**Работа с таблицами в PyQt5**

***QTableWidget***

***Записать все методы в тетрадь!***

**QTableWidget** состоит из клеток, каждая клетка — экземпляр класса **QTableWidgetItem**.

Cоздадим форму, которая содержит **QTableWidget** и **QPushButton** После этого, сохраните форму как **table.ui** и конвертируйте дизайн в **table.py** (см. рис. 1.).

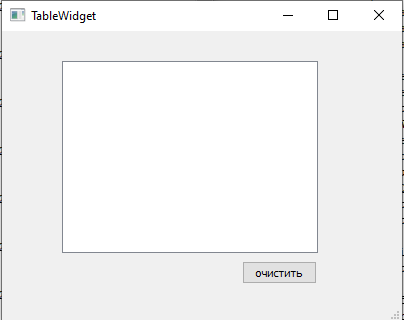


Рисунок 1. Дизайн программы table.py

Чтобы добавлять ряды в **QTableWidget** - метод **setRowCount**().

Для внесения столбцов в **QTableWidget** - метод **setColumnCount**().

Возьмите «заготовку» main.py. Импортируйте виджеты из файла table.py.

В класс MyWin добавьте код, для добавления строк и столбцов в виджет TableWidget (см.рис.2).



Рисунок 2. Добавление строк и столбцов в таблицу

Добавьте функцию очистки содержимого tableWidget по нажатию кнопки «Очистить» (см. рис.3)



Рисунок 3. Очистка ячеек tableWidget

Чтобы заполнить tableWidget программно, нужно использовать метод **setItem()** для каждого объекта tableWidgetItem.

Создадим список трех кортежей Python;

Перебираем и получаем каждый кортеж в списке, для заполнения клеток таблицы, при помощи метода **setItem**() (см. рис.4.)

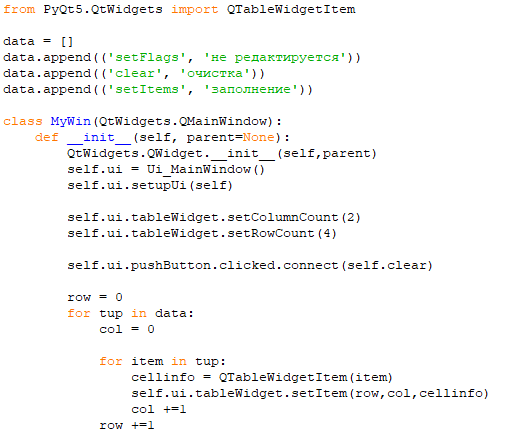


Рисунок 4. Добавление заполнения таблицы

Чтобы сделать **QTableWidget** не редактируемым, можно использовать метод **setFlags**() (см. рис.5).

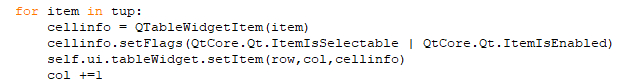


Рисунок 5. tableWidget не доступен для редактирования

***Заголовки строк и столбцов***

Для того чтобы задать текст заголовкам **QTableWidget**, вы можете использовать метод **setHorizontalHeaderLabels**() (см. рис. 6):

Таким же образом, можно **менять заголовок ряда**, при помощи метода **setVerticalHeaderLabels**():

***Сортировка***

Отсортировать QTableWidget - метод **setSortingEnabled**() (см. рис. 6)

Сортировка только для определенного столбца – метод **sortByColumn**(), и указать индекс столбца и порядок сортировки

Метод **sortItems**() для сортировки **QTableWidget** в возрастающем порядке по умолчанию:

**self.ui.tableWidget.sortItems(0)**

self.ui.tableWidget.sortItems(1, QtCore.Qt.**DescendingOrder**) – по убыванию

self.ui.tableWidget.sortItems(1, QtCore.Qt.**AscendingOrder**) – по возрастанию

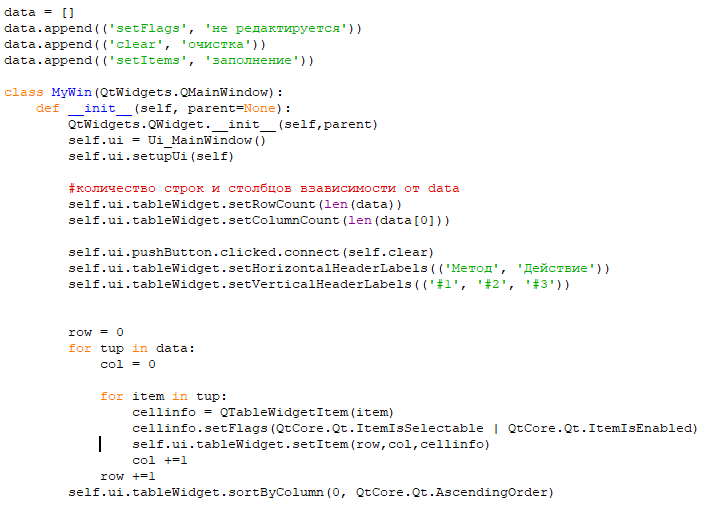


Рисунок 6. Код программы tablemain.py

***QComboBox в QTableWidget***

Чтобы добавить **QComboBox** внутрь **QTableWidgetItem**, можно использовать метод **setCellWidget**() :

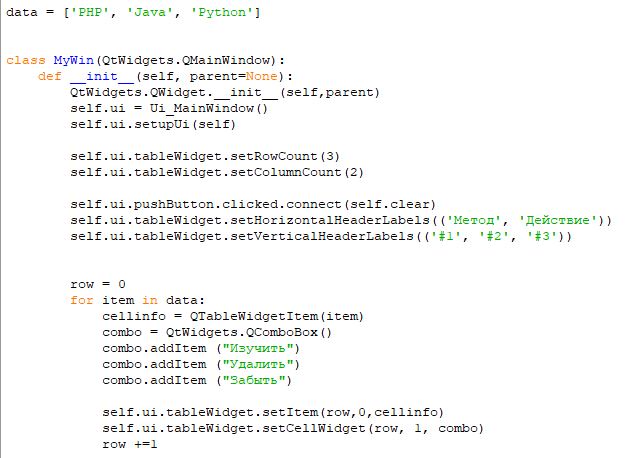


Рисунок 7. Добавление ComboBox в tableWidget

***QProgressBar в QTableWidget***

ProgressBar – виджет, который мы используем, когда обрабатываем продолжительные задачи (загрузка).

**progress = QtWidgets.QProgressBar()** – создаем новый виджет ProgressBar

**progress.setMinimum(0)** – задаем минимальное значение

**progress.setMaximum(100)** – задаем максимальное значение

**progress.setFormat**('{0:.2f}%'.format(item[1])) – устанавливаем формат вывода

**setCellWidget –** добавитьв ячейку (значение или виджет) (см.рис.8)



Рисунок 8. Добавление ProgressBar в ячейку таблицы

**Самостоятельно по вариантам:**

1. Добавить еще один столбец в программу и вставьте виджет QСheckbox.
2. Заполнение ячеек из lineEdit
3. Заполнение ячеек из файла
4. Добавление строк и столбцов по нажатию кнопки
5. Сумма значений ячеек по строке
6. Сумма значений ячеек по столбцу
7. Сохранение значений из таблицы в файл
8. Сумма четных ячеек и не четных в edit1 и edit2