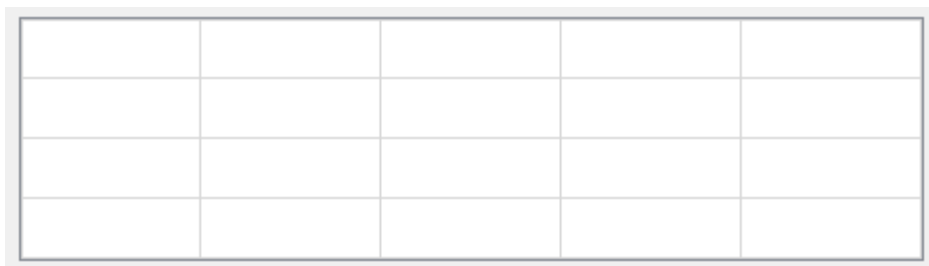



ЛЕКЦИЯ 5 – РАБОТА С ТАБЛИЦАМИ И ТЕКСТОМ В PYQT5

5.1 QWidget

Таблица



№	Свойство		Назначение
1.	objectName	tableWidget	Имя таблицы
2.	enabled	<input checked="" type="checkbox"/>	Доступность
3.	geometry [(10, 10), 452 x 122] X 10 Y 10 Width 452 Height 122 sizePolicy [Expanding, Expanding, 0, 0] Horizontal Policy Expanding Vertical Policy Expanding Horizontal Stretch 0 Vertical Stretch 0 minimumSize 441 x 0 Width 441 Height 0 maximumSize 455 x 220 Width 455 Height 220		Размеры таблицы
4.	font A [Segoe UI, 10] Family Segoe UI Point Size 10		Шрифт, размер шрифта
5.	cursor  Arrow		Вид курсора
6.	QTableView showGrid <input checked="" type="checkbox"/> gridStyle SolidLine sortingEnabled <input type="checkbox"/> wordWrap <input checked="" type="checkbox"/> cornerButtonEnabled <input checked="" type="checkbox"/>		Вид сетки, сортировка столбцов, перенос слов
7.	QTableWidget rowCount 4 columnCount 5		Количество строк и столбцов

8.	Header	
	horizontalHeaderVisible	<input type="checkbox"/> ←
	horizontalHeaderCascadingSectionResizes	<input type="checkbox"/>
	horizontalHeaderDefaultSectionSize	90 ←
	horizontalHeaderHighlightSections	<input checked="" type="checkbox"/>
	horizontalHeaderMinimumSectionSize	55
	horizontalHeaderShowSortIndicator	<input type="checkbox"/>
	horizontalHeaderStretchLastSection	<input type="checkbox"/>
	verticalHeaderVisible	<input type="checkbox"/> ←
	verticalHeaderCascadingSectionResizes	<input type="checkbox"/>
	verticalHeaderDefaultSectionSize	30 ←
	verticalHeaderHighlightSections	<input type="checkbox"/>
	verticalHeaderMinimumSectionSize	25
	verticalHeaderShowSortIndicator	<input type="checkbox"/>
	verticalHeaderStretchLastSection	<input type="checkbox"/>

Шапки таблицы,
ширина и высота
ячейки

Запрещаем редактирование таблицы
self.tableWidget.setEditTriggers(QAbstractItemView.NoEditTriggers)

```
...
row = 0
col = 0
while row < self.tableWidget.rowCount():
    while col < self.tableWidget.columnCount():
        random_num = randint(0, 101)
        self.tableWidget.setItem(row, col, QTableWidgetItem(str(random_num)))
        item = self.tableWidget.item(row, col).text()
        col += 1
    row += 1
    col = 0
...
```

5.1.1 Практический пример

Работа с визуальными табличными данными в Python

56	11	14	89	48
70	36	54	44	24
72	78	55	39	70
16	59	6	42	63

Максимальный элемент: 89.0 [4;1]
Сумма единиц перед максимальным элементом:

Если перед максимальным элементом таблицы расположены все единицы, то заменить максимальный элемент таблицы на количество этих единиц

Заполнить случ. числами

Выполнить задание

```
import sys

from random import randint
from PyQt5 import QtGui
from PyQt5.QtWidgets import *
from PyQt5.uic import loadUi

class Main(QDialog):
    def __init__(self):
        super(Main, self).__init__()
        loadUi('uis/main.ui', self) # загрузка формы в py-скрипт

        self.setWindowTitle('Работа с визуальными табличными данными в Python')
        self.setWindowIcon(QtGui.QIcon('images/logo.png'))

        self.btn_random_number.clicked.connect(self.fill_random_numbers)
        self.btn_solve.clicked.connect(self.solve)

    def fill_random_numbers(self):
        """
        заполняем таблицу случайными числами
        :return:
        """
        row = 0
        col = 0

        # заполняем таблицу случайными числами
        while row < self.tableWidget.rowCount():
            while col < self.tableWidget.columnCount():
                random_num = randint(0, 101)
                self.tableWidget.setItem(row, col, QTableWidgetItem(str(random_num)))
                col += 1
            row += 1
            col = 0

        # находим максимальное число и его координаты
        # [0] - максимальное число, [1] - строка максимума, [2] - столбец максимума
        list_information_max_num = find_max(self.tableWidget)

        if not list_information_max_num:
            self.label_error.setText('Введены неправильные данные!')
        else:
            # выводим на экран информацию о расположении максимального числа
            self.label_max_el.setText(
                'Максимальный элемент: ' + str(list_information_max_num[0]) + ' [' +
                str(list_information_max_num[1]) + ';' +
                str(list_information_max_num[2]) + ']'
            )

    def solve(self):
        list_information_max_num = find_max(self.tableWidget)

        if not list_information_max_num:
            self.label_error.setText('Введены некорректные данные!')
        else:
            self.label_max_el.setText(
                'Максимальный элемент: ' + str(list_information_max_num[0]) + ' [' +
                str(list_information_max_num[1]) + ';' +
                str(list_information_max_num[2]) + ']'
            )

        # -*- решение задания -*-
        row = 0
        col = 0

        number_of_units = 0 # количество единиц, стоящих перед нашим числом
```

```

        flag = False

        while row < self.tableWidget.rowCount():
            while col < self.tableWidget.columnCount():
                item = self.tableWidget.item(row, col).text()
                # три случая:
                # 1) элемент равен единице
                # 2) элемент равен максимальному числу
                #    2.1) максимальный элемент располагается в 1-ой ячейке и перед
                #        ним нет ячеек
                # 3) элемент равен иному числу
                if float(item) == 1:
                    number_of_units += 1
                    col += 1
                elif float(item) == list_information_max_num[0]:
                    if row == 0 and col == 0:
                        self.tableWidget.setItem(row, col,
QTableWidgetItem(str(item)))
                    else:
                        self.tableWidget.setItem(row, col,
QTableWidgetItem(str(number_of_units)))

                    self.label_sum.setText('Сумма единиц перед максимальным
элементом: ' + str(number_of_units))
                    flag = True
                    break
                else:
                    self.label_sum.setText('Сумма единиц перед максимальным
элементом: 0')

                    flag = True
                    break

            if flag:
                break

            row += 1
            col = 0

        self.label_error.setText('')

def find_max(table_widget):
    """
    находим максимальное число из таблицы и его координаты
    :param table_widget: таблица
    :return: [max_num, row_max_number, col_max_number], если выкинуто исключение,
            то None
    """

    row_max_number = 0 # номер строки, в котором находится максимальное число
    col_max_number = 0 # номер столбца, в котором находится максимальное число
    max_num = float(table_widget.item(row_max_number, col_max_number).text()) #
    Максимальное значение

    row = 0
    col = 0

    try:
        while row < table_widget.rowCount():
            while col < table_widget.columnCount():
                number = float(table_widget.item(row, col).text())
                if number > max_num:
                    max_num = number
                    row_max_number = row

```

```

        col_max_number = col
        col += 1
        row += 1
        col = 0
    return [max_num, row_max_number, col_max_number]
except Exception:
    return None

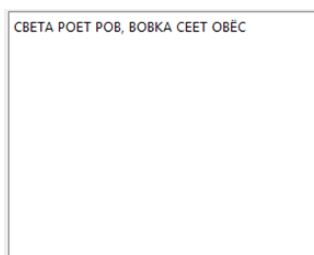
def main():
    app = QApplication(sys.argv)
    window = Main()
    window.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()



```

5.2 QTextEdit

Компонент для ввода и отображения многострочного текста



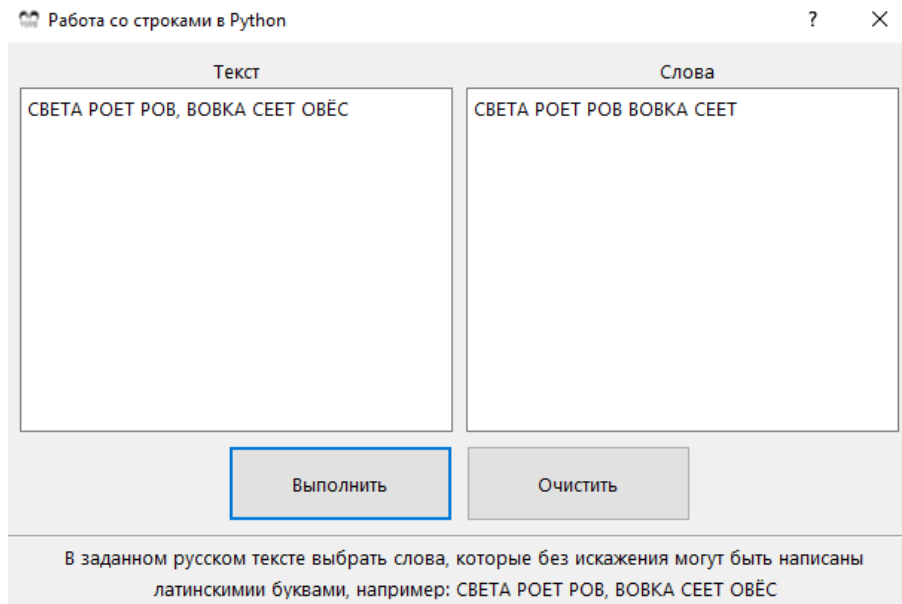
№	Свойство		Назначение
1.	objectName	textEdit_text	Имя компонента
2.	enabled <input checked="" type="checkbox"/>		Доступность
3.	geometry	[(10, 30), 291 x 231]	Размеры компонента
	X	10	
	Y	30	
	Width	291	
	Height	231	
	sizePolicy	[Expanding, Expanding, 0, 0]	
	Horizontal Policy	Expanding	
	Vertical Policy	Expanding	
	Horizontal Stretch	0	
	Vertical Stretch	0	
	minimumSize	0 x 0	
	Width	0	
	Height	0	
	maximumSize	16777215 x 16777215	
	Width	16777215	
	Height	16777215	
4.	font	A [Segoe UI, 10]	Шрифт, размер шрифта
	Family	Segoe UI	
	Point Size	10	

5.	cursor	 Arrow	Вид курсора
6.	<div><div>QTextEdit</div><div><div>> autoFormatting</div>AutoNone</div><div><div>tabChangesFocus</div><input type="checkbox"/></div><div><div>> documentTitle</div></div><div><div>undoRedoEnabled</div><input checked="" type="checkbox"/></div><div><div>lineWrapMode</div>WidgetWidth</div><div><div>lineWrapColumnOrWidth</div>0</div><div><div>readOnly</div><input type="checkbox"/></div><div><div>> html</div><!DOCTYPE HTML PUB</div><div><div>overwriteMode</div><input type="checkbox"/></div><div><div>tabStopWidth</div>80</div><div><div>acceptRichText</div><input checked="" type="checkbox"/></div><div><div>cursorWidth</div>1</div><div><div>> textInteractionFlags</div>TextSelectableByMouse</div><div><div>> placeholderText</div></div></div> <div></div>		Текст в компоненте

Текст в компоненте

```
self.textEdit_text.clear() # Очистка текста в компоненте
self.textEdit_words.insertPlainText("Hi!") # Добавление текста в компонент
text = self.textEdit_text.toPlainText() # Чтение текста из компонента
```

5.2.1 Практический пример



```
import re
import sys

from PyQt5 import QtGui
from PyQt5.QtWidgets import *
from PyQt5.uic import loadUi
```

```
class Main(QDialog):
    def __init__(self):
        super(Main, self).__init__()
        loadUi('uis/main.ui', self)

        self.setWindowTitle('Работа со строками в Python')
        self.setWindowIcon(QtGui.QIcon('images/logo.png'))

        self.btn_solve.clicked.connect(self.solve)
        self.btn_clear.clicked.connect(self.clear)

    def solve(self):
        text = self.textEdit_text.toPlainText() # получаем наш текст

        # для строки "СВЕТА РОЕТ РОВ, ВОВКА СЕЕТ ОВЁС"
        # получится список: ['СВЕТА', 'РОЕТ', 'РОВ', 'ВОВКА', 'СЕЕТ']
        #
        # \b -- ищет границы слов
        # [АВСТРХОНКМУЕ] -- описывает что ищем
        # + -- говорит, что искать нужно минимум от 1 символа
        for word in re.findall(r'\b[АВСТРХОНКМУЕ]+\b', text):
            self.textEdit_words.insertPlainText(word + " ")

    def clear(self):
        self.textEdit_text.clear()
        self.textEdit_words.clear()

def main():
    app = QApplication(sys.argv)
    window = Main()
    window.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    main()
```