**Руководство разработчика к приложению “Python Word” – текстовый редактор на языке Python.**

Приложение написано в рамках индивидуального проекта на курсе Проектный семинар “Python в науке о данных”.

Проект выполнила: Грошева Анастасия, группа БИВ214.

МОСКВА 2022

# **Технические требования**

64-битная операционная система Windows, на которую возможна установка интерпретатора Python 3.9 (<https://www.python.org/downloads/>).

# **Версии используемых библиотек и интерпретатора**

Интерпретатор – Python 3.9.

Библиотеки, использованные в написании программы:

|  |  |
| --- | --- |
| **Библиотека** | **Версия** |
| PyQt5 | 5.15.4 |
| sys | 3.9 |

# **Архитектура приложения**

Приложение содержит один модуль main.py, который является основным и выполняет все функции.

# **Листинг основного скрипта**

|  |  |
| --- | --- |
| **Код** | **Описание** |
| **class** Window(QMainWindow, QWidget):      """" Создание основного класса """  **def** \_\_init\_\_(self):          super(Window, self).\_\_init\_\_()          self.setWindowTitle("Текстовый редактор") *# Заголовок окна*          self.setWindowIcon(QtGui.QIcon('lo.png')) *# Иконка окна*          self.setGeometry(420, 250, 500, 350) *# Размер и расположение окна*          self.text = QtWidgets.QTextEdit(self) *# Создание поля для редактирования*          self.setCentralWidget(self.text)          self.createMenuBar()          self.flag = 0          self.flag2 = 0 | Создание основного класса, в котором будут все последующие функции |
| **def** createMenuBar(self):  *# Создание меню*          self.menuBar = QMenuBar(self)          self.setMenuBar(self.menuBar)  *# Создание элементов меню*          self.fileMenu = QMenu("&Файл", self)          self.menuBar.addMenu(self.fileMenu)          self.fileMenu2 = QMenu("&Формат", self)          self.menuBar.addMenu(self.fileMenu2)          self.fileMenu3 = QMenu("&Действия", self)          self.menuBar.addMenu(self.fileMenu3)          self.fileMenu4 = QMenu("&Тема", self)          self.menuBar.addMenu(self.fileMenu4)  *# Создание подпунктов меню*          self.fileMenu.addAction("Новый файл", self.action\_clicked)          self.fileMenu.addAction("Открыть", self.action\_clicked)          self.fileMenu.addAction("Сохранить", self.action\_clicked)          self.fileMenu.addAction("Печать", self.action\_clicked)          self.fileMenu2.addAction("Шрифт", self.action\_clicked)          self.fileMenu2.addAction("Цвет шрифта", self.action\_clicked)          self.background\_color = self.fileMenu2.addMenu("&Цвет фона")          self.fileMenu3.addAction("Копировать", self.action\_clicked)          self.fileMenu3.addAction("Вставить", self.action\_clicked)          self.background\_color.addAction("Черный", self.action\_clicked)          self.background\_color.addAction("Зеленый", self.action\_clicked)          self.background\_color.addAction("Красный", self.action\_clicked)          self.background\_color.addAction("Синий", self.action\_clicked)          self.background\_color.addAction("Оранжевый", self.action\_clicked)          self.background\_color.addAction("Белый", self.action\_clicked)          self.fileMenu4.addAction("Светлая", self.action\_clicked)          self.fileMenu4.addAction("Темная", self.action\_clicked) | Создание меню: пунктов и подпунктов |
| @QtCore.pyqtSlot() *# Аннотация, которая обрабатывает нажатия на пункты меню*  **def** action\_clicked(self):          action = self.sender() *# Получение информации об объекте*  *# Открытие файла*  **if** action.text() == "Открыть":              fname, \_ = QFileDialog.getOpenFileName(self, "Open file", "",                              "Text documents (\*.txt)")  **try**:  **with** open(fname, 'r') **as** f:                      data = f.read()                      self.text.setText(data)    **except** FileNotFoundError:  **return**  *# Сохранение файла*  **elif** action.text() == "Сохранить":              fname, \_ = QFileDialog.getSaveFileName(self, "Save file", "",                              "Text documents (\*.txt)")  **try**:  **with** open(fname, 'w') **as** f:                      save\_text = self.text.toPlainText()                      f.write(save\_text.encode('utf-8').decode('utf-8'))  **except** FileNotFoundError:  **return**  *# Создание нового файла*  **elif** action.text() == "Новый файл":              self.text.setText("")  *# Выбор шрифта*  **elif** action.text() == "Шрифт":              font, ok = QtWidgets.QFontDialog.getFont(self.text.currentFont(), self)  **if** **not** ok:  **return**              self.text.setFont(font)  *# Печать файла*  **elif** action.text() == "Печать":              dlg = QPrintDialog()  **try**:  **if** dlg.exec\_():                      self.editor.print\_(dlg.printer())  **except**:                  QMessageBox.about(self, "Ошибка!", "Невозможно начать печать. "                                                     "**\n**Для продолжения работы закройте это окно.**\t**")  *# Цвет фона*  **elif** action.text() == "Черный":              self.text.setStyleSheet("background-color: #000000;")              self.text.setAutoFillBackground(True)              self.flag = 1              self.flag2 = 1  **elif** action.text() == "Красный":              self.text.setStyleSheet("background-color: #EB0707;")              self.text.setAutoFillBackground(True)              self.flag = 2              self.flag2 = 1  **elif** action.text() == "Зеленый":              self.text.setStyleSheet("background-color: #11A94E;")              self.text.setAutoFillBackground(True)              self.flag = 3              self.flag2 = 1  **elif** action.text() == "Белый":              self.text.setStyleSheet("background-color: #FFFFFF;")              self.text.setAutoFillBackground(True)              self.flag = 4              self.flag2 = 1  **elif** action.text() == "Синий":              self.text.setStyleSheet("background-color: #1749CA;")              self.text.setAutoFillBackground(True)              self.flag = 5              self.flag2 = 1  **elif** action.text() == "Оранжевый":              self.text.setStyleSheet("background-color: #FA9C2F;")              self.text.setAutoFillBackground(True)              self.flag = 6              self.flag2 = 1  *# Цвет шрифта*  **elif** action.text() == "Цвет шрифта":  **if** self.flag2 == 0:                  QMessageBox.about(self, "Ошибка!",                                    "Сначала выберите цвет фона.**\t**")  **else**:                  col = QColorDialog.getColor()  **if** col.isValid():  **if** self.flag == 6:                          self.text.setStyleSheet("QWidget { color: %s; background-color: #FA9C2F}" % col.name())  **elif** self.flag == 5:                          self.text.setStyleSheet("QWidget { color: %s; background-color: #1749CA}" % col.name())  **elif** self.flag == 4 **or** self.flag == 0:                          self.text.setStyleSheet("QWidget { color: %s; background-color: #FFFFFF}" % col.name())  **elif** self.flag == 3:                          self.text.setStyleSheet("QWidget { color: %s; background-color: #11A94E}" % col.name())  **elif** self.flag == 2:                          self.text.setStyleSheet("QWidget { color: %s; background-color: #EB0707}" % col.name())  **elif** self.flag == 1:                          self.text.setStyleSheet("QWidget { color: %s; background-color: #000000}" % col.name())  **else**:                          self.text.setStyleSheet("QWidget { color: %s; background-color: #FFFFFF}" % col.name())  *# Определение темы*  **elif** action.text() == "Темная":              self.text.setStyleSheet("QWidget { color: #35E917; background-color: #111111}")  **elif** action.text() == "Светлая":              self.text.setStyleSheet("QWidget { color: #000000; background-color: #FFFFFF}")  *# Копирование текста*  **elif** action.text() == "Копировать":              save\_text = self.text.toPlainText()              clipboard = QApplication.clipboard()              clipboard.setText(save\_text)  *# Вставка текста*  **elif** action.text() == "Вставить":              textt = QApplication.clipboard().text()              self.text.insertPlainText(textt) | Обработка нажатий на кнопки меню |
| **def** application():      app = QApplication(sys.argv)      window = Window()      window.show()      sys.exit(app.exec\_()) | Создание основного метода, который позволит запустить программу |
| **if** \_\_name\_\_ == "\_\_main\_\_":      application() | Запуск программы |