|  |
| --- |
|  |
| 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»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

**Игра Тетрис с подключением к базе данных SQLite**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бутов И.А.

подпись, дата

Группа БСБО-02-15

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 группа: БСБО-02-15

1 Тема:

**Игра Тетрис с подключением к базе данных SQLite**

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

3 Исходные данные для разработки

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

4 Содержание пояснительной записки:

Титульный лист

Задание

Содержание

Введение

1 Требования к программе или программному изделию

2 Стадии и этапы разработки

3 Составляющие проекта

Заключение

Список использованных источников

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

подпись, дата

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бутов И.А.

подпись, дата

**Содержание**

[Введение 4](#_Toc501742702)

[Основания для разработки 5](#_Toc501742703)

[Назначение разработки 5](#_Toc501742704)

[Требования к программе или программному изделию 5](#_Toc501742705)

[Требования к функциональным характеристикам 5](#_Toc501742706)

[Требования к вычислительным ресурсам и дисплею 10](#_Toc501742707)

[Требования к информационной и программной совместимости 10](#_Toc501742708)

[Стадии и этапы разработки 11](#_Toc501742709)

[Составляющие проекта 11](#_Toc501742710)

[Заключение 12](#_Toc501742711)

[Приложение 1 14](#_Toc501742712)

[tetrisfinal.py 14](#_Toc501742713)

# **Введение**

Игра Тетрис – одна из самых популярных компьютерных игр. Оригинальная игра была разработана и запрограммирована русским программистом Алексеем Пажитновым в 1985 году. С тех пор, Тетрис доступен на почти каждой компьютерной платформе в множестве вариаций.

Создание простой компьютерной игры на PyQt5 – позволит сделать ее кроссплатформенной.

Тетрисом называется игра-головоломка с падающими блоками. В этой игре, мы имеем 7 разных фигур, называемых так: S-фигура, Z-фигура, T-фигура, L-фигура, фигура-линия, фигура "Г", и квадрат. Каждая из этих фигур формируется с помощью четырёх квадратиков. Фигуры падают вниз на доску. Цель игры Тетрис – перемещать и вращать фигуры так, чтобы их приземлилось как можно больше. Если мы сумеем сформировать ряд, ряд разрушается, и мы получаем очки. Игра продолжается до тех пор, пока не будет достигнут верх экрана.**Цель**

Разработать игру «Тетрис», хранящую таблицу рекордов пользователей в базе данных SQLite.

**Задачи**

1. Разработать игру “Тетрис” с применение технологии PyQt5.
2. Разработать пользовательский интерфейс с применение технологии tkinter.
3. Разработать возможность запуска, приостановки, а также с запись рекордов пользователей в базу данных с применение технологии PyQt5.

# **Основания для разработки**

Учебный план по дисциплине «Технология программирования в среде Python», направление бакалавриата «09.03.02», МТУ, 2017.

# **Назначение разработки**

Приложение предназначено для игры в “Тетрис” с сохранением результата в базе данных SQLite.

# **Требования к программе или программному изделию**

## **Требования к функциональным характеристикам**

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

1. Входа в главное меню, при запуске программы

2. Запуск игры при нажатии “Новая игра”

3. Приостановку игры при нажатии на клавишу “P”

4. Выводить окно регистрации в конце игры

5. Сохранять результаты игры в базу данных

6. Открывать таблицу рекордов при нажатии в главном меню на кнопку “Рекорды”

7. Производить выход при нажатии на кнопку “Выход”

При запуске программы отображается главное окно. (рисунок 1)

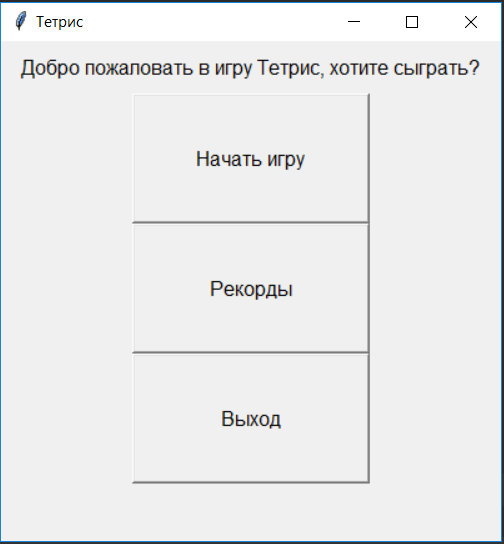


Рисунок 1 - Главное меню

Чтобы запустить игру необходимо нажать на кнопку “Начать игру”. (рисунок 2)

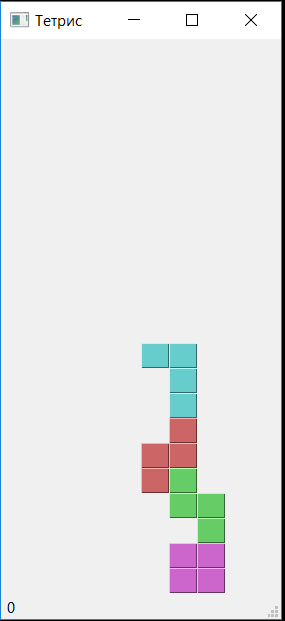
**

Рисунок 2 - Окно игрового процесса

В результате завершения игры пользователю становится доступно окно сохранения результата игры. В котором пользователь может указать своё имя и нажав кнопку “Сохранить результат” сохранить свой рекорд в базе данных SQLite или при нажатии на кнопку “Выйти без сохранения” пользователь выходит из окна регистрации, не сохраняя свой результат. (рисунок 3)

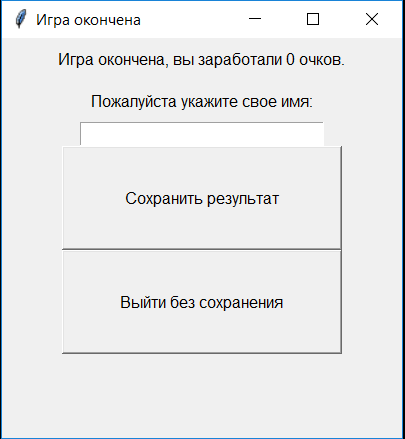


Рисунок 3 - Окно регистрации пользователей

При нажатии на кнопку “Рекорды” в главном меню, пользователю предоставляется окно с таблицей рекордов пяти лучших игроков. (рисунок 4)

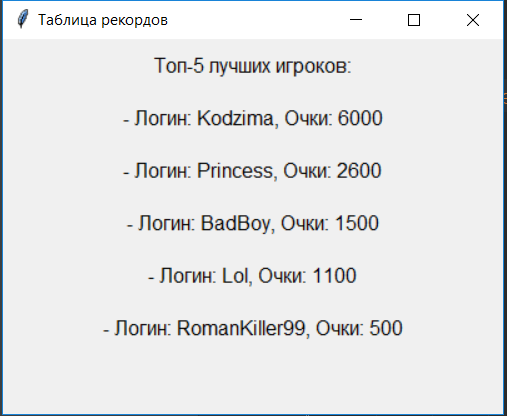


Рисунок 4 - Окно рекордов

При нажатии на кнопку “Выход” в главном меню, пользователю предоставляется окно с подтверждением выхода из игры. (рисунок 5)

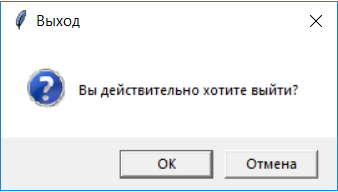


Рисунок 5 - Окно выхода

Во встроенной базе данных SQLite хранится список рекордов. (рисунок 6)

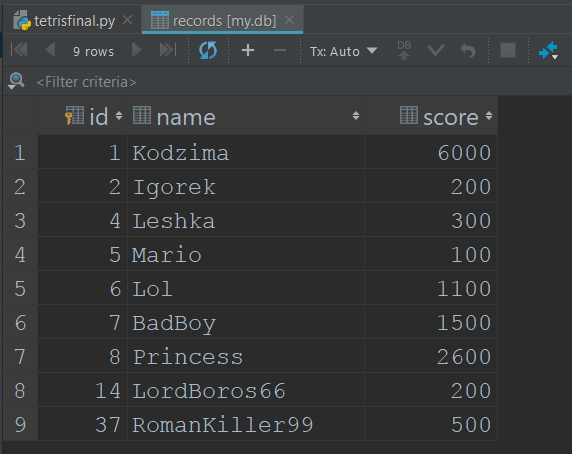


Рисунок 6 - База данных

## **Требования к вычислительным ресурсам и дисплею**

1. Поддержка компьютера с дистрибутивом Fedora старше версии 25.
2. Оперативная память объемом от 1 Гб.
3. Свободное место на SSD или HDD диске объемом не менее 80 Мб.

## **Требования к информационной и программной совместимости**

Изображение Тетриса, написано инструментами рисования в программном инструментарии PyQt5. Пользовательский графический интерфейс(GUI) для данного проекта написан с помощью библиотеки tkinter.

#### **PyQt**

PyQt представляет собой графический интерфейс виджетов инструментарий. Это интерфейс Python для Qt, один из самых мощных и популярных кросс-платформенной библиотеки GUI. PyQt был разработан Riverbank Computing Ltd. Последняя версия PyQt можно скачать с официального сайта - riverbankcomputing.com

PyQt API представляет собой набор модулей, содержащих большое количество классов и функций. В то время как QtCore модуль содержит функциональные возможности без графического интерфейса для работы с файлом и каталогом и т.д., QtGui модуль содержит все графические элементы управления. Кроме того, имеются модули для работы с XML (QtXml), SVG (QtSvg) и SQL (QtSql) и т.д.

PyQt совместим со всеми популярными операционными системами, включая Windows, Linux и Mac OS. Это двойной лицензии, доступен под лицензией GPL, а также коммерческой лицензии.

#### **Tkinter**

Tkinter – это кроссплатформенная библиотека для разработки графического интерфейса на языке Python (начиная с Python 3.0 переименована в tkinter). Tkinter расшифровывается как Tk interface, и является интерфейсом к Tcl/Tk.Tkinter входит в стандартный дистрибутив Python.

# **Стадии и этапы разработки**

1. Анализ предметной области.
2. Разработка схемы базы данных.
3. Разработка технического задания на разработку Программы.
4. Разработка структур данных, предназначенных для представления сущностей в оперативной памяти.
5. Разработка программного кода, реализующего функциональные требования к программе.

Для разработки данного программного обеспечения (ПО) используется интерпретатор Python 3.6.2 и интегрированная среда разработки (IDE) PyCharm 2017, а также библиотеки PyQt5, Tkinter.

# **Составляющие проекта**

Файл tetrisfinal.py содержит основной код программы.

Файл my.db – содержит базу данных.

# **Заключение**

1. Разработана игра “Тетрис” с применение технологии PyQt5.
2. Разработан пользовательский интерфейс с применение технологии tkinter.
3. Разработана возможность запуска, приостановки, а также с запись рекордов пользователей в базу данных с применение технологии PyQt5.
4. Изучены библиотеки PyQt5, Tkinter.

**Список использованных источников**

1. URL: http://www.w3ii.com/ru/pyqt/pyqt\_quick\_guide.html - Документация к библиотеке PyQt (дата обращения 21.12.2017)
2. URL: http://www.w3ii.com/ru/python/python\_gui\_programming.html - Документация к библиотеке Tkinter (дата обращения 21.12.2017)
3. Лутц, М. Программирование на Python. Том 1, 4-е издание. Москва: Символ-Плюс 2011.
4. Алексей Горожанов "PyQT для лингвистов". Том 1, 1-е издание. Москва: Символ-Плюс 2015.
5. Прохоренок Н., Дронов В. - Python 3 и PyQt 5. Том 1, 2-е издание. Санкт-Петербург: Питер 2016.

# **Приложение 1**

Исходный код программы:

## **tetrisfinal.py**

from tkinter import \*  
from tkinter import messagebox  
import sys, random  
from PyQt5.QtWidgets import QMainWindow, QFrame, QDesktopWidget, QApplication  
from PyQt5.QtCore import Qt, QBasicTimer, pyqtSignal  
from PyQt5.QtGui import QPainter, QColor  
import sqlite3  
  
conn = sqlite3.connect('my.db')  
c = conn.cursor()  
  
root = Tk()  
root.geometry("400x400")  
root.title("Тетрис")  
  
def menu():  
 text = Label(width=50, height=2, font='Arial 12', text="Добро пожаловать в игру Тетрис, хотите сыграть?")  
 button1 = Button(width=20, height=5, text= "Начать игру", font='Arial 12', command=lambda: start(), fg="black")  
 button2 = Button(width=20, height=5, text= "Рекорды", fg="black", font='Arial 12', command=lambda: recorder())  
 button3 = Button(width=20, height=5, text= "Выход", fg="black", command=lambda: on\_closing(), font='Arial 12')  
 text.pack(side=TOP)  
 button1.pack(side=TOP)  
 button2.pack(side=TOP)  
 button3.pack(side=TOP)  
  
 def on\_closing():  
 if messagebox.askokcancel("Выход", "Вы действительно хотите выйти?"):  
 root.destroy()  
  
 root.protocol("WM\_DELETE\_WINDOW", on\_closing)  
  
 def recorder():  
 root.destroy()  
 rec = Tk()  
 rec.geometry("400x300")  
 rec.title("Таблица рекордов")  
 text1 = Label(width=50, height=2, font='Arial 12', text="Топ-5 лучших игроков:")  
 text1.pack()  
 c.execute('SELECT \* FROM records ORDER BY score DESC LIMIT 5')  
 rows = c.fetchall()  
 for row in rows:  
 text2 = Label(width=50, height=2, font='Arial 12', text="-"+" Логин: " + str(row[1]) + ", Очки: " + str(row[2]))  
 text2.pack()  
 rec.mainloop()  
  
 c.close()  
 conn.close()  
 sys.exit()  
   
 def start():  
 root.destroy()  
 class Tetris(QMainWindow):  
  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
  
 self.initUI()  
  
 def initUI(self):  
 self.tboard = Board(self)  
 self.setCentralWidget(self.tboard)  
  
 self.statusbar = self.statusBar()  
 self.tboard.msg2Statusbar[str].connect(self.statusbar.showMessage)  
  
 self.tboard.start()  
  
 self.resize(280, 580)  
 self.center()  
 self.setWindowTitle('Тетрис')  
 self.show()  
  
 def center(self):  
 screen = QDesktopWidget().screenGeometry()  
 size = self.geometry()  
 self.move((screen.width() - size.width()) / 2,  
 (screen.height() - size.height()) / 2)  
  
 class Board(QFrame):  
  
 msg2Statusbar = pyqtSignal(str)  
  
 BoardWidth = 10  
 BoardHeight = 22  
 Speed = 300  
  
 def \_\_init\_\_(self, parent):  
 super().\_\_init\_\_(parent)  
  
 self.initBoard()  
  
 def initBoard(self):  
  
 self.timer = QBasicTimer()  
 self.isWaitingAfterLine = False  
  
 self.curX = 0  
 self.curY = 0  
 self.numLinesRemoved = 0  
 self.board = []  
  
 self.setFocusPolicy(Qt.StrongFocus)  
 self.isStarted = False  
 self.isPaused = False  
 self.clearBoard()  
  
 def shapeAt(self, x, y):  
 return self.board[(y \* Board.BoardWidth) + x]  
  
 def setShapeAt(self, x, y, shape):  
 self.board[(y \* Board.BoardWidth) + x] = shape  
  
 def squareWidth(self):  
 return self.contentsRect().width() // Board.BoardWidth  
  
 def squareHeight(self):  
 return self.contentsRect().height() // Board.BoardHeight  
  
 def start(self):  
  
 if self.isPaused:  
 return  
  
 self.isStarted = True  
 self.isWaitingAfterLine = False  
 self.numLinesRemoved = 0  
 self.clearBoard()  
  
 self.msg2Statusbar.emit(str(self.numLinesRemoved))  
  
 self.newPiece()  
 self.timer.start(Board.Speed, self)  
  
 def pause(self):  
  
 if not self.isStarted:  
 return  
  
 self.isPaused = not self.isPaused  
  
 if self.isPaused:  
 self.timer.stop()  
 self.msg2Statusbar.emit("paused")  
  
 else:  
 self.timer.start(Board.Speed, self)  
 self.msg2Statusbar.emit(str(self.numLinesRemoved))  
  
 self.update()  
  
 def paintEvent(self, event):  
  
 painter = QPainter(self)  
 rect = self.contentsRect()  
  
 boardTop = rect.bottom() - Board.BoardHeight \* self.squareHeight()  
  
 for i in range(Board.BoardHeight):  
 for j in range(Board.BoardWidth):  
 shape = self.shapeAt(j, Board.BoardHeight - i - 1)  
  
 if shape != Tetrominoe.NoShape:  
 self.drawSquare(painter,  
 rect.left() + j \* self.squareWidth(),  
 boardTop + i \* self.squareHeight(), shape)  
  
 if self.curPiece.shape() != Tetrominoe.NoShape:  
  
 for i in range(4):  
 x = self.curX + self.curPiece.x(i)  
 y = self.curY - self.curPiece.y(i)  
 self.drawSquare(painter, rect.left() + x \* self.squareWidth(),  
 boardTop + (Board.BoardHeight - y - 1) \* self.squareHeight(),  
 self.curPiece.shape())  
  
 def keyPressEvent(self, event):  
  
 if not self.isStarted or self.curPiece.shape() == Tetrominoe.NoShape:  
 super(Board, self).keyPressEvent(event)  
 return  
  
 key = event.key()  
  
 if key == Qt.Key\_P:  
 self.pause()  
 return  
  
 if self.isPaused:  
 return  
  
 elif key == Qt.Key\_Left:  
 self.tryMove(self.curPiece, self.curX - 1, self.curY)  
  
 elif key == Qt.Key\_Right:  
 self.tryMove(self.curPiece, self.curX + 1, self.curY)  
  
 elif key == Qt.Key\_Down:  
 self.tryMove(self.curPiece.rotateRight(), self.curX, self.curY)  
  
 elif key == Qt.Key\_Up:  
 self.tryMove(self.curPiece.rotateLeft(), self.curX, self.curY)  
  
 elif key == Qt.Key\_Space:  
 self.dropDown()  
  
 elif key == Qt.Key\_D:  
 self.oneLineDown()  
  
 else:  
 super(Board, self).keyPressEvent(event)  
  
 def timerEvent(self, event):  
  
 if event.timerId() == self.timer.timerId():  
  
 if self.isWaitingAfterLine:  
 self.isWaitingAfterLine = False  
 self.newPiece()  
 else:  
 self.oneLineDown()  
  
 else:  
 super(Board, self).timerEvent(event)  
  
 def clearBoard(self):  
  
 for i in range(Board.BoardHeight \* Board.BoardWidth):  
 self.board.append(Tetrominoe.NoShape)  
  
 def dropDown(self):  
  
 newY = self.curY  
  
 while newY > 0:  
  
 if not self.tryMove(self.curPiece, self.curX, newY - 1):  
 break  
  
 newY -= 1  
  
 self.pieceDropped()  
  
 def oneLineDown(self):  
  
 if not self.tryMove(self.curPiece, self.curX, self.curY - 1):  
 self.pieceDropped()  
  
 def pieceDropped(self):  
  
 for i in range(4):  
 x = self.curX + self.curPiece.x(i)  
 y = self.curY - self.curPiece.y(i)  
 self.setShapeAt(x, y, self.curPiece.shape())  
  
 self.removeFullLines()  
  
 if not self.isWaitingAfterLine:  
 self.newPiece()  
  
 def removeFullLines(self):  
  
 numFullLines = 0  
 rowsToRemove = []  
  
 for i in range(Board.BoardHeight):  
  
 n = 0  
 for j in range(Board.BoardWidth):  
 if not self.shapeAt(j, i) == Tetrominoe.NoShape:  
 n = n + 1  
  
 if n == 10:  
 rowsToRemove.append(i)  
  
 rowsToRemove.reverse()  
  
 for m in rowsToRemove:  
  
 for k in range(m, Board.BoardHeight):  
 for l in range(Board.BoardWidth):  
 self.setShapeAt(l, k, self.shapeAt(l, k + 1))  
  
 numFullLines = numFullLines + len(rowsToRemove)  
  
 if numFullLines >= 0:  
 self.numLinesRemoved = self.numLinesRemoved + numFullLines  
 self.ourScore = self.numLinesRemoved \* 100  
 self.msg2Statusbar.emit(str(self.ourScore))  
 self.isWaitingAfterLine = True  
 self.curPiece.setShape(Tetrominoe.NoShape)  
 self.update()  
  
  
  
 def newPiece(self):  
  
 self.curPiece = Shape()  
 self.curPiece.setRandomShape()  
 self.curX = Board.BoardWidth // 2 + 1  
 self.curY = Board.BoardHeight - 1 + self.curPiece.minY()  
 if not self.tryMove(self.curPiece, self.curX, self.curY):  
 self.curPiece.setShape(Tetrominoe.NoShape)  
 self.timer.stop()  
 self.isStarted = False  
 self.msg2Statusbar.emit("Игра окончена")  
  
 root = Tk()  
 root.geometry("400x400")  
 root.title("Игра окончена")  
 v = StringVar()  
 text1 = Label(width=50, height=2, font='Arial 12', text="Игра окончена, вы заработали "+str(self.ourScore)+" очков.")  
 text2 = Label(width=50, height=2, font='Arial 12', text="Пожалуйста укажите свое имя:")  
 mylogin = Entry(textvariable=v, width=30)  
 button1 = Button(width=30, height=5, text="Сохранить результат", font='Arial 12', command=lambda: login(mylogin.get(),self.ourScore))  
 button2 = Button(width=30, height=5, text="Выйти без сохранения", font='Arial 12', command=lambda: closing())  
 text1.pack(side=TOP)  
 text2.pack(side=TOP)  
 mylogin.pack(side=TOP)  
 button1.pack(side=TOP)  
 button2.pack(side=TOP)  
  
 def login(username,userscore):  
 c.execute("INSERT INTO records (name,score) VALUES ('%s','%s')" % (str(username), userscore))  
 conn.commit()  
 messagebox.showinfo("Сохранение", "Ваш результат успешно сохранен.")  
 c.close()  
 conn.close()  
 sys.exit()  
  
 def closing():  
 if messagebox.askokcancel("Выход", "Вы действительно хотите выйти?"):  
 root.destroy()  
 sys.exit()  
  
 root.protocol("WM\_DELETE\_WINDOW", closing)  
  
 root.mainloop()  
  
 def tryMove(self, newPiece, newX, newY):  
  
 for i in range(4):  
  
 x = newX + newPiece.x(i)  
 y = newY - newPiece.y(i)  
  
 if x < 0 or x >= Board.BoardWidth or y < 0 or y >= Board.BoardHeight:  
 return False  
  
 if self.shapeAt(x, y) != Tetrominoe.NoShape:  
 return False  
  
 self.curPiece = newPiece  
 self.curX = newX  
 self.curY = newY  
 self.update()  
  
 return True  
  
 def drawSquare(self, painter, x, y, shape):  
  
 colorTable = [0x000000, 0xCC6666, 0x66CC66, 0x6666CC,  
 0xCCCC66, 0xCC66CC, 0x66CCCC, 0xDAAA00]  
  
 color = QColor(colorTable[shape])  
 painter.fillRect(x + 1, y + 1, self.squareWidth() - 2,  
 self.squareHeight() - 2, color)  
  
 painter.setPen(color.lighter())  
 painter.drawLine(x, y + self.squareHeight() - 1, x, y)  
 painter.drawLine(x, y, x + self.squareWidth() - 1, y)  
  
 painter.setPen(color.darker())  
 painter.drawLine(x + 1, y + self.squareHeight() - 1,  
 x + self.squareWidth() - 1, y + self.squareHeight() - 1)  
 painter.drawLine(x + self.squareWidth() - 1,  
 y + self.squareHeight() - 1, x + self.squareWidth() - 1, y + 1)  
  
 class Tetrominoe(object):  
  
 NoShape = 0  
 ZShape = 1  
 SShape = 2  
 LineShape = 3  
 TShape = 4  
 SquareShape = 5  
 LShape = 6  
 MirroredLShape = 7  
  
 class Shape(object):  
  
 coordsTable = (  
 ((0, 0), (0, 0), (0, 0), (0, 0)),  
 ((0, -1), (0, 0), (-1, 0), (-1, 1)),  
 ((0, -1), (0, 0), (1, 0), (1, 1)),  
 ((0, -1), (0, 0), (0, 1), (0, 2)),  
 ((-1, 0), (0, 0), (1, 0), (0, 1)),  
 ((0, 0), (1, 0), (0, 1), (1, 1)),  
 ((-1, -1), (0, -1), (0, 0), (0, 1)),  
 ((1, -1), (0, -1), (0, 0), (0, 1))  
 )  
  
 def \_\_init\_\_(self):  
  
 self.coords = [[0, 0] for i in range(4)]  
 self.pieceShape = Tetrominoe.NoShape  
  
 self.setShape(Tetrominoe.NoShape)  
 def shape(self):  
 return self.pieceShape  
  
 def setShape(self, shape):  
  
 table = Shape.coordsTable[shape]  
 for i in range(4):  
 for j in range(2):  
 self.coords[i][j] = table[i][j]  
  
 self.pieceShape = shape  
 def setRandomShape(self):  
 self.setShape(random.randint(1, 7))  
  
 def x(self, index):  
 return self.coords[index][0]  
  
 def y(self, index):  
 return self.coords[index][1]  
  
 def setX(self, index, x):  
 self.coords[index][0] = x  
  
 def setY(self, index, y):  
 self.coords[index][1] = y  
  
 def minX(self):  
  
 m = self.coords[0][0]  
 for i in range(4):  
 m = min(m, self.coords[i][0])  
 return m  
 def maxX(self):  
  
 m = self.coords[0][0]  
 for i in range(4):  
 m = max(m, self.coords[i][0])  
 return m  
 def minY(self):  
  
 m = self.coords[0][1]  
 for i in range(4):  
 m = min(m, self.coords[i][1])  
  
 return m  
  
 def maxY(self):  
  
 m = self.coords[0][1]  
 for i in range(4):  
 m = max(m, self.coords[i][1])  
  
 return m  
  
 def rotateLeft(self):  
  
 if self.pieceShape == Tetrominoe.SquareShape:  
 return self  
  
 result = Shape()  
 result.pieceShape = self.pieceShape  
  
 for i in range(4):  
 result.setX(i, self.y(i))  
 result.setY(i, -self.x(i))  
  
 return result  
 def rotateRight(self):  
  
 if self.pieceShape == Tetrominoe.SquareShape:  
 return self  
  
 result = Shape()  
 result.pieceShape = self.pieceShape  
 for i in range(4):  
 result.setX(i, -self.y(i))  
 result.setY(i, self.x(i))  
  
 return result  
  
 if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication([])  
 tetris = Tetris()  
 sys.exit(app.exec\_())  
  
menu()  
root.mainloop()