Рассмотрим последний фильтр — **ANTIALIAS**. Поменяем код:

|  |
| --- |
| image = image.resize((width, height), Image.ANTIALIAS) |



## 

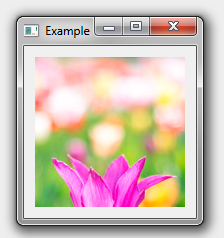
## Обрезка изображений

Обрезать изображение можно с помощью функции **crop**. Четырьмя точками зададим область под кадрирование. Напишем базовый каркас и используем там инструмент **crop**:

|  |
| --- |
| imageFile = "image.jpg"   image = Image.open(imageFile)   image = image.crop((0, 0, 150, 150))   draw = ImageDraw.Draw(image)   img\_tmp = ImageQt(image.convert('RGBA'))   hbox = QHBoxLayout(self)  pixmap = QPixmap.fromImage(img\_tmp)   lbl = QLabel(self)  lbl.setPixmap(pixmap)   hbox.addWidget(lbl)  self.setLayout(hbox)   self.move(300, 200)  self.setWindowTitle('Example')  self.show() |

Этот код обрезает изображение по точкам 0,0 и 150,150:

|  |
| --- |
| image = image.crop((0, 0, 150, 150)) |



# Сохранение изображений в SQLite

Все, что мы напишем, должно выполняться на сервере: клиент взаимодействует со скриптом, а сервер исполняет.

Для локального сохранения будем использовать SQLite. Сначала создадим базу, а затем в командной строке SQLite выполним следующее:

|  |
| --- |
| SQLite> CREATE TABLE Images(Id INTEGER PRIMARY KEY, Data BLOB); |

Подключим библиотеку, необходимую для работы с базой:

|  |
| --- |
| import SQLite3 as lite |

Эта команда создаст таблицу **Images** с полями **Id** и **Data**. Напишем код:

|  |
| --- |
| con = lite.connect('image.db')  cur = con.cursor()  file = open("image.jpg", "rb")  img = file.read()  binary = lite.Binary(img)  cur.execute("INSERT INTO Images(Data) VALUES (?)", (binary,))  con.commit()  con.close()  file.close() |

Сначала подключаемся к базе с помощью этой команды:

|  |
| --- |
| lite.connect('image.db') |

Затем открываем и считываем файл:

|  |
| --- |
| file = open("image.jpg", "rb")   img = file.read()  file.close() |

Вставляем считанный файл в базу данных:

|  |
| --- |
| binary = lite.Binary(img)  cur.execute("INSERT INTO Images(Data) VALUES (?)", (binary,)) |

И не забываем закрыть соединение с базой и файл:

|  |
| --- |
| con.close()  file.close() |

Результат — добавили в базу данных наш файл:



## Сохранение изображений с помощью SQLAlchemy

Подключаем необходимые библиотеки.

|  |
| --- |
| from sqlalchemy import Column, ForeignKey, Integer, String, BLOB from sqlalchemy.ext.declarative import declarative\_base from sqlalchemy.orm import relationship from sqlalchemy import create\_engine from sqlalchemy.orm import sessionmaker |

В специальном классе описываем структуру таблицы в базе данных, в которую идет сохранение.

|  |
| --- |
| class Image(Base):  \_\_tablename\_\_ = 'image'  id = Column(Integer, primary\_key=True)  Data = Column(BLOB) |

Пишем код, который загружает изображение и сохраняет его в базу.

|  |
| --- |
| engine = create\_engine('SQLite:///image.db')  engine.echo = True  session = sessionmaker() session.configure(bind=engine) Base.metadata.create\_all(engine)  file = open("image.jpg", "rb")  img = file.read()  file.close()  s = session() images = Image( Data = img ) s.add(images) s.commit() |

Подключаем базу, создаем сессию и сохраняем изображение.

# Передача изображений на сервер и сохранение ссылки на них в MySQL

***С этим разделом ознакомьтесь самостоятельно.***

Рассмотрим тему на примере кода, который сохраняет имя изображения в базе MySQL. Главные преимущества этой БД — она быстро работает с данными, поддерживает многопоточность и бесплатна практически во всех случаях. Покажем, как передать изображение напрямую с компьютера клиента по протоколу **ftp**.

Сохраняя изображения на сервере, будем использовать **ftp**, так как у него нет ограничений по скорости, быстрый обмен файлами и поддерживается докачка.

|  |
| --- |
| FTP(хост, ftp-пользователь, пароль пользователя, учетная запись, время ожидания) |

Подключаем необходимые библиотеки:

|  |
| --- |
| import ftplib import MySQLdb |

Пишем код, который будет передавать изображение на сервер:

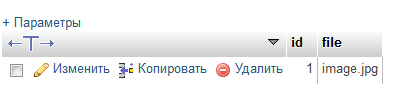
|  |
| --- |
| host = "host" *# Ваш хост*  ftp\_user = "user" *# Имя пользователя*  ftp\_password = "password" *# Пароль*  filename = "image.jpg"   con = ftplib.FTP(host, ftp\_user, ftp\_password)  file = open(filename, "rb")  send = con.storbinary("STOR " + filename, file)  con.close  file.close() |

Для работы с MySQL используем библиотеку **MySQLdb**.

Когда передали файл на сервер, добавим код, который будет записывать изображение в базу MySQL на сервере.

|  |
| --- |
| host = "host"  MySQL\_db = "database"  MySQL\_user = "user"  MySQL\_password = "password"   try:  conn = MySQLdb.connect(host, MySQL\_user, MySQL\_password, MySQL\_db)   except MySQLdb.Error as err:  print("Connection error: {}".format(err))  conn.close()   sql = "INSERT INTO image(file) VALUES('image.jpg');"   conn.autocommit(True)   try:  cur = conn.cursor()  cur.execute(sql)   except MySQLdb.Error as err:  print("Query error: {}".format(err))   conn.close() |

В базу данных записалось имя изображения.



# Практика

Если на уроке остается время, решаем задачи:

1. Разработать программу, которая обрезает изображение по выделенному прямоугольнику.
2. Написать программу, которая с помощью **SQLAlchemy** сохраняет в базу обрезанное изображение.

# Практическое задание

Продолжаем работать над мессенджером: задачи выполняем с использованием базы данных.

1. Сделать сохранение аватара в базу.
2. Применить масштабирование к аватару и сохранить его в локальную базу.
3. Применить обрезку к аватару и сохранить его в локальную базу.
4. \* Применить масштабирование и обрезку к аватару. Передать файл на сервер и записать его в базу.
5. \* Сделать сохранение в базу текстовых сообщений.

# Дополнительные материалы

1. [Pillow 2.7 — существенное улучшение качества и производительности](https://habrahabr.ru/post/247219/).
2. [Работа с FTP](http://python-3.ru/page/python-ftp-ftplib).

# Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. [Ресайз (resize) картинок в Python](http://dmitry.miramik.ru/2010/09/03/resajz-resize-kartinok-v-python/).
2. [Ликбез: методы ресайза изображений](https://habrahabr.ru/post/243285/).
3. [Учим Python качественно](https://habrahabr.ru/post/150302/).
4. [Самоучитель по Python](http://pythonworld.ru/samouchitel-python).
5. [Марк Лутц. Изучаем Python (4-е издание)](http://www.proklondike.com/books/python/lutz_python_2011.html).