|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  (МГУПИ) |

Институт КБСП направление 09.03.02

Кафедра КБ-4 «Автоматизированные системы управления»

Дисциплина «Технология программирования в среде Python»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

**ТЕМА**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Фамилия И.О.

подпись, дата

Группа БСБО-01-15

Работа защищена на оценку\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев А.С.

подпись, дата

Члены комиссии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись, дата инициалы и фамилия

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«Московский технологический университет»**  (МГУПИ) |

Институт КБСП направление 09.03.02

Кафедра КБ4 «Автоматизированные системы управления»

Дисциплина «Технология программирования в среде Python»

**ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ**

Студент: 3 курса Гончаров Евгений Андреевич группа БСБО-01-15

1 Тема:

**Разработка Desktop приложения реализующее предметную область Туристического оператора "Airlands"**

2 Срок представления проекта (работы) к защите 23.12.2017 г.

3 Исходные данные для разработки

Высокоуровневый язык программирования Python

4 Содержание пояснительной записки:

Титульный лист

Содержание

Введение

1 Исследовательский раздел

2 Составляющие проекта

3 Программное обеспечение

4 Техническое задание

Заключение

Приложения

Список использованных источников

Руководитель проекта (работы)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Лебедев

подпись, дата

Задание принял к исполнению\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Лебедев

подпись, дата

**Содержание**

[Введение 4](#_Toc501696480)

[Составляющие проекта 6](#_Toc501696481)

[Техническое задание 7](#_Toc501696482)

[Основания для разработки. 7](#_Toc501696483)

[Назначение разработки. 7](#_Toc501696484)

[Требования к программе. 7](#_Toc501696485)

[1. Требования к функциональным характеристикам. 7](#_Toc501696486)

[2. Требования к составу и параметрам технических средств. 7](#_Toc501696487)

[3. Требования к информационной и программной совместимости. 7](#_Toc501696488)

[Стадии и этапы разработки. 8](#_Toc501696489)

[Заключение 8](#_Toc501696490)

[Приложение 9](#_Toc501696491)

[Список использованной литературы 14](#_Toc501696492)

**Введение**

**База данных** — это некоторый набор данных, организованный по определенным правилам и имеющий определенную структуру.

Другими словами база данных это хранилище данных. Базу данных можно сравнить с библиотекой, где книги хранятся в определенном порядке, позволяющем работнику быстро найти нужное произведение.

Существует большое количество разновидностей баз данных, отличающихся по различным свойствам и критериям. К основным типам баз данных относятся:

* Иерархическая;
* Сетевая;
* Объектно-ориентированная;
* Реляционная.

Самыми распространенными являются реляционные базы данных. **Реляционная база данных** состоит из таблиц, которые в свою очередь состоят из строк и столбцов. Содержащиеся в таблицах данные связываются между собой по ключевым значениям.

Для работы с базами данных используются специальные программные средства — **системы управления базами данных** (СУБД). **СУБД** позволяет создавать базы данных, осуществлять доступ к данным, выполнять различные манипуляции с данными (добавлять, редактировать, удалять) и обеспечивать безопасность данных.

Для оперирования данными в реляционной базе данных с помощью СУБД используется специальный язык SQL.

**SQL** (structured query language) — в переводе с английского язык структурированных запросов, применяющийся для создания, изменения и удаления данных.

Применение баз данных в туризме — один из способов увеличения эффективности использования данных организаций, работающих в туристической сфере.

Использование БД позволяет решить следующие важные задачи: хранение информации о клиентах — в данном случае БД являются источником информации с возможностью оперативного получения данных о параметрах клиентов и их заявках; информирование клиентов — в данном случае посредством БД пользователям предоставляется информация об услугах организации.

Первая задача может быть реализована двумя способами: работники туристической компании могут вносить необходимые данные самостоятельно (внутренний характер использования), или информация будет заноситься пользователями посредством интернета (внешний характер использования). Для решения второй задачи также могут быть использованы два способа: предоставление данных в статичном виде (пользователь увидит полный перечень услуг вкупе с их описанием), в динамичном виде (пользователь сможет осуществлять поиск нужных данных и выбор интересующих его услуг: приобретения билетов, бронирования гостиничных номеров и прочих) .

Таким образом, применение связки «база данных в туризме — интернет» позволяет оперативно получить все необходимые сведения как клиентам туристических фирм, так и их работникам.

Информационные технологии — важнейший инструмент развития туристического и социально-культурного сервиса. Эффективность применения баз данных в туризме определяет продуктивность деятельности организаций в туристической сфере, что обусловлено необходимостью обеспечить оперативность обработки информации и надёжность ей хранения. Поэтому успешная деятельность современной турфирмы невозможна без работы со специализированными программными ресурсами (базами данных), позволяющими автоматизировать рабочие процессы. А использование более сложных компьютерных сетей необходимо для эффективного контакта с клиентами, которые получат возможность резервировать места в гостиницах и приобретать услуги посредством интернета

**Цель**

Разработать базу данных для туристического агенства, которая будет помогать кадрам туристической фирмы выбирать страну, курорт, отель, рейс для поездки, а также будет помогать создавать заявку на выбранную поездку.

**Список задач**

Создать реляционную базу данных второй нормальной формы с 6 сущностями (таблицами).

Разработать таблицу графического интерфейса, выводящую записи сущностей.

Создать методы для добавления/удаления записей из таблиц.

Разработать виджет подробного просмотра информации по выбранной сущности.

**Составляющие проекта**

**PyCharm** — интегрированная среда разработки для языка программирования Python. Предоставляет средства для анализа кода, графический отладчик, инструмент для запуска юнит-тестов и поддерживает веб-разработку на Django.PyCharm разработана компанией JetBrains на основе IntelliJ IDEA.

**SQLite** - это библиотека, написанная на языке C, которая обеспечивает работу с SQL. Данный инструмент относится к Реляционным системам управления базами данных. Большинство баз данных SQL работает по схеме клиент/сервер

**PyQt** — набор «привязок» графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python.PyQt разработан британской компанией Riverbank Computing. PyQt работает на всех платформах, поддерживаемых Qt: Linux и другие UNIX-подобные ОС, Mac OS X и Windows

Текст исходного кода приведен в приложении 1 к пояснительной записке.

**UML** (англ. Unified Modeling Language — унифицированный язык моделирования) — язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур

**Git** — распределённая [система управления версиями](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8). Проект был создан [Линусом Торвальдсом](https://ru.wikipedia.org/wiki/%D0%A2%D0%BE%D1%80%D0%B2%D0%B0%D0%BB%D1%8C%D0%B4%D1%81,_%D0%9B%D0%B8%D0%BD%D1%83%D1%81" \o "Торвальдс, Линус) для управления разработкой [ядра Linux](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_Linux), первая версия выпущена [7 апреля](https://ru.wikipedia.org/wiki/7_%D0%B0%D0%BF%D1%80%D0%B5%D0%BB%D1%8F) [2005 года](https://ru.wikipedia.org/wiki/2005_%D0%B3%D0%BE%D0%B4). На сегодняшний день его поддерживает [Джунио Хамано](https://ru.wikipedia.org/wiki/%D0%A5%D0%B0%D0%BC%D0%B0%D0%BD%D0%BE,_%D0%94%D0%B6%D1%83%D0%BD%D0%B8%D0%BE" \o "Хамано, Джунио).

**Microsoft Visio** — векторный графический редактор, редактор диаграмм и блок-схем для Windows. Выпускается в трёх редакциях: Standard, Professional и Pro for Office 365.

Техническое задание

Основания для разработки.

Учебный план по дисциплине «Технология программирования в среде Python», направление бакалавриата «Информационные системы и Технологии» кафедры КБ4 «Автоматизированные системы управления», Московский технологический университет (МТУ), 2017 г.

Назначение разработки.

Разработанное Desktop-приложение предназначено для туристического агенства, база данных которого будет помогать кадрам туристической фирмы выбирать страну, курорт, отель, рейс для поездки, а также будет помогать создавать заявку на выбранную поездку.

Требования к программе.

К разработанной программе предъявляются следующие требования:

1. Требования к функциональным характеристикам.

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* 1. Выполнять .
  2. Накладывать фильтры: «оттенки серого», «сепия», «негатив», «шум»,«увеличение яркости», «черное/белое», «размытие» на исходное изображение.
  3. «Переворачивать» изображение по вертикали и горизонтали.
  4. Поворачивать изображение по(против) часовой стрелки.
  5. Окно приложения должно представлять рабочую область с панелью меню и панелью инструментов.

1. Требования к составу и параметрам технических средств.
   1. Монитор с разрешением 1024x768 (рекомендуется 1280x800), поддержкой 16-битного цвета, а также 512 МБ видеопамяти (рекомендуется 1 ГБ)
   2. Оперативная память объемом от 1 Гб.
   3. Свободное место на SSD или HDDдиске объемом не менее 80 Мб.
2. Требования к информационной и программной совместимости.

3.1 Требования к операционной системе

В состав технических средств должен входить персональный компьютер. на базе операционной системы Windows 7 и новее.

На персональном компьютере должны присутствовать следующие библиотеки следующих версий:

3.1.1. PyQt5 – это набор Python библиотек для создания графического интерфейса на базе платформы Qt5 от компании Digia.Требуемая версия – 5.9.2.

Стадии и этапы разработки.

1. Анализ имеющихся решений и существующих алгоритмов.

2. Разработка интерфейса и алгоритмов работы с изображением.

3. Разработка технического задания на разработку программы.

4. Разработка программного кода, реализующего функциональные требования к программе.

Разработка программного кода должна происходить на языке программирования Pythonверсии 3.6.4 в среде разработки PyCharm от JetBrains s.r.o.

Заключение

Разработать метод для «переворачивания» изображения.

Разработать метод для поворота изображения по(против) часовой стрелки.

Разработан метод для определения координат по которым осуществляется обрезка изображения.

Разработан методы для наложения фильтров: «оттенки серого», «сепия», «негатив», «шум», «увеличение яркости», «черное/белое», «размытие» на исходное изображение.

Разработан метод для «переворачивания» изображения.

Разработан метод для поворота изображения по(против) часовой стрелки

Приложение 1. Исходный код программы.

Main.py:

#!/usr/bin/env python

import os, sys, random

from PyQt5.QtCore import Qt, QRectF, pyqtSignal

from PyQt5.QtGui import QKeySequence, QImage, QPixmap, QPainterPath, QIcon

from PyQt5.QtWidgets import (QAction, QFileDialog, QApplication, QGraphicsScene,

QMainWindow, QMessageBox, QVBoxLayout, QGraphicsView, QWidget)

from PIL import Image, ImageDraw, ImageFilter

class MainWindow(QMainWindow):

class QtImageViewer(QGraphicsView):

"""

Left mouse button drag: Pan image.

Right mouse button drag: Zoom box.

Right mouse button doubleclick: Zoom to show entire image.

"""

leftMouseButtonPressed = pyqtSignal(float, float)

rightMouseButtonPressed = pyqtSignal(float, float)

leftMouseButtonReleased = pyqtSignal(float, float)

rightMouseButtonReleased = pyqtSignal(float, float)

leftMouseButtonDoubleClicked = pyqtSignal(float, float)

rightMouseButtonDoubleClicked = pyqtSignal(float, float)

def \_\_init\_\_(self):

QGraphicsView.\_\_init\_\_(self)

self.filename = ''

self.scene = QGraphicsScene()

self.setScene(self.scene)

self.\_pixmapHandle = None

self.aspectRatioMode = Qt.KeepAspectRatio

self.setHorizontalScrollBarPolicy(Qt.ScrollBarAsNeeded)

self.setVerticalScrollBarPolicy(Qt.ScrollBarAsNeeded)

self.zoomStack = []

self.canZoom = True

self.canPan = True

def hasImage(self):

return self.\_pixmapHandle is not None

def clearImage(self):

if self.hasImage():

self.scene.removeItem(self.\_pixmapHandle)

self.\_pixmapHandle = None

def pixmap(self):

if self.hasImage():

return self.\_pixmapHandle.pixmap()

return None

def image(self):

if self.hasImage():

return self.\_pixmapHandle.pixmap().toImage()

return None

def setImage(self, image):

""" Set the scene's current image pixmap to the input QImage or QPixmap.

Raises a RuntimeError if the input image has type other than QImage or QPixmap.

:type image: QImage | QPixmap

"""

if type(image) is QPixmap:

pixmap = image

elif type(image) is QImage:

pixmap = QPixmap.fromImage(image)

else:

raise RuntimeError("ImageViewer.setImage: Argument must be a QImage or QPixmap.")

if self.hasImage():

self.\_pixmapHandle.setPixmap(pixmap)

else:

self.\_pixmapHandle = self.scene.addPixmap(pixmap)

self.setSceneRect(QRectF(pixmap.rect()))

self.updateViewer()

def loadImageFromFile(self, fileName=""):

if len(fileName) == 0:

fileName, dummy = QFileDialog.getOpenFileName(self, "Open image file.")

if len(fileName) and os.path.isfile(fileName):

image = QImage(fileName)

self.filename = fileName

self.setImage(image)

def updateViewer(self):

if not self.hasImage():

return

if len(self.zoomStack) and self.sceneRect().contains(self.zoomStack[-1]):

self.fitInView(self.zoomStack[-1], Qt.IgnoreAspectRatio)

else:

self.zoomStack = self.fitInView(self.sceneRect(),

self.aspectRatioMode)

def resizeEvent(self, event):

self.updateViewer()

def mousePressEvent(self, event):

scenePos = self.mapToScene(event.pos())

if event.button() == Qt.LeftButton:

if self.canPan:

self.setDragMode(QGraphicsView.ScrollHandDrag)

self.leftMouseButtonPressed.emit(scenePos.x(), scenePos.y())

elif event.button() == Qt.RightButton:

if self.canZoom:

self.setDragMode(QGraphicsView.RubberBandDrag)

self.rightMouseButtonPressed.emit(scenePos.x(), scenePos.y())

QGraphicsView.mousePressEvent(self, event)

def mouseReleaseEvent(self, event):

QGraphicsView.mouseReleaseEvent(self, event)

scenePos = self.mapToScene(event.pos())

if event.button() == Qt.LeftButton:

self.setDragMode(QGraphicsView.NoDrag)

self.leftMouseButtonReleased.emit(scenePos.x(), scenePos.y())

elif event.button() == Qt.RightButton:

if self.canZoom:

viewBBox = self.zoomStack[-1] if len(self.zoomStack) else self.sceneRect()

selectionBBox = self.scene.selectionArea().boundingRect().intersected(viewBBox)

self.scene.setSelectionArea(QPainterPath()) # Clear current selection area.

if selectionBBox.isValid() and (selectionBBox != viewBBox):

self.zoomStack.append(selectionBBox)

self.updateViewer()

self.setDragMode(QGraphicsView.NoDrag)

self.rightMouseButtonReleased.emit(scenePos.x(), scenePos.y())

def mouseDoubleClickEvent(self, event):

scenePos = self.mapToScene(event.pos())

if event.button() == Qt.LeftButton:

self.leftMouseButtonDoubleClicked.emit(scenePos.x(), scenePos.y())

elif event.button() == Qt.RightButton:

if self.canZoom:

self.zoomStack = [] # Clear zoom stack.

self.updateViewer()

self.rightMouseButtonDoubleClicked.emit(scenePos.x(), scenePos.y())

QGraphicsView.mouseDoubleClickEvent(self, event)

def \_\_init\_\_(self):

super(MainWindow, self).\_\_init\_\_()

widget = QWidget()

self.setCentralWidget(widget)

self.img = self.QtImageViewer()

self.createActions()

self.createMenus()

self.createToolBars()

vbox = QVBoxLayout()

vbox.setContentsMargins(5, 5, 5, 5)

vbox.addWidget(self.img)

widget.setLayout(vbox)

self.setWindowTitle("Simple graphic editor")

self.setMinimumSize(480, 320)

self.resize(800, 600)

def open(self):

self.img.loadImageFromFile()

def save(self):

if self.img.filename:

file = self.img.filename

self.img.pixmap().save(file)

else:

print('error')

def saveAs(self):

file, \_ = QFileDialog.getSaveFileName(filter='\*.png \*.bmp \*.jpg')

print(file)

if file:

t = self.img.pixmap().save(file)

print(t)

else:

print('error')

def about(self):

QMessageBox.about(self, "About Menu",

"Простой графический редактор. Сделали Аваков Я.И. и Белоглазов Д.В.")

def aboutQt(self):

pass

def createActions(self):

self.openAct = QAction(QIcon('images/open.png'), "&Open...", self, shortcut=QKeySequence.Open,

triggered=self.open)

self.saveAct = QAction(QIcon('images/save.png'), "&Save", self, shortcut=QKeySequence.Save,

triggered=self.save)

self.saveAsAct = QAction(QIcon('images/save-as.png'), "&Save As...", self, shortcut='Ctrl+Shift+S',

triggered=self.saveAs)

self.exitAct = QAction(QIcon('images/exit.png'), "&Exit", self, shortcut="Ctrl+Q", triggered=self.close)

self.aboutAct = QAction(QIcon('images/about.png'), "&About", self,

triggered=self.about)

self.aboutQtAct = QAction(QIcon('images/aboutqt.png'), "About &Qt", self,

triggered=self.aboutQt)

self.aboutQtAct.triggered.connect(QApplication.instance().aboutQt)

self.greyAct = QAction("Greyscale", self, triggered=self.grey)

self.sepiaAct = QAction("Sepia", self, triggered=self.sepia)

self.wbAct = QAction("Black/White", self, triggered=self.wb)

self.negativeAct = QAction("Negative", self, triggered=self.negative)

self.brightAct = QAction("Brighter", self, triggered=self.bright)

self.blurAct = QAction("Blur", self, triggered=self.blur)

self.noiseAct = QAction("Noise", self, triggered=self.noise)

self.cropAct = QAction(QIcon('images/crop.png'), "Crop", self, triggered=self.crop)

self.rotateCWAct = QAction(QIcon('images/cw.png'), "CW", self, triggered=self.rotateCW)

self.rotateCCWAct = QAction(QIcon('images/ccw.png'), "CCW", self, triggered=self.rotateCCW)

self.mirrorAct = QAction(QIcon('images/mirror.png'), "Mirror", self, triggered=self.mirror)

self.flipAct = QAction(QIcon('images/flip.png'), "Flip", self, triggered=self.flip)

def createMenus(self):

self.fileMenu = self.menuBar().addMenu("&File")

self.fileMenu.addAction(self.openAct)

self.fileMenu.addAction(self.saveAct)

self.fileMenu.addAction(self.saveAsAct)

self.fileMenu.addSeparator()

self.fileMenu.addAction(self.exitAct)

self.editMenu = self.menuBar().addMenu("&Edit")

self.editMenu.addAction(self.rotateCWAct)

self.editMenu.addAction(self.rotateCCWAct)

self.editMenu.addAction(self.mirrorAct)

self.editMenu.addAction(self.flipAct)

self.helpMenu = self.menuBar().addMenu("&About")

self.helpMenu.addAction(self.aboutAct)

self.helpMenu.addAction(self.aboutQtAct)

self.formatMenu = self.editMenu.addMenu("&Filter")

self.formatMenu.addAction(self.greyAct)

self.formatMenu.addAction(self.sepiaAct)

self.formatMenu.addAction(self.negativeAct)

self.formatMenu.addAction(self.noiseAct)

self.formatMenu.addAction(self.brightAct)

self.formatMenu.addAction(self.wbAct)

self.formatMenu.addAction(self.blurAct)

def createToolBars(self):

self.fileToolBar = self.addToolBar("File")

self.fileToolBar.addAction(self.openAct)

self.fileToolBar.addAction(self.saveAct)

self.fileToolBar.addAction(self.saveAsAct)

self.editToolBar = self.addToolBar("Edit")

self.editToolBar.addAction(self.rotateCWAct)

self.editToolBar.addAction(self.rotateCCWAct)

self.editToolBar.addAction(self.mirrorAct)

self.editToolBar.addAction(self.flipAct)

self.editToolBar.addAction(self.cropAct)

def grey(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

image = image.convert("L")

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def wb(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

draw = ImageDraw.Draw(image)

width = image.size[0]

height = image.size[1]

pix = image.load()

factor = 100

for i in range(width):

for j in range(height):

a = pix[i, j][0]

b = pix[i, j][1]

c = pix[i, j][2]

S = a + b + c

if (S > (((255 + factor) // 2) \* 3)):

a, b, c = 255, 255, 255

else:

a, b, c = 0, 0, 0

draw.point((i, j), (a, b, c))

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def bright(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

draw = ImageDraw.Draw(image)

width = image.size[0]

height = image.size[1]

pix = image.load()

factor = 50

for i in range(width):

for j in range(height):

a = pix[i, j][0] + factor

b = pix[i, j][1] + factor

c = pix[i, j][2] + factor

if (a < 0):

a = 0

if (b < 0):

b = 0

if (c < 0):

c = 0

if (a > 255):

a = 255

if (b > 255):

b = 255

if (c > 255):

c = 255

draw.point((i, j), (a, b, c))

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def negative(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

draw = ImageDraw.Draw(image)

width = image.size[0]

height = image.size[1]

pix = image.load()

for i in range(width):

for j in range(height):

a = pix[i, j][0]

b = pix[i, j][1]

c = pix[i, j][2]

draw.point((i, j), (255 - a, 255 - b, 255 - c))

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def blur(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

image = image.filter(ImageFilter.BLUR)

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def noise(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

draw = ImageDraw.Draw(image)

width = image.size[0]

height = image.size[1]

pix = image.load()

factor = 70

for i in range(width):

for j in range(height):

rand = random.randint(-factor, factor)

a = pix[i, j][0] + rand

b = pix[i, j][1] + rand

c = pix[i, j][2] + rand

if (a < 0):

a = 0

if (b < 0):

b = 0

if (c < 0):

c = 0

if (a > 255):

a = 255

if (b > 255):

b = 255

if (c > 255):

c = 255

draw.point((i, j), (a, b, c))

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def sepia(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

draw = ImageDraw.Draw(image)

width = image.size[0]

height = image.size[1]

pix = image.load()

depth = 30

for i in range(width):

for j in range(height):

a = pix[i, j][0]

b = pix[i, j][1]

c = pix[i, j][2]

S = (a + b + c) // 3

a = S + depth \* 2

b = S + depth

c = S

if (a > 255):

a = 255

if (b > 255):

b = 255

if (c > 255):

c = 255

draw.point((i, j), (a, b, c))

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def rotateCW(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

image = image.rotate(-90, expand=True)

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def rotateCCW(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

image = image.rotate(90, expand=True)

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def mirror(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

image = image.transpose(Image.FLIP\_LEFT\_RIGHT)

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def flip(self):

if self.img.hasImage():

image = Image.open(self.img.filename)

if image:

image = image.transpose(Image.FLIP\_TOP\_BOTTOM)

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

def crop(self):

if self.img.hasImage():

rect = self.img.zoomStack[-1].getCoords()

image = Image.open(self.img.filename)

if image:

image = image.crop((int(rect[0]), int(rect[1]), int(rect[2]), int(rect[3])))

image.save(self.img.filename)

self.img.loadImageFromFile(self.img.filename)

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

window = MainWindow()

window.show()

sys.exit(app.exec\_())

# Список использованной литературы

[1] Clark A. Pillow (PIL Fork) Documentation. – 2015.

[2] Rossum G. Python tutorial. – 1995.

[3] Summerfield M. Rapid GUI programming with Python and Qt: the definitive guide to PyQt programming. – Pearson Education, 2007.

[4] Саммерфилд М. Программирование на Python 3. – 2011.

[5] Сузи Р. А. Язык программирования Python //М.: Бином. Лаборатория знаний. – 2006.

[6] Изучаем Python, 4-е издание. Марк Лутц. – 2010.

# [7] <https://youtu.be/tKTZoB2Vjuk> - Google Python Class

[8] <https://docs.python.org/3/> - Python Documentations