Министерство науки и высшего образования Российской Федерации
Калужский филиал
федерального государственного бюджетного образовательного
учреждения высшего образования
**«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»**
(КФ МГТУ им. Н.Э. Баумана)

Н.И. Пчелинцева, И.И. Ерохин

Домашняя работа по дисциплине

«ПЕРСПЕКТИВНЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ»:
учебное пособие

Калуга – 2023

УДК 004.42

Рецензенты:
зав. кафедрой «Информационные системы и сети» (ИУК2)
к.т.н., доцент И.В. Чухраев

руководитель проектного департамента
АО «Калуга-Астрал» Н.И. Мезенцев

Утверждено Методической комиссией КФ МГТУ им. Н.Э. Баумана
(протокол № X от XX.XX.2023 г., рег. номер XXX/МК2023-XX)

Пчелинцева Н.И., Ерохин И.И.

Учебное пособие по выполнению домашних работ по дисциплине «Перспективные языки программирования»: учебное пособие / Н.И. Пчелинцева, И.И. Ерохин – Калуга: КФ МГТУ им. Н.Э. Баумана, 2023. – 117 с.

В учебном пособии приведены теоретические сведения, примеры и задания для выполнения домашних работ по дисциплине «Перспективные языки программирования».

Учебное пособие предназначено для студентов КФ МГТУ им. Н.Э. Баумана, обучающихся по направлению подготовки 09.03.04 «Программная инженерия».

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Настоящие методические указания составлены в соответствии с программой проведения лабораторных работ по курсу «Высокоуровневое программирование» на кафедре «Программное обеспечение ЭВМ, информационные технологии» факультета «Информатика и управление» Калужского филиала МГТУ им. Н.Э. Баумана.

Требования к программному обеспечению:

1. Интерпретатор Python версии 3.8 и старше
2. PyCharm или аналог
3. Библиотека PyQt

ДОМАШНЯЯ РАБОТА №1
РАЗРАБОТКА ГРАФИЧЕСКОГО ИНТЕРФЕЙСА С ПОМОЩЬЮ БИБЛИОТЕКИ PYQT ЯЗЫКА PYTHON

Целью выполнения домашней работы является приобретение практических навыков разработки графического интерфейса с помощью библиотеки PyQt средствами языка Python.

Основными задачами выполнения домашней работы являются:

Задачи:

1. Ознакомиться с конструкцией библиотеки PyQt;
2. Изучить способы создания мини-приложений с помощью PyQt;
3. Изучить возможности данной платформы;
4. Закрепить полученные в ходе выполнения домашней работы навыки.

Результатами работы являются:

1. Реализация разработанных алгоритмов на языке программирования Python;
2. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Общие сведения о PyQt

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. Существует 2 версии: PyQt5, поддерживающий Qt 5, и PyQt4, поддерживающий Qt4. PyQt распространяется под лицензиями GPL (2 и 3 версии) и коммерческой.

PyQt практически полностью реализует возможности Qt. А это более 600 классов, более 6000 функций и методов, включая:

- Существующий набор виджетов графического интерфейса;
- Стили виджетов;
- Доступ к базам данных с помощью SQL (ODBC, MySQL, PostgreSQL , Oracle);
- QScintilla, основанный на Scintilla (виджет текстового редактора);
- Поддержку интернационализации (i18n);
- Парсер XML;
- Поддержку SVG;
- Интеграцию с WebKit, движком рендеринга HTML;
- Поддержку воспроизведения видео и аудио.

PyQt5 является одним из наиболее часто используемых модулей для создания GUI приложений в Python, и это связанно с его простотой, о которой вы узнаете далее.

PyQt также включает в себя Qt Designer (Qt Creator) — дизайнер графического интерфейса пользователя. Программа pyuic генерирует Python код из файлов, созданных в Qt Designer. Это делает PyQt очень

полезным инструментом для быстрого прототипирования. Кроме того, можно добавлять новые графические элементы управления, написанные на Python, в Qt Designer.

Установка PyQt5

Существует две версии PyQt5: коммерческая и бесплатная версия GPL, которой мы и будем пользоваться.

Есть несколько способов установки PyQt5:

- Установка PyQt5 через pip;
- Установка PyQt5 из исходников на Linux.

Установка PyQt5 через pip

Чтобы установить PyQt5 при помощи pip, выполните следующую команду:

```
sudo pip3 install PyQt5
```

Чтобы убедиться в том, что установка прошла успешно, запустите следующий код:

```
import PyQt5
```

Если не возникло ни одной ошибки, это значит, что вы успешно установили PyQt5. В случае, если ошибки возникли, возможно это связанно с тем, что вы используете версию Python, которая не поддерживается.

Установка PyQt5 из исходников на Linux

Для установки PyQt5 из исходника, вам нужно сделать следующее:

1. Установить SIP;
2. Скачать исходник PyQt5;
3. Настроить и установить.

Как вы возможно знаете, PyQt5 связывает Python с популярной библиотекой Qt, которая написана на C++.

Инструмент, который создает эту связь, называется SIP. Так что для установки PyQt5 из исходника, вам для начала нужно установить SIP. Для установки SIP, запустите следующую команду:

```
sudo pip3 install PyQt5-sip
```

Теперь вы можете загрузить и установить исходник PyQt5.

Скачать исходник PyQt5 можно отсюда: https://www.riverbankcomputing.com/software/pyqt/download5

Перейдя по ссылке, вы должны будете в открывшейся вкладке выбрать PyQt5_gpi511.3tar.gz (Рис.1).

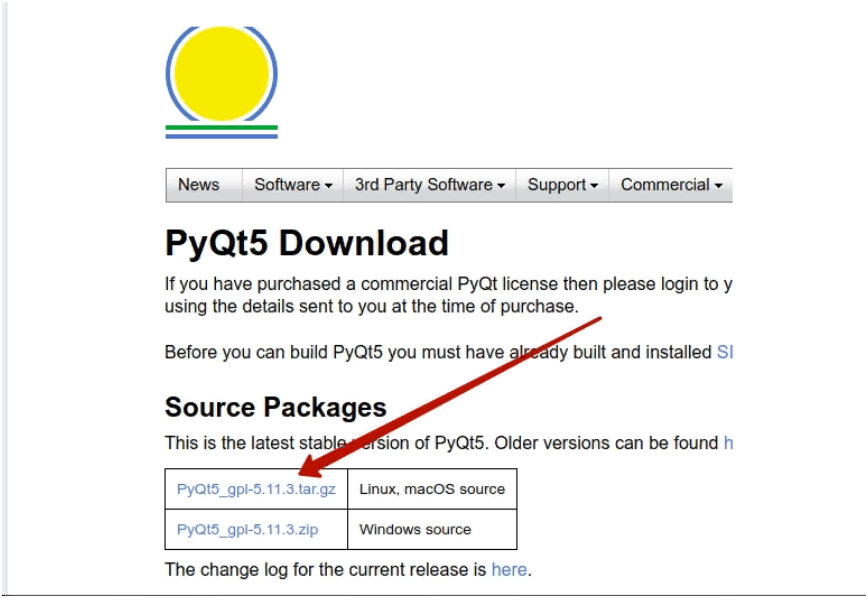


Рис.1.Скачивание исходника PyQt5для Linux

После распаковки сжатого исходника, можно запустить следующие команды внутри корня папки:

```
sudo python3 configure.py
sudo make
sudo make install
```

Установка PyQt5 из исходников на Windows

Перейдя по ссылке, указанной в предыдущем пункте, можно скачать также исходники для Windows (Рис.2).

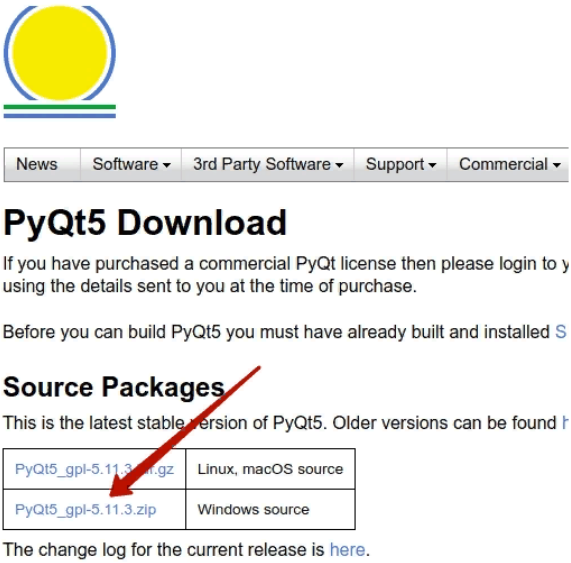


Рис.2.Скачивание исходника PyQt5 для Windows

Скачивайте и распакуйте архив с сайта, указанного выше .Так как SIP требует компилятор GCC, вам нужно установить MinGW, который является портом Windows для компилятора Linux, GCC.

Единственное, что нужно поменять — это момент конфигурации. Вам нужно сообщить Python о платформе.

Это можно выполнить следующим образом:

```
python configure.py --platform win32-g++
```

```
make
make install
```

Установка PyQt5 Designer

Есть два способа создания GUI приложений при помощи PyQt5:

1. Дизайн виджетов при помощи кода;
2. Использование PyQt5 Designer.

Мы будем использовать PyQt5 Designer, который упрощает процесс на столько, что вы можете выполнить большой объем работы за секунды. Дизайнер PyQt5 поставляется вместе с набором инструментов. Для его установки, вам нужно установить эти инструменты.

```
$ pip3 install PyQt5-tools
```

После удачной установки, вы можете найти дизайнер PyQt5 здесь:

```
C:\Program Files\Python36\Lib\site-packages\PyQt5-tools\
```

Если вы установили только для своего текущего пользовательского аккаунта, вы найдете дизайнер PyQt5 вот здесь:

```
C:\Users\PythonUser\AppData\Local\Programs\Python\Python36-32\Lib\site-packages\ PyQt5-tools\
```

Вы можете создать короткий путь для него, вместо того, чтобы постоянно переходить к этому расположению для запуска дизайнера PyQt5. Откройте designer.exe и увидите диалоговое окно, спрашивающее о том, какую форму шаблона вы предпочитаете (Рис. 3):

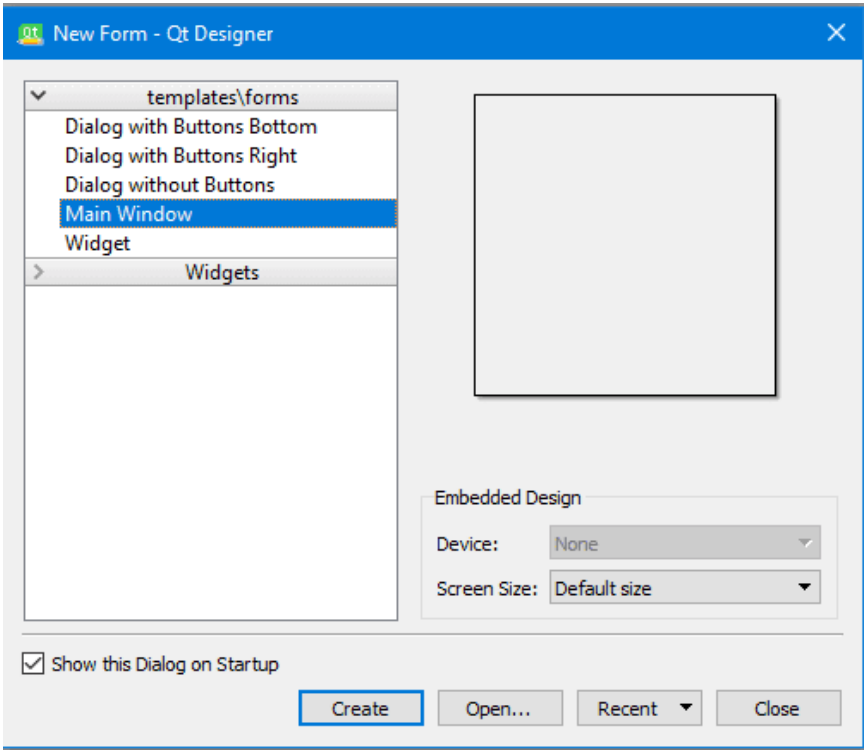


Рис. 3.Открывшееся диалоговое окно с различными шаблонами

Существует пять видов доступных шаблонов:

1. Диалоговое окно с кнопками внизу (Dialog with Buttons Bottom): создает форму с кнопками OK и Cancel в правом нижнем углу формы.
2. Диалоговое окно с кнопками справа (Dialog with Buttons Right): создает форму с кнопками OK и Cancel в верхнем правом углу формы.
3. Диалоговое окно без кнопок (Dialog without Buttons): создает пустую форму;
4. Главное окно (Main Window): создает окно с панелью меню и набором инструментов. Унаследовано из QmainWindow;

5. Виджет (Widget): создает виджет, наследуемый из класса Qwidget. Отличается от диалоговых шаблонов тем, что они наследуются из класса QBialog.

Основные инструменты, необходимые работы

QPen (ручка) – это элементарный графический объект. Он используется, чтобы рисовать линии, кривые и контуры фигур.

```
import sys
from PyQt5.QtWidgets import QWidget, QApplication
from PyQt5.QtGui import QPainter, QColor, QPen
from PyQt5.QtCore import Qt

class Example(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()
    def initUI(self):
        self.setGeometry(300, 300, 280, 270)
        self.setWindowTitle('Pen styles')
        self.show()
    def paintEvent(self, e):
        qp = QPainter()
        qp.begin(self)
        self.drawLines(qp)
        qp.end()
    def drawLines(self, qp):
        pen = QPen(Qt.black, 2, Qt.SolidLine)
        qp.setPen(pen)
        qp.drawLine(20, 40, 250, 40)
        pen.setStyle(Qt.DashLine)
        qp.setPen(pen)
        qp.drawLine(20, 80, 250, 80)
        pen.setStyle(Qt.DashDotLine)
        qp.setPen(pen)
        qp.drawLine(20, 120, 250, 120)
        pen.setStyle(Qt.DotLine)
        qp.setPen(pen)
        qp.drawLine(20, 160, 250, 160)
```

```
        pen.setStyle(Qt.DashDotDotLine)
        qp.setPen(pen)
        qp.drawLine(20, 200, 250, 200)
        pen.setStyle(Qt.CustomDashLine)
        pen.setDashPattern([1, 4, 5, 4])
        qp.setPen(pen)
        qp.drawLine(20, 240, 250, 240)
if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

Реализацию различных графических_объектов можно увидеть на Рис. 4:

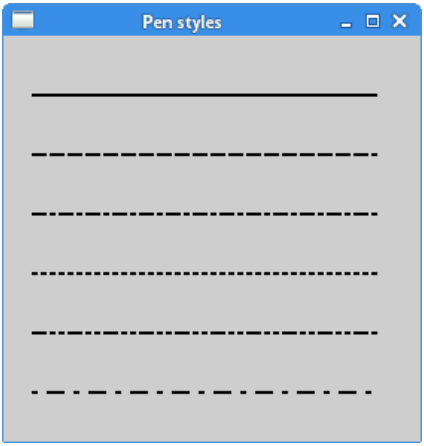


Рис.4.Рисование линий, кривых и контуров фигур

Цвет – это объект, представляющий собой комбинацию красного, зелёного и синего (RGB) значений интенсивности. Корректные значения RGB находятся в диапазоне от 0 до 255. Мы можем определить цвет разными способами. Самый распространённый – десятичные или шестнадцатеричные значения RGB. Мы также можем использовать значения RGBA, которые обозначают красный,

зелёный, синий и альфа-канал. Значение альфа 255 определяет полную непрозрачность, 0 – полная прозрачность, т.е. цвет невидим.

```
import sys
from PyQt5.QtWidgets import QWidget, QApplication
from PyQt5.QtGui import QPainter, QColor,QBrush

class Example(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):

        self.setGeometry(300, 300, 350, 100)
        self.setWindowTitle('Colours')
        self.show()

    def paintEvent(self, e):

        qp = QPainter()
        qp.begin(self)
        self.drawRectangles(qp)
        qp.end()

    def drawRectangles(self, qp):

        col = QColor(0, 0, 0)
        col.setNamedColor('#d4d4d4')
        qp.setPen(col)

        qp.setBrush(QColor(200, 0, 0))
        qp.drawRect(10, 15, 90, 60)

        qp.setBrush(QColor(255, 80, 0, 160))
        qp.drawRect(130, 15, 90, 60)

        qp.setBrush(QColor(25, 0, 90, 200))
        qp.drawRect(250, 15, 90, 60)

if __name__ == '__main__':
```

```
app = QApplication(sys.argv)
ex = Example()
sys.exit(app.exec_())
```

Результат запуска, показывающая работу с цветом(Рис.5):

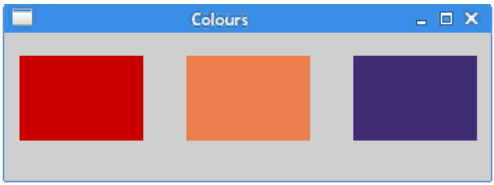


Рис.5.Работа с цветом

Точка – это самый простой графический объект, который может быть нарисован. Это маленькое пятнышко в окне.

```
import sys, random
from PyQt5.QtWidgets import QWidget, QApplication
from PyQt5.QtGui import QPainter, QColor, QPen
from PyQt5.QtCore import Qt

class Example(QWidget):

    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setGeometry(300, 300, 280, 170)
        self.setWindowTitle('Points')
        self.show()

    def paintEvent(self, e):

        qp = QPainter()
        qp.begin(self)
        self.drawPoints(qp)
```

```
        qp.end()

    def drawPoints(self, qp):
        qp.setPen(Qt.red)
        size = self.size()
        for i in range(1000):
            x = random.randint(1, size.width()-1)
            y = random.randint(1, size. height()-1)
            qp.drawPoint(x, y)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

Реализация представлена ниже (Рис.6):

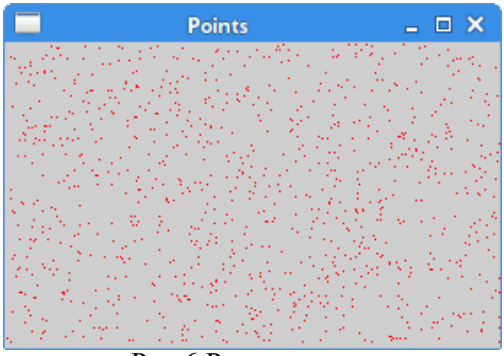


Рис.6.Рисование точек

Следующие код необходим для рисования текста.

```
import sys
from PyQt5.QtWidgets import QWidget, QApplication
from PyQt5.QtGui import QPainter, QColor, QFont
from PyQt5.QtCore import Qt

class Example(QWidget):
    def __init__(self):
```

```
        super().__init__()
        self.initUI()
    def initUI(self):
        self.text       =       u'\u041b\u0435\u0432
\u041d\u0438\u043a\u043e\u043b\u0430\
\u0435\u0432\u0438\u0447
\u0422\u043e\u043b\u0441\u0442\u043e\u0439: \n\
\u0410\u043d\u043d\u0430
\u041a\u0430\u0440\u0435\u043d\u0438\u043d\u0430'
        self.setGeometry(300, 300, 280, 170)
        self.setWindowTitle('Draw text')
        self.show()

    def paintEvent(self, event):

        qp = QPainter()
        qp.begin(self)
        self.drawText(event, qp)
        qp.end()

    def drawText(self, event, qp):

        qp.setPen(QColor(168, 34, 3))
        qp.setFont(QFont('Decorative', 10))
        qp.drawText(event.rect(),    Qt.AlignCenter,
self.text)

if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = Example()
      sys.exit(app.exec_())
```

Реализацию можно посмотреть на Рис.7:

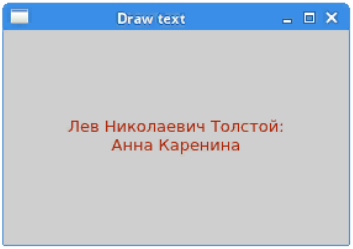


Рис.7. Рисование текста

QBrush – это элементарный графический объект. Он используется для рисования фона графических форм, таких как прямоугольники, эллипсы или многоугольники. Кисть может быть трёх разных типов: предопределённая кисть, градиент, образец текстуры.

```
import sys
from PyQt5.QtWidgets import QWidget, QApplication
from PyQt5.QtGui import QPainter, QBrush
from PyQt5.QtCore import Qt

class Example(QWidget):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):
        self.setGeometry(300, 300, 355, 280)
        self.setWindowTitle('Brushes')
        self.show()

    def paintEvent(self, e):
        qp = QPainter()
        qp.begin(self)
        self.drawBrushes(qp)
```

```
        qp.end()

    def drawBrushes(self, qp):
        brush = QBrush(Qt.SolidPattern)
        qp.setBrush(brush)

        qp.drawRect(10, 15, 90, 60)
        brush.setStyle(Qt.Dense1Pattern)
        qp.setBrush(brush)
        qp.drawRect(130, 15, 90, 60)
        brush.setStyle(Qt.Dense2Pattern)
        qp.setBrush(brush)
        qp.drawRect(250, 15, 90, 60)

        brush.setStyle(Qt.Dense3Pattern)
        qp.setBrush(brush)
        qp.drawRect(10, 105, 90, 60)
        brush.setStyle(Qt.DiagCrossPattern)
        qp.setBrush(brush)
       qp.drawRect(10, 105, 90, 60)
        brush.setStyle(Qt.Dense5Pattern)
        qp.setBrush(brush)
        qp.drawRect(130, 105, 90, 60)

        brush.setStyle(Qt.Dense6Pattern)
        qp.setBrush(brush)
        qp.drawRect(250, 105, 90, 60)

        brush.setStyle(Qt.HorPattern)
        qp.setBrush(brush)
        qp.drawRect(10, 195, 90, 60)

        brush.setStyle(Qt.VerPattern)
        qp.setBrush(brush)
        qp.drawRect(130, 195, 90, 60)

        brush.setStyle(Qt.BDiagPattern)
        qp.setBrush(brush)
        qp.drawRect(250, 195, 90, 60)

if __name__ == '__main__':
```

```
    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

Вы также можете использовать теоретические и практические знания, полученные при работе с библиотеками Python, и использовать для построения графиков Matplotlib. Реализацию данного алгоритма можно увидеть на Рис.8:

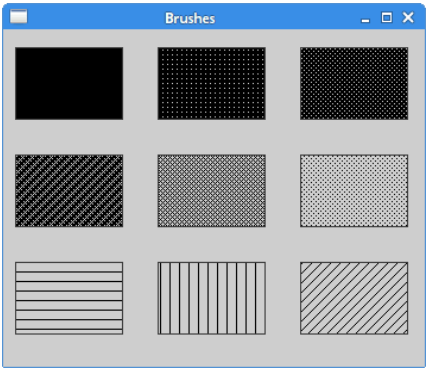


Рис.8. Рисование элементарных графических объектов

Вывод текстового сообщения:

```
import sys
from PyQt5 import QtGui

def window():
    app = QtGui.QApplication(sys.argv)
    w = QtGui.QWidget()
    b = QtGui.QLabel(w)
    b.setText("Hello World!")
    w.setGeometry(100,100,200,50)
    b.move(50,20)
    w.setWindowTitle("PyQt")
    w.show()
```

```
    sys.exit(app.exec_())

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

Реализация данной функции библиотеки(Рис. 9):

Рис.9.Вывод текстового сообщения

Для выполнения домашнего задания необходимо будет создание активных кнопок. Примерную реализацию можно увидеть на Рис.10.

```
import sys
from PyQt5.QtCore import pyqtSignal, QObject
from PyQt5.QtWidgets import QMainWindow,
QPushButton, QApplication

# closeApp
class Communicate(QObject):
    closeApp = pyqtSignal()

class Example(QMainWindow):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):

        btn1 = QPushButton("Button 1",self)
        btn1.move(30,50)
        btn2 = QPushButton("Button 2",self)
        btn2.move(150,50)
        btn1.clicked.connect(self.buttonClicked)
        btn2.clicked.connect(self.buttonClicked)
```

```
10
        self.statusBar()


        self.c = Communicate()
        self.c.closeApp.connect(self.close)

        self.setGeometry(300,300,250,350)
        self.setWindowTitle('Signal and slot')
        self.show()

    # slot, statusBar
    def buttonClicked(self):
        sender = self.sender()

self.statusBar().showMessage(sender.text()+" was
pressed")

    # closeApp
    def mousePressEvent(self,event):
        self.c.closeApp.emit()

if __name__=='__main__':
    app=QApplication(sys.argv)
    ex=Example()
    sys.exit(app.exec_())
```

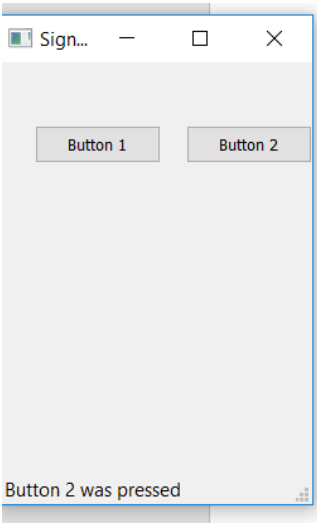


Рис.10.Создание кнопок

Для выполнение некоторых пунктов домашней работы необходимо будет создать специальную ячейку для ввода текстового сообщения(Рис.11).

```
  import sys
  from PyQt5 import QtCore, QtWidgets
  from  PyQt5.QtWidgets  import  QMainWindow,  QWidget,
QLabel, QLineEdit
  from PyQt5.QtWidgets import QPushButton
  from PyQt5.QtCore import QSize

  class MainWindow(QMainWindow):
      def __init__(self):
          QMainWindow.__init__(self)

          self.setMinimumSize(QSize(320, 140))
          self.setWindowTitle("PyQt Line Edit example
(textfield) - pythonprogramminglanguage.com")

          self.nameLabel = QLabel(self)
          self.nameLabel.setText('Name:')
    self.line = QLineEdit(self)
```

```
            self.line.move(80, 20)
            self.line.resize(200, 32)
            self.nameLabel.move(20, 20)

            pybutton = QPushButton('OK', self)
            pybutton.clicked.connect(self.clickMethod)
            pybutton.resize(200,32)
            pybutton.move(80, 60)

        def clickMethod(self):
            print('Your name: ' + self.line.text())

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    mainWin = MainWindow()
    mainWin.show()
    sys.exit( app.exec_() )
```

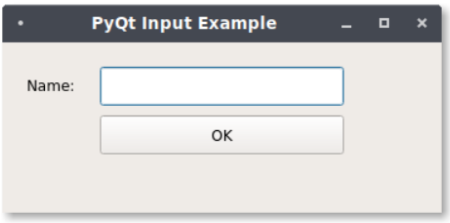


Рис.11. Ввод текста в специальную ячейку

Также для некоторых вариантов домашней работы необходимо реализовать возможность использования кнопок-переключателей. Данную функцию можно получить, запустив следующий код. Реализацию можно увидеть на Рис.12.

```
if b.isChecked() == True:
      print b.text()+" is selected"
   else:
      print b.text()+" is deselected"

import sys
```

```
   from PyQt4.QtCore import *
   from PyQt4.QtGui import *

   class Radiodemo(QWidget):

      def __init__(self, parent = None):
         super(Radiodemo, self).__init__(parent)

         layout = QHBoxLayout()
         self.b1 = QRadioButton("Button1")
         self.b1.setChecked(True)

self.b1.toggled.connect(lambda:self.btnstate(self.b1))
         layout.addWidget(self.b1)

         self.b2 = QRadioButton("Button2")

self.b2.toggled.connect(lambda:self.btnstate(self.b2))

         layout.addWidget(self.b2)
         self.setLayout(layout)
         self.setWindowTitle("RadioButton demo")

      def btnstate(self,b):

         if b.text() == "Button1":
            if b.isChecked() == True:
               print b.text()+" is selected"
            else:
               print b.text()+" is deselected"

         if b.text() == "Button2":
            if b.isChecked() == True:
               print b.text()+" is selected"
            else:
               print b.text()+" is deselected"

   def main():

      app = QApplication(sys.argv)
      ex = Radiodemo()
```

```
    ex.show()
    sys.exit(app.exec_())

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

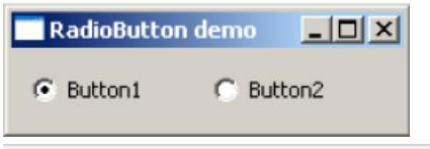


Рис.12. Создание кнопок-переключателей

Для того,чтобы на при тестирование программы можно было одновременно пользовать сразу несколькими кнопками с возможностью выбора необходимо использовать Groupbox(Рис.13).

```
import sys
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import (QApplication,
QCheckBox, QGridLayout, QGroupBox,
        QMenu, QPushButton, QRadioButton,
QVBoxLayout, QWidget)

class Window(QWidget):
    def __init__(self, parent=None):
        super(Window, self).__init__(parent)
        grid = QGridLayout()

grid.addWidget(self.createExampleGroup(),0,0)
grid.addWidget(self.createExampleGroup(),1,0)
grid.addWidget(self.createExampleGroup(),0,1)
grid.addWidget(self.createExampleGroup(),1,1)
        self.setLayout(grid)

        self.setWindowTitle("PyQt5 Group Box")
        self.resize(400, 300)

    def createExampleGroup(self):
        groupBox = QGroupBox("Best Food")
```

```
        radio1 = QRadioButton("&Radio pizza")
        radio2 = QRadioButton("R&adio taco")
        radio3 = QRadioButton("Ra&dio burrito")

        radio1.setChecked(True)

        vbox = QVBoxLayout()
        vbox.addWidget(radio1)
        vbox.addWidget(radio2)
        vbox.addWidget(radio3)
        vbox.addStretch(1)
        groupBox.setLayout(vbox)

        return groupBox

if __name__ == '__main__':
    app = QApplication(sys.argv)
    clock = Window()
    clock.show()
    sys.exit(app.exec_())
```

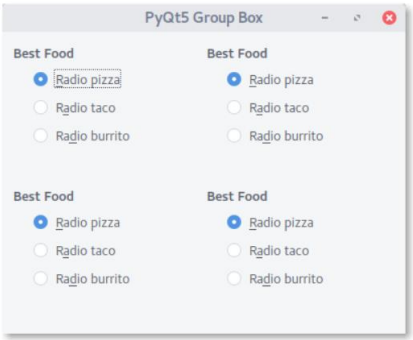


Рис.13. Создание нескольких кнопок-переключателей

QCheckBox – чекбокс (виджет, который имеет два состояния: включен и выключен). Как правило, чекбоксы используют для функций приложения, которые могут быть включены или выключены.

```
import sys
from PyQt5.QtWidgets import QWidget, QCheckBox,
QApplication
from PyQt5.QtCore import Qt


class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()


    def initUI(self):

        cb = QCheckBox('Show title', self)
        cb.move(20, 20)
        cb.toggle()
        cb.stateChanged.connect(self.changeTitle)

        self.setGeometry(300, 300, 250, 150)
        self.setWindowTitle('QCheckBox')
        self.show()


    def changeTitle(self, state):

        if state == Qt.Checked:
            self.setWindowTitle('QCheckBox')

        else:
            self.setWindowTitle('')


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

В нашем примере, мы создаём чекбокс, который переключает заголовок окна.

```
cb = QCheckBox('Show title', self)
```

Это конструктор QCheckBox.

```
cb.toggle()
```

Мы установили заголовок окна так, что мы должны к тому же проверять чекбокс. По умолчанию, заголовок окна не установлен и чекбокс выключен.

```
cb.stateChanged.connect(self.changeTitle)
```

Мы связываем наш метод changeTitle(), с сигналом stateChanged. Метод changeTitle() будет переключать заголовок окна.

```
def changeTitle(self, state):

    if state == Qt.Checked:

        self.setWindowTitle('QCheckBox')
    else:
        self.setWindowTitle('')
```

Если виджет помечен галочкой, мы устанавливаем заголовок окна. В противном случае, мы устанавливаем пустую строку в заголовке(Рис.14).

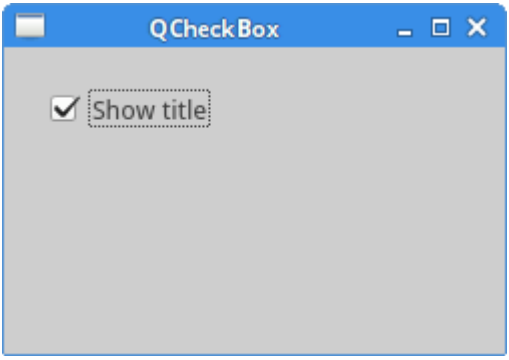


Рис.14. Виджет, имеющий два состояния: включен и выключен

Qslider – ползунок (виджет, который имеет простой регулятор). Этот регулятор может быть утянут назад и вперёд. Таким способом, мы выбираем значение для конкретной задачи. Иногда, использование ползунка более естественно, чем ввод числа или использование переключателя-счётчика. В нашем примере, мы покажем один ползунок и одну метку. Метка будет показывать изображение. Ползунок будет контролировать метку.

```
import sys
from PyQt5.QtWidgets import (QWidget, QSlider,
    QLabel, QApplication)
from PyQt5.QtCore import Qt
from PyQt5.QtGui import QPixmap


class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()


    def initUI(self):
```

```
        sld = QSlider(Qt.Horizontal, self)
        sld.setFocusPolicy(Qt.NoFocus)
        sld.setGeometry(30, 40, 100, 30)

sld.valueChanged[int].connect(self.changeValue)

        self.label = QLabel(self)
        self.label.setPixmap(QPixmap('mute.png'))
        self.label.setGeometry(160, 40, 80, 30)

        self.setGeometry(300, 300, 280, 170)
        self.setWindowTitle('QSlider')
        self.show()


    def changeValue(self, value):

        if value == 0:

self.label.setPixmap(QPixmap('mute.png'))
        elif value > 0 and value <= 30:

self.label.setPixmap(QPixmap('min.png'))
        elif value > 30 and value < 80:

self.label.setPixmap(QPixmap('med.png'))
        else:

self.label.setPixmap(QPixmap('max.png'))


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

В нашем примере, мы симулируем контроль громкости. Путём перетаскивания регулятора ползунка, мы меняем изображение на метке.

```
sld = QSlider(Qt.Horizontal, self)
```

Здесь мы создаём горизонтальный ползунок.

```
self.label = QLabel(self)
self.label.setPixmap(QPixmap('mute.png'))
```

Мы создаём виджет QLabel и устанавливаем начальное изображение "Mute" на него.

```
sld.valueChanged[int].connect(self.changeValue)
```

Мы привязываем сигнал valueChanged к определенному нами методу changeValue().

```
if value == 0:
    self.label.setPixmap(QPixmap('mute.png'))
...
```

Основываясь на значении ползунка, мы устанавливаем изображение на метку. В коде выше, мы устанавливаем изображение mute.png на метку, если ползунок приравнен к нулю(Рис.15)

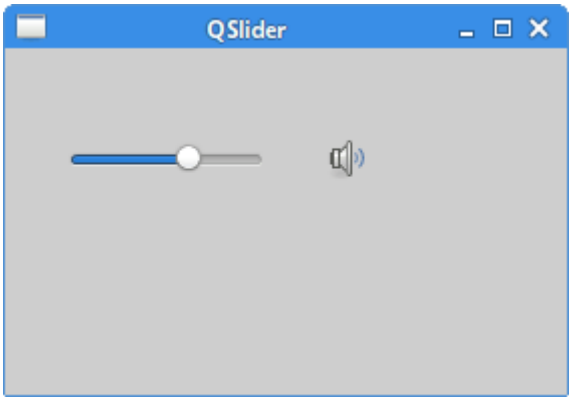


Рис.15.Слайдер

QProgressBar – прогресс бар (виджет, который используется, когда мы обрабатываем продолжительные задачи). Он анимирует процесс,

чтобы пользователи знали, что задача продвигается. Мы можем установить минимальное и максимальное значение для прогресс бара. Значения по умолчанию – 0 и 99.

```
import sys
from PyQt5.QtWidgets import (QWidget, QProgressBar,
    QPushButton, QApplication)
from PyQt5.QtCore import QBasicTimer


class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()

    def initUI(self):

        self.pbar = QProgressBar(self)
        self.pbar.setGeometry(30, 40, 200, 25)

        self.btn = QPushButton('Start', self)
        self.btn.move(40, 80)
        self.btn.clicked.connect(self.doAction)

        self.timer = QBasicTimer()
        self.step = 0

        self.setGeometry(300, 300, 280, 170)
        self.setWindowTitle('QProgressBar')
        self.show()


    def timerEvent(self, e):

        if self.step >= 100:
            self.timer.stop()
            self.btn.setText('Finished')
            return
```

```
            self.step = self.step + 1
            self.pbar.setValue(self.step)


        def doAction(self):

            if self.timer.isActive():
                self.timer.stop()
                self.btn.setText('Start')
            else:
                self.timer.start(100, self)
                self.btn.setText('Stop')


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

В нашем примере мы имеем горизонтальный индикатор прогресса и кнопку. Кнопка запускает и останавливает индикатор прогресса.

```
self.pbar = QProgressBar(self)
```

Это конструктор QProgressBar.

```
self.timer = QtCore.QBasicTimer()
```

Чтобы активировать индикатор прогресса, мы используем объект таймера.

```
self.timer.start(100, self)
```

Чтобы запустить событие таймера, мы вызываем его методом start(). Этот метод имеет два параметра: таймаут, и объект, который будет принимать события.

```
def timerEvent(self, e):
```

```
        if self.step >= 100:

            self.timer.stop()
            self.btn.setText('Finished')
            return

        self.step = self.step + 1
        self.pbar.setValue(self.step)
```

Каждый QObject и его наследники имеют обработчик событий timerEvent(). Для того, чтобы реагировать на события таймера, мы переопределяем обработчик событий.

```
    def doAction(self):

        if self.timer.isActive():
            self.timer.stop()
            self.btn.setText('Start')

        else:
            self.timer.start(100, self)
            self.btn.setText('Stop')
```

Внутри метода doAction(), мы запускаем и останавливаем таймер(Рис.16).

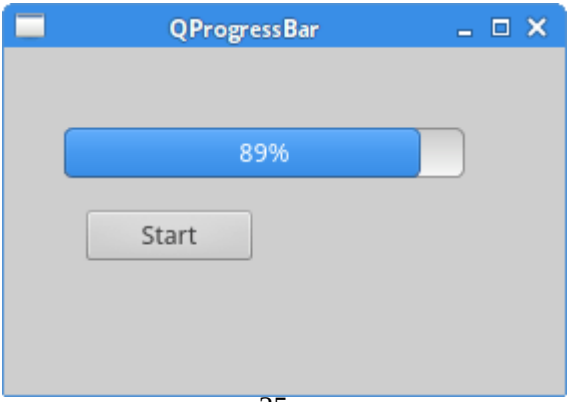


Рис.16. Индикатор выполнения

QComboBox – это виджет, который позволяет пользователю выбирать из списка вариантов (выпадающий список).

```
import sys
from PyQt5.QtWidgets import (QWidget, QLabel,
    QComboBox, QApplication)


class Example(QWidget):

    def __init__(self):
        super().__init__()

        self.initUI()


    def initUI(self):

        self.lbl = QLabel("Ubuntu", self)

        combo = QComboBox(self)
        combo.addItems(["Ubuntu", "Mandriva",
                        "Fedora", "Arch", "Gentoo"])

        combo.move(50, 50)
        self.lbl.move(50, 150)


combo.activated[str].connect(self.onActivated)

        self.setGeometry(300, 300, 300, 200)
        self.setWindowTitle('QComboBox')
        self.show()


    def onActivated(self, text):
```

```
            self.lbl.setText(text)
            self.lbl.adjustSize()


if __name__ == '__main__':

    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

Пример показывает QComboBox и QLabel. Блок со списком имеет список из пяти вариантов. Это имена дистрибутивов Linux. Виджет метки показывает выбранный вариант.

```
combo = QComboBox(self)
combo.addItems(["Ubuntu", "Mandriva",
                "Fedora", "Arch", "Gentoo"])
```

Мы создаём виджет QComboBox с пятью вариантами.

```
combo.activated[str].connect(self.onActivated)
```

После выбора пункта, мы вызываем метод onActivated().

```
def onActivated(self, text):

    self.lbl.setText(text)
    self.lbl.adjustSize()
```

Внутри метода, мы устанавливаем текст выбранного пункта в виджет метки. Мы приспосабливаем размер метки, как в прошлом примере(Рис.17).

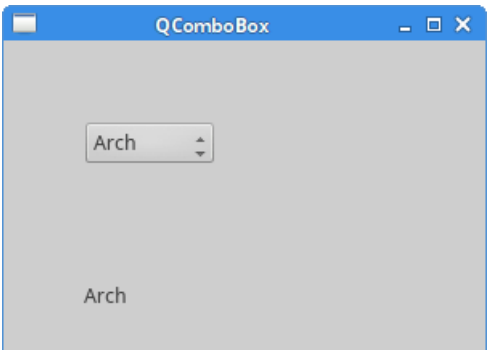


Рис.17.Возможность выбора из представленного списка

В некоторых вариантах необходимо использовать SpinBox.Он показывает окошко счетчика значений(Рис.18).

```
   import sys
   from PyQt4.QtCore import *
   from PyQt4.QtGui import *

   class spindemo(QWidget):
      def __init__(self, parent = None):
         super(spindemo, self).__init__(parent)

         layout = QVBoxLayout()
         self.l1 = QLabel("current value:")
         self.l1.setAlignment(Qt.AlignCenter)
         layout.addWidget(self.l1)

         self.sp = QSpinBox()

         layout.addWidget(self.sp)

self.sp.valueChanged.connect(self.valuechange)

         self.setLayout(layout)

         self.setWindowTitle("SpinBox demo")

      def valuechange(self):

         self.l1.setText("current

    value:"+str(self.sp.value()))
```

```
def main():
    app = QApplication(sys.argv)
    ex = spindemo()
    ex.show()
    sys.exit(app.exec_())

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

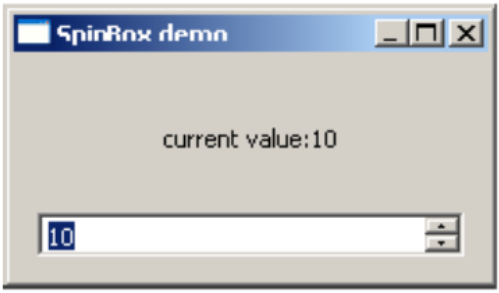


Рис.18.Окошко счётчика значений

Также в большинстве заданий придется столкнуться с работай с картинкой, поэтому нужно научится методам реализации загрузки картинки(Рис.19).

```
import sys
from PyQt4.QtCore import *
from PyQt4.QtGui import *

def window():
    app = QApplication(sys.argv)
    win = QWidget()

    l1 = QLabel()
    l2 = QLabel()
```

```
    l3 = QLabel()
    l4 = QLabel()

    l1.setText("Hello World")
    l4.setText("TutorialsPoint")
    l2.setText("welcome to Python GUI Programming")

    l1.setAlignment(Qt.AlignCenter)
    l3.setAlignment(Qt.AlignCenter)
    l4.setAlignment(Qt.AlignRight)
    l3.setPixmap(QPixmap("python.jpg"))

    vbox = QVBoxLayout()
    vbox.addWidget(l1)
    vbox.addStretch()
    vbox.addWidget(l2)
    vbox.addStretch()
    vbox.addWidget(l3)
    vbox.addStretch()
    vbox.addWidget(l4)

    l1.setOpenExternalLinks(True)
    l4.linkActivated.connect(clicked)
    l2.linkHovered.connect(hovered)

l1.setTextInteractionFlags(Qt.TextSelectableByMouse)
    win.setLayout(vbox)

    win.setWindowTitle("QLabel Demo")
    win.show()
    sys.exit(app.exec_())

def hovered():
   print "hovering"
def clicked():
   print "clicked"

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

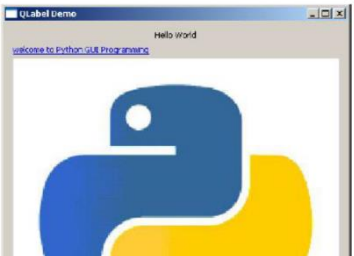


Рис.19.Загрузка изображения

Мы можем предоставить всплывающую подсказку для любого из виджетов. Использование подсказка (tooltip) показано на примере ниже(Рис.20).

```
   import sys
   from PyQt5 import QtCore, QtWidgets
   from PyQt5.QtWidgets import QMainWindow, QWidget,
QPushButton
   from PyQt5.QtCore import QSize

   class MainWindow(QMainWindow):
       def __init__(self):
           QMainWindow.__init__(self)

           self.setMinimumSize(QSize(300, 100))
           self.setWindowTitle("PyQt tooltip example -
pythonprogramminglanguage.com")

           pybutton = QPushButton('Pyqt', self)
           pybutton.clicked.connect(self.clickMethod)
           pybutton.resize(100,32)
           pybutton.move(50, 20)
           pybutton.setToolTip('This is a tooltip
message.')
       def clickMethod(self):
           print('PyQt')
```

```
if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    mainWin = MainWindow()
    mainWin.show()
    sys.exit( app.exec_() )
```

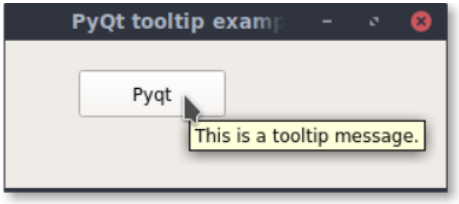


Рис.20.Использование виджета подсказки

QMessageBox предоставляет собой простое диалоговое окно для информирования пользователя или для запроса вопроса и получения ответа(Рис.21.)

```
import sys
from PyQt5 import QtCore, QtWidgets
from PyQt5.QtWidgets import QMainWindow, QLabel,
QGridLayout, QWidget
from PyQt5.QtWidgets import QPushButton

from PyQt5.QtWidgets import QMessageBox
from PyQt5.QtCore import QSize

class MainWindow(QMainWindow):

    def __init__(self):

        QMainWindow.__init__(self)

        self.setMinimumSize(QSize(300, 200))
        self.setWindowTitle("PyQt messagebox
example - pythonprogramminglanguage.com")
```

```
            pybutton = QPushButton('Show messagebox',
self)
            pybutton.clicked.connect(self.clickMethod)
            pybutton.resize(200,64)

            pybutton.move(50, 50)

        def clickMethod(self):
            QMessageBox.about(self, "Title", "Message")

    if __name__ == "__main__":
        app = QtWidgets.QApplication(sys.argv)
        mainWin = MainWindow()
        mainWin.show()
        sys.exit( app.exec_() )
```

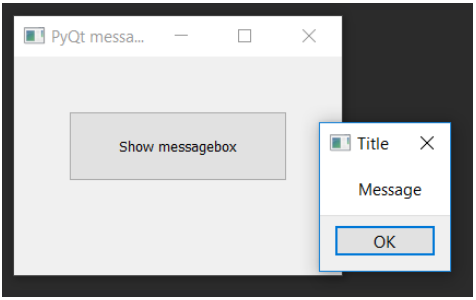


Рис.21. Диалоговое окно для предоставления ответа

ЗАДАЧИ И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для начала рассмотрим процесс создания программы в PyQt состоящий из трех этапов:

1. рисуется интерфейс в программе Designer, сохраняется как .ui файл;
2. .ui файл конвертируется в файл .py с помощью утилиты pyuic
3. сконвертированный файл подключается к основному модулю программы, пишется код программы.

Первое задание.Попробуем реализовать очень простую программу, которая будет складывать два числа. В программе будет два поля ввода, кнопка и текстовая надпись. Данный пример максимально просто позволит нам начать работать с PyQt.

Сперва нарисуем интерфейс нашего будущего приложения. Запускаем программу Designer, которая находится в меню пуск в группе программ, относящихся к PyQt.

При запуске программы возникнет окно, которое предлагает создать проект. В списке сверху нужно выбрать Main Window и нажать кнопку Создать(Рис.22.)

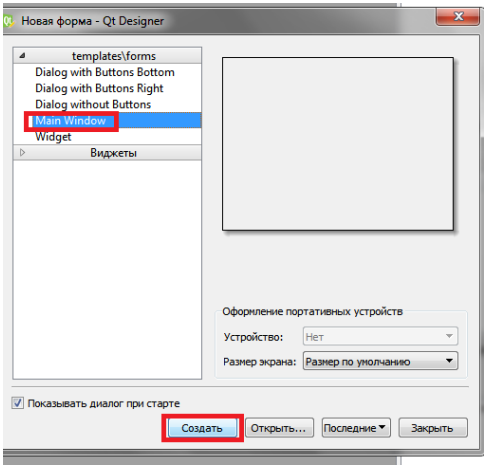


Рис.22.Создание проекта

Увидим перед собой среду визуальной разработки интерфейса. Посередине находится заготовка окна нашей будущей программы. На нее нужно перетаскивать мышкой элементы интерфейса из списка слева (кнопки, поля ввода и т.д.) После того, как элемент интерфейса расположен в нужном месте, можно выделить его щелчком мыши и настроить его свойства в списке справа.

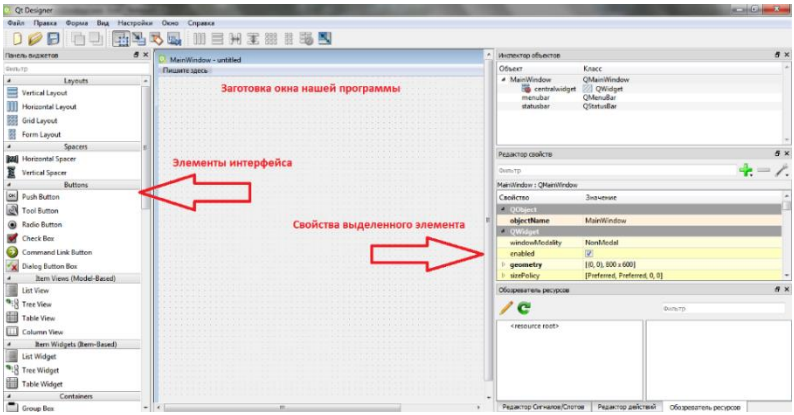


Рис.23.Настройка свойств

Поместим на заготовку окна два элемента LineEdit, пролистав список, расположенный слева до самого низа, и найдя там группу элементов под названием Input Widgets. Нам нужно взять третий элемент этой группы под названием LineEdit и перетащить его мышкой на наше окно посередине. Потом повторить это действие, разместив на заготовки второе поле ввода(Рис.24).

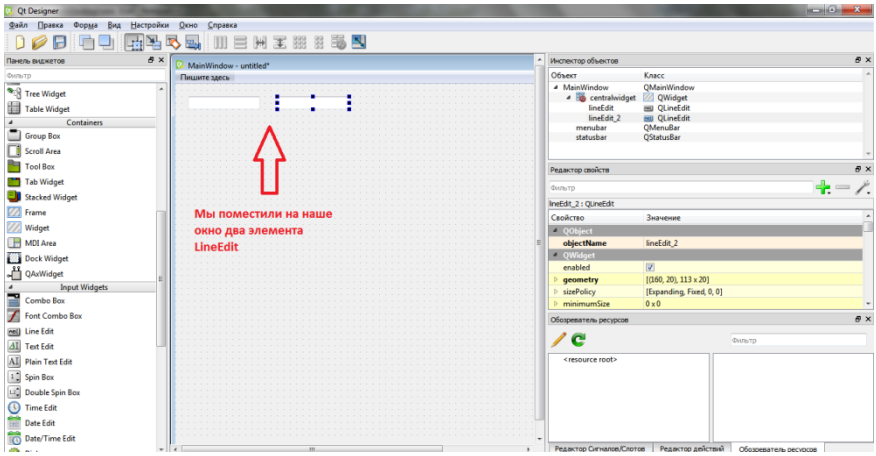


Рис.24.Размещение второго поля для ввода

Пролистаем список слева еще ниже и найдем группу элементов под названием Display Widgets. Найдем в ней элемент Label и перетащим его на нашу заготовку. Теперь добавим кнопку, также перетащив ее из списка слева (группа Buttons, элемент PushButton)(Рис.25).

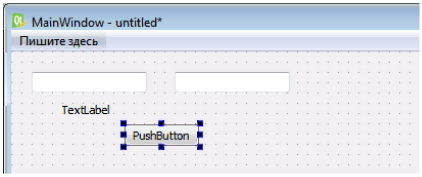


Рис.25.Добавление кнопки

Теперь измените размеры элементов так, чтобы получилось красиво. Для этого просто тяните за квадратики, которые появляются вокруг элементов при его выделении мышью. Щелкните на элементе Label дважды и введите надпись "Здесь будет результат сложения".

Два раза щелкните на кнопки и введите текст "Сложить числа"(Рис.26.)

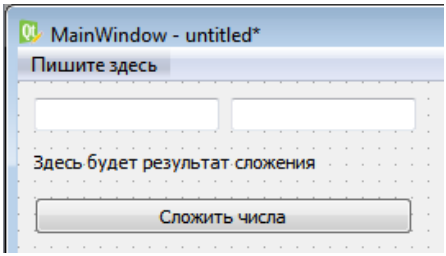


Рис.26.Оформление интерфейса

Теперь уменьшим размер основного окна программы (можно этого и не делать, так как программа тестовая, но можно и сделать для красоты).

Сохраним созданный интерфейс в какую-нибудь папку на диске D, желательно назвать ее на английском языке и без пробелов.

Переходим ко второму этапу. У нас уже есть созданный интерфейс программы. Но нам нужно сконвертировать его в файл Python с расширением .py. Для этого в командной строке Windows даём команду

```
D:\Python36\Scripts\pyuic5.exe D:\test\my.ui -o D:\test\my.py
```

Обратите внимание что в этой команде у вас будет другой путь до папки Python, так как у меня Python установлен в папку Python36 на диске D, а у вас это скорее всего не так. Также не забудьте про опцию -o которая есть в данной строке.

После выполнения данной команды в нашей папке test появится модуль my.py который является сконвертированным модулем интерфейса, и который мы уже можем подключить к нашей будущей программе.

Используем стандартную заготовку (скелет) основной программы:

```
from PyQt5 import QtCore, QtGui, QtWidgets
import sys

# Импортируем наш модуль интерфейса my.py
from my import *


class MyWin(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        # Здесь прописываем событие нажатия на кнопку

self.ui.pushButton.clicked.connect(self.buttonpress
)

    # Пока пустая функция которая выполняется
    # при нажатии на кнопку
    def buttonpress(self):
        # Здесь должны быть действия по нажатию
кнопки
        # но пока здесь заглушка pass
        pass

if __name__=="__main__":
    app = QtWidgets.QApplication(sys.argv)
    myapp = MyWin()
    myapp.show()
    sys.exit(app.exec_())
```

В этой заготовке уже прописан импорт нашего модуля интерфейса my.py и событие нажатия на кнопку (но внутри него пока ничего нет). Расширение программы .pyw что позволяет избавиться от окна консоли. Если мы поместим данный файл в нашу папку test рядом с my.py и запустим то у нас откроется наша программа, которая впрочем еще не складывает числа(Рис.27).

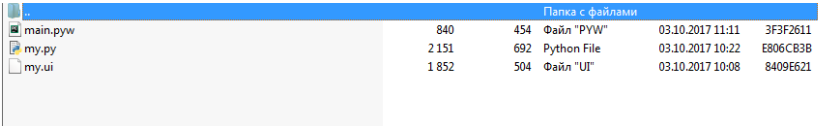


Рис.27. Содержание папки, откуда будет запускаться программа

Дальше доведем проект до логического завершения и пропишем в функцию **def** buttonpress(**self**): вместо заглушки pass команды, которые получат переменные, введенные пользователем в поля ввода, и выведут результат сложения в текстовую надпись Label.

```
from PyQt5 import QtCore, QtGui, QtWidgets
import sys

# Импортируем наш модуль интерфейса my.py
from my import *
class MyWin(QtWidgets.QMainWindow):
    def __init__(self, parent=None):
        QtWidgets.QWidget.__init__(self, parent)
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)

        # Здесь прописываем событие нажатия на кнопку

self.ui.pushButton.clicked.connect(self.buttonpress)
    def buttonpress(self):
        a=self.ui.lineEdit.text()
        b=self.ui.lineEdit_2.text()
        c=int(a)+int(b)
        self.ui.label.setText(str(c))
if __name__=="__main__":
    app = QtWidgets.QApplication(sys.argv)
    myapp = MyWin()
    myapp.show()
    sys.exit(app.exec_())
```

Обратите внимание, что для обращения к элементам графического интерфейса мы пишем спереди их имени **self**.ui.

Итак, запустим нашу программу, введём два числа и нажав кнопку, увидим результат сложения чисел в текстовой надписи над ней(Рис.28).

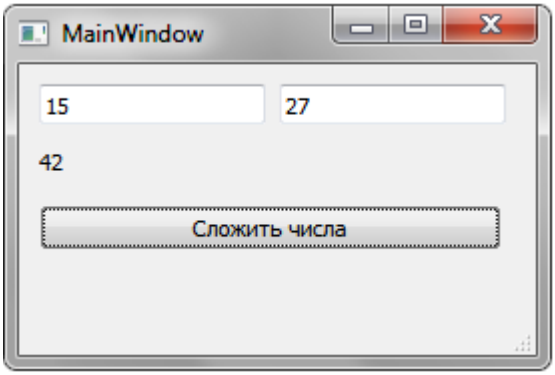


Рис.28.Проверка работы первого задания

Вторая задача. Необходимо выполнить построение графика $x^3\,cos^3\,x - \frac{1}{x}$.Работать будет не только с использованием функций библиотеки **PyQt5**но и, вспоминая прошлый опыт выполнения работ по курсы, с использованием библиотек Matplotlib и Numpy

```
import sys
from PyQt5.QtCore import pyqtSignal, QObject
from PyQt5.QtWidgets import QMessageBox
from PyQt5.QtWidgets import QApplication, QLabel
from PyQt5.QtWidgets import QMainWindow, QPushButton


import matplotlib.pyplot as plt
import numpy as np
import math
```

```
class Communicate(QObject):
    closeApp = pyqtSignal()

class Example(QMainWindow):
    def __init__(self):
        super().__init__()
        self.initUI()

    def initUI(self):

        btn1 = QPushButton('Built me!', self)
        btn1.move(30, 30)

        btn1.clicked.connect(btn1_clicked)

self.setMinimumSize(640, 140)

def btn1_clicked():

 x = np.arange(30, 40, 0.5)

y = (np.cos(x * x * x) * (x * x * x))

for i in range(0, len(x)):
        y[i] = y[i] * (1 / x[i])
    plt.plot(x, y)
    plt.show()


if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = Example()
    sys.exit(app.exec_())
```

Реализация(Рис.29):

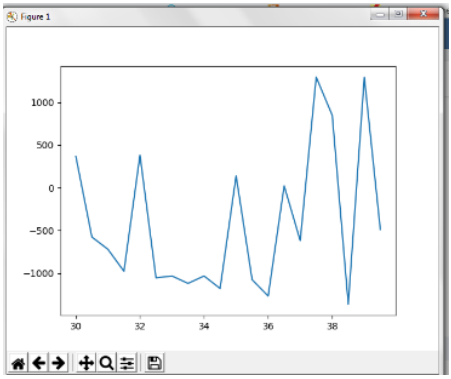


Рис.29.График заданной функции

ЗАДАНИЕ НА ДОМАШНЮЮ РАБОТУ

Вариант 1

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\left(1 - 4x - \frac{x^3}{17}\right) sin(x^2)$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Запрограммировать кнопку «Built me!», нажав на которую график вашего варианта общего задания появится в окне. На свое усмотрение вы также можете использовать различные цвета и стили для оформления кнопки.
2. По центру выведите цветное текстовое сообщение “Сложный выбор. Добавьте на экран вывода картинку. Создайте две кнопки “Чай” и “Кофе”, при нажатии на каждую из которых в окне будет появляться сообщение о том, что пользователь выбирает чай или кофе соответственно. Пример показан на Рис.30

Рис.30.Пример созданного сообщение на экране

3. Организовать Message Box, при нажатии на которую выводится сообщение, содержащее:

- ФИО;
- Специальность;
- Группа.

4. Создать Button, при наведении на которую всплывает подсказка, из которой можно узнать о пользе чая и кофе. При нажатии на кнопку в окне ввода-вывода появляется сообщение “Обращение за подсказкой”.

Вариант 2

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\left(1 - 10x - \frac{x^2}{13}\right) tg(x^3)$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Создать цветное текстовое сообщение “Радуга”.
2. Создать CheckBox, позволяющий выводить на экран круги семи цветов радуги. Создайте подсказку (ToolTipe), при наведении на которую всплывает подсказка, в которой говорится о возможных вариантах изменения цвета(Рис.31)

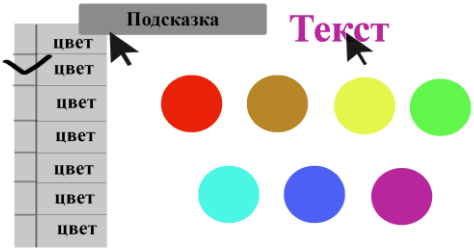


Рис.31 Пример выполнение задания

3. Организовать Slider, позволяющий увидеть количество (цифра) выбранных в CheckBox кругов.

Вариант 3

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\frac{|sin(2x)|}{sin\,x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Поставить надпись «График (формула) был построен» напротив графика.
2. Создать Toggle button, содержащий варианты выбора цвета светофора. При выборе рядом стоящий квадрат должен появляться, окрашенный в выбранный цвет. Создайте цветное текстовое сообщение “Светофор”,пример наРис.33.

Рис.33.Пример реализации программы

3. Создать Button с названием “Справка”, при наведении на которую всплывает подсказка, из которой можно узнать о истории появления светофора. При нажатии на кнопку в окне ввода-вывода появляется сообщение “Обращение за справкой”.

Вариант 4

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\frac{|cos(2x)|}{cos\, x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Запрограммировать кнопку «Built me!», нажав на которую график вашего варианта общего задания появится в окне, на свое усмотрение, вы также можете использовать различные цвета и стили для оформления кнопки.
2. Задать изначальные значения логина и пароля в программе. Создайте 2 текстовых поля: логин и пароль и кнопку “Войти”. При введении неправильного логина или пароля должно появляться сообщение об ошибке. При верно введённом логине и пароля программа выводит сообщение об успешном входе в систему(Рис.34).

Рис.34.Пример реализации программы

Организовать Message Box, при нажатии на которую выводится сообщение, содержащее:

- ФИО;
- Специальность;
- Группа.

Вариант 5

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\frac{|sin(5x)|}{sin(x-1)}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Задать нумерацию оси координат вашего графика
2. Создайте список продуктов и CheckBox выбора для каждого из них. Напишите цену каждого из продуктов после названия. При выборе всех нужных пользователю продуктов и нажатии на кнопку должно выводиться сообщение об

итоговой цене покупки(Рис.35)

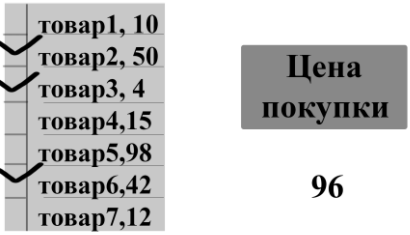


Рис.35.Пример реализации задания

3. Организовать Message Box, при нажатии на которую выводится сообщение, содержащее:
   - ФИО;
   - Специальность;
   - Группа.

Вариант 6

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$x^2 \sin^2 x - \frac{1}{x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Создайте цветное текстовое сообщение “Эмоции ”.
2. Создайте графические рисунки в виде смайликов, используя любой графический редактор (3 шт). Создайте ChoiceBox, позволяющий выбрать эмоцию (весёлый, грустный, злой). При выборе одной из этих эмоций и нажатии

на кнопку, справа от ChoiceBox'а должен появляться соответствующий смайлик(Рис.36).

Рис.36.Пример выполнения задания

3. Создайте Button с названием "Данные", при наведении на которую всплывает подсказка, из которой можно узнать ФИО студента и номер группы. При нажатии на кнопку в окне ввода-вывода появляется сообщение "Данные о студенте просмотрены".

Вариант 7

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$x^5 \cos^3 x - \frac{5x}{\sin x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Задать отдельное окно для вашего графика (окно подразумевает отдельную рамку для графике в главном окне).
2. Создайте поле ввода, позволяющее пополнить количество товара на складе на N единиц (число N вводится пользователем). Разделите окно на две части и разместите

элемент пополнения и кнопку “Пополнить” в левой части окна. В правой части должно быть текстовое поле, принимающее от пользователя число, определяющее количество товара, которое им было куплено. При нажатии кнопки “Купить” должно выводиться сообщение об успешной покупке (если товара на складе достаточно) и из склада должен вычитаться купленный товар, или же сообщение о том, что на складе недостаточно товара и что его необходимо пополнить(Рис. 37).

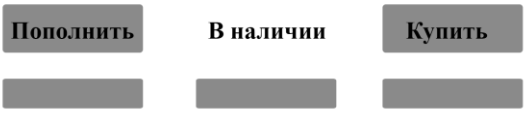


Рис.37.Приимер выполнения задания

3. Создайте Button с названием “Справка”, при наведении на которую всплывает подсказка, из которой можно узнать ФИО и номер группы студента, выполняющего домашнее задание. При нажатии на кнопку в окне ввода-вывода появляется сообщение “Обращение за справкой”.

Вариант 8

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\left(15 - x - \frac{cos\ x^3}{x}\right) sin(x^3)$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Запрограммировать кнопку «Built me!», нажав на которую график вашего варианта общего задания появится в окне. На свое усмотрение вы также можете использовать различные цвета и стили для оформления кнопки.
2. По центру выведите цветное текстовое сообщение “Сложный выбор. Добавьте на экран вывода картинку. Создайте две кнопки “Чай” и “Кофе”, при нажатии на каждую из которых в окне будет появляться сообщение о том, что пользователь выбирает чай или кофе соответственно. Пример показан на Рис.38

Рис.38.Пример созданного сообщение на экране

3. Организовать Message Box, при нажатии на которую выводится сообщение, содержащее:

ФИО

Специальность

Группа

5. Создать Button, при наведении на которую всплывает подсказка, из которой можно узнать о пользе чая и кофе. При нажатии на кнопку в окне ввода-вывода появляется сообщение “Обращение за подсказкой”.

Вариант 9

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\left(1 - 10x - \frac{x^2}{13}\right) tg(x^3)$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Создать цветное текстовое сообщение “Радуга”.
2. Создать CheckBox, позволяющий выводить на экран круги семи цветов радуги. Создайте подсказку (ToolTipe), при наведении на которую всплывает подсказка, в которой говорится о возможных вариантах изменения цвета(Рис.39)

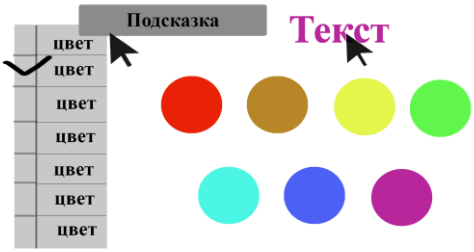


Рис.39 Пример выполнение задания

3. Организовать Slider, позволяющий увидеть количество (цифра) выбранных в CheckBox кругов.

Вариант 10

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\frac{|cos(2x)\ \sin\ (3x)|}{sin\ x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Поставить надпись «График (формула) был построен» напротив графика.
2. Создать Toggle button, содержащий варианты выбора цвета светофора. При выборе рядом стоящий квадрат должен появляться, окрашенный в выбранный цвет. Создайте цветное текстовое сообщение “Светофор”,пример наРис.40.

Рис.40.Пример реализации программы

3. Создать Button с названием “Справка”, при наведении на которую всплывает подсказка, из которой можно узнать о

истории появления светофора. При нажатии на кнопку в окне ввода-вывода появляется сообщение “Обращение за справкой”.

Вариант 11

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\frac{|cos(2x)|}{sin\,x\cos x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Запрограммировать кнопку «Built me!», нажав на которую график вашего варианта общего задания появится в окне, на свое усмотрение, вы также можете использовать различные цвета и стили для оформления кнопки.
2. Задать изначальные значения логина и пароля в программе. Создайте 2 текстовых поля: логин и пароль и кнопку “Войти”. При введении неправильного логина или пароля должно появляться сообщение об ошибке. При верно введённом логине и пароля программа выводит сообщение об успешном входе в систему(Рис.41).

Рис.41.Пример реализации программы

3. Организовать Message Box, при нажатии на которую выводится сообщение, содержащее:
   - ФИО;
   - Специальность;
   - Группа.

Вариант 12

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\frac{|sin(5x)|}{sin(x-15)}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Создать цветное текстовое сообщение “Эмоции ”.
2. Создать графические рисунки в виде смайликов, используя любой графический редактор (3 шт). Создайте ChoiceBox, позволяющий выбрать эмоцию (весёлый, грустный, злой). При выборе одной из этих эмоций и нажатии на кнопку, справа от ChoiceBox’a должен появляться соответствующий смайлик(Рис.42).

Рис.42.Пример выполнения задания

3. Создайте Button с названием “Данные”, при наведении на которую всплывает подсказка, из которой можно узнать ФИО студента и номер группы. При нажатии на кнопку в окне ввода-вывода появляется сообщение “Данные о студенте просмотрены”.

Вариант 13

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$x^4 \cos^2 x - \frac{1}{14x}$$

Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Задать нумерацию оси координат вашего графика
2. Создать список продуктов и CheckBox выбора для каждого из них. Напишите цену каждого из продуктов после названия. При выборе всех нужных пользователю продуктов и нажатии на кнопку должно выводиться сообщение об итоговой цене покупки(Рис.43).

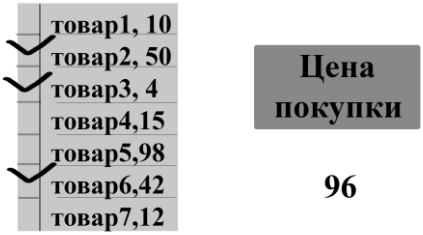


Рис.43.Пример реализации задания

3. Организовать Message Box, при нажатии на которую выводится сообщение, содержащее:
ФИО;
Специальность;
Группа.

Вариант 14

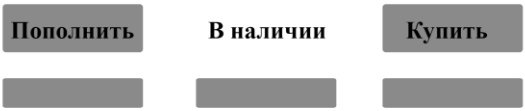
Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$x\,cos^3\,x - \frac{x}{\sin x \cos x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Задать отдельное окно для вашего графика (окно подразумевает отдельную рамку для графике в главном окне).
2. Создать поле ввода, позволяющее пополнить количество товара на складе на N единиц (число N вводится

пользователем). Разделите окно на две части и разместите элемент пополнения и кнопку “Пополнить” в левой части окна. В правой части должно быть текстовое поле, принимающее от пользователя число, определяющее количество товара, которое им было куплено. При нажатии кнопки “Купить” должно выводиться сообщение об успешной покупке (если товара на складе достаточно) и из склада должен вычитаться купленный товар, или же сообщение о том, что на складе недостаточно товара и что его необходимо пополнить(Рис. 44).



Недостаточно товара на складе!

Рис.44.Приимер выполнения задания

3. Создать Button с названием “Справка”, при наведении на которую всплывает подсказка, из которой можно узнать ФИО и номер группы студента, выполняющего домашнее задание. При нажатии на кнопку в окне ввода-вывода появляется сообщение “Обращение за справкой”.

Вариант 15

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\left(10 + 10x + \frac{x^2}{100}\right) ctg(x)$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Создать цветное текстовое сообщение “Радуга”.
2. Создать CheckBox, позволяющий выводить на экран круги семи цветов радуги. Создайте подсказку (ToolTipe), при наведении на которую всплывает подсказка, в которой говорится о возможных вариантах изменения цвета(Рис.45).

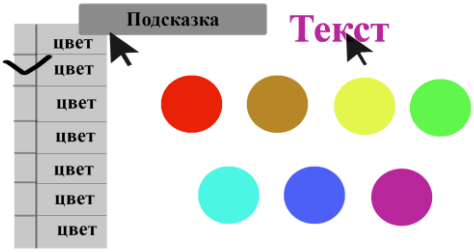


Рис.45. Пример выполнение задания

3. Организовать Slider, позволяющий увидеть количество (цифра) выбранных в CheckBox кругов.

Вариант 16

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\frac{|tg(2x)|}{sin\, x\ ctg\ x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1 Запрограммировать кнопку «Built me!», нажав на которую график вашего варианта общего задания появится в окне. На свое усмотрение вы также можете использовать различные цвета и стили для оформления кнопки.

2 По центру выведите цветное текстовое сообщение “Сложный выбор. Добавьте на экран вывода картинку. Создайте две кнопки “Чай” и “Кофе”, при нажатии на каждую из которых в окне будет появляться сообщение о том, что пользователь выбирает чай или кофе соответственно. Пример показан на Рис.46

Рис.46.Пример созданного сообщение на экране

3 Организовать Message Box, при нажатии на которую выводится сообщение, содержащее:

- ФИО;
- Специальность;
- Группа.

4 Создайте Button, при наведении на которую всплывает подсказка, из которой можно узнать о пользе чая и кофе. При нажатии на кнопку в окне ввода-вывода появляется сообщение “Обращение за подсказкой”.

Вариант 17

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\frac{|cos(2x)\ \sin\ (3x)|}{cos\ x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Поставить надпись «График (формула) был построен» напротив графика.
2. Создайте Toggle button, содержащий варианты выбора цвета светофора. При выборе рядом стоящий квадрат должен появляться, окрашенный в выбранный цвет. Создайте цветное текстовое сообщение “Светофор”,пример наРис.47.

Рис.47.Пример реализации программы

3. Создайте Button с названием “Справка”, при наведении на которую всплывает подсказка, из которой можно узнать о истории появления светофора. При нажатии

на кнопку в окне ввода-вывода появляется сообщение “Обращение за справкой”.

Вариант 18

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$\frac{|cos(2x)|}{\cos x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Запрограммировать кнопку «Built me!», нажав на которую график вашего варианта общего задания появится в окне, на свое усмотрение, вы также можете использовать различные цвета и стили для оформления кнопки.
2. Задайте изначальные значения логина и пароля в программе. Создайте 2 текстовых поля: логин и пароль и кнопку “Войти”. При введении неправильного логина или пароля должно появляться сообщение об ошибке. При верно введённом логине и пароля программа выводит сообщение об успешном входе в систему(Рис.48).

Рис.48.Пример реализации программы

3. Организовать Message Box, при нажатии на которую выводится сообщение, содержащее:
   - ФИО;
   - Специальность;
   - Группа.

Вариант 19

Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$cos^2\ x\ \ tg\ 2x - \frac{1}{4x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Создайте цветное текстовое сообщение “Эмоции ”.
2. Создайте графические рисунки в виде смайликов, используя любой графический редактор (3 шт). Создайте ChoiceBox, позволяющий выбрать эмоцию (весёлый, грустный, злой). При выборе одной из этих эмоций и нажатии на кнопку, справа от ChoiceBox’a должен появляться соответствующий смайлик(Рис.49).

Рис.49.Пример выполнения задания

3. Создайте Button с названием “Данные”, при наведении на которую всплывает подсказка, из которой можно узнать ФИО студента и номер группы. При нажатии на кнопку в окне ввода-вывода появляется сообщение “Данные о студенте просмотрены”.

Вариант 20

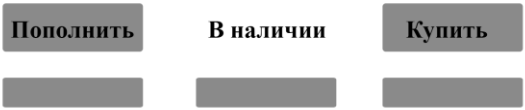
Задача 1. В созданном окне вам необходимо построить график функций на осях x и y по заданной формуле:

$$cos^3 x \sin x - \frac{5x}{\sin 2x \cos x}$$

Задача 2. Используйте предыдущее окно с нарисованным графиком, вам необходимо:

1. Задать отдельное окно для вашего графика (окно подразумевает отдельную рамку для графике в главном окне).
2. Создайте поле ввода, позволяющее пополнить количество товара на складе на N единиц (число N вводится пользователем). Разделите окно на две части и разместите элемент пополнения и кнопку “Пополнить” в левой части окна. В правой части должно быть текстовое поле,

принимающее от пользователя число, определяющее количество товара, которое им было куплено. При нажатии кнопки “Купить” должно выводиться сообщение об успешной покупке (если товара на складе достаточно) и из склада должен вычитаться купленный товар, или же сообщение о том, что на складе недостаточно товара и что его необходимо пополнить(Рис. 50).



Недостаточно товара на складе!

Рис.50.Приимер выполнения задания

3. Создайте Button с названием “Справка”, при наведении на которую всплывает подсказка, из которой можно узнать ФИО и номер группы студента, выполняющего домашнее задание. При нажатии на кнопку в окне ввода-вывода появляется сообщение “Обращение за справкой”.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Опишите процесс установки библиотеки PyQt, которым вы пользовались.
2. Кратко опишите процесс создание графического интерфейса.
3. Расскажите с помощью чего можно нарисовать простейшие геометрические фигуры, такие как линия или кривая
4. Опишите методы, с помощью которого вы будете реализовывать простейшие геометрические фигуры, такие как прямоугольники, эллипсы и многоугольники.
5. Объясните, как нужно использовать виджет всплывающей подсказки.
6. Сформулируйте определение библиотеки PyQt и опишите его основное предназначение.
7. Расскажите, как можно работать с цветом и какие цвета используются в основном при работе с библиотекой PyQt
8. Опишите виджет, который используется, когда необходимо обработать и вывести на экран процесс выполнения задачи.
9. Объясните, как можно загрузить изображение с помощью библиотеки PyQt
10. Объясните по какому принципу работаю кнопки-переключатели и как в одном диалоговом окне создать несколько активных кнопок-переключателей.
11. Сформулируйте основное предназначение виджета ComboBox.
12. Расскажите какими еще библиотеками, помимо PyQt,вы пользовались при выполнении домашнего задания.

ФОРМА ОТЧЕТА ПО ДОМАШНЕЙ РАБОТЕ

На выполнение домашней работы отводится 10 академических часа: 9 часа на выполнение и сдачу домашней работы и 1 час на подготовку отчета. Отчет на защиту предоставляется в печатном виде.

Порядок выполнения:

1. Изучить теоретический материал.
2. Получить вариант у преподавателя.
3. Разработать графический интерфейс согласно варианту.
4. Выполнить тестирование программы.
5. Продемонстрировать работу программы преподавателю.
6. Оформить отчет.
7. Защитить выполненную работу у преподавателя.

СОДЕРЖАНИЕ ОТЧЁТА

1. Титульный лист.
2. Цель, задачи работы.
3. Формулировка задания.
4. UML-диаграмма классов, используемых в программе.
5. Листинг программы.
6. Результаты выполнения программы.
7. Выводы по работе в целом.
8. Список литературы.

ДОМАШНЯЯ РАБОТА №2
РАБОТА С ФАЙЛАМИ

Целью выполнения домашней работы является приобретение практических навыков работы с файлами и файловой системой на языке программирования Python.

Основными задачами выполнения домашней работы являются:

Задачи:

1. Ознакомиться со способами работы с файлами и файловой системой в python;
2. Изучить способы работы с файлами формата csv, json, xml;
3. Закрепить полученные в ходе выполнения домашней работы навыки.

Результатами работы являются:

3. Реализация разработанных алгоритмов на языке программирования Python;
4. Подготовленный отчет.

КРАТКАЯ ХАРАКТЕРИСТИКА ОБЪЕКТА ИЗУЧЕНИЯ, ИССЛЕДОВАНИЯ

Работа с текстовым файлом в Python

Для открытия файлов в python используется функция open:

```
file = open("/path/for/your/file.txt", "r")
```

Она возвращает поток - интерфейс взаимодействия с содержимым файла.

Функция open принимает первым аргументом полное имя файла (с путём, абсолютным или относительным), вторым - режим, в котором мы откроем файл. (Табл.1.)

Таблица 1. Режимы открытия файлов

Режим	Обозначение
'r'	открытие на чтение (является значением по умолчанию).
'w'	открытие на запись, содержимое файла удаляется, если файла не существует, создается новый.
'x'	открытие на запись, если файла не существует, иначе исключение.
'a'	открытие на дозапись, информация добавляется в конец файла.
'b'	открытие в двоичном режиме.
't'	открытие в текстовом режиме (является значением по умолчанию).
'+'	открытие на чтение и запись

По умолчанию файл открывается в режиме rt - для чтения в текстовом формате.

Стоит заметить, что файл можно открыть в двух разных форматах: текстовом и бинарном (двоичном). Файлы, открытые в текстовом формате (по умолчанию, или явно добавляя “t” к аргументу режима), обрабатываются Python-ом и возвращаются как строки. При открытии файла в бинарном формате никакой обработки содержимого не производится, содержимое возвращается побайтово.

Таким образом, если мы хотим открыть файл в двоичном формате для записи, надо использовать режим “wb”, если мы хотим дописать содержимое в конец файла в текстовом формате, то - “a” или “at”, “r+b” - открыть двоичный файл на чтение и запись.

Обычно, файлы, в которых содержится текст, например, файлы txt, код вашей программы, файлы формата csv, открываются в текстовом формате, а файлы, которые нельзя проинтерпретировать как текст - в бинарном (например, картинки, музыку). Иногда файлы с текстом открывают в бинарном режиме, для более явного управления всеми спецсимволами (например табуляция ↹).

При открытии файла в текстовом режиме, также можно указать подходящую кодировку. Например, если в вашем файле содержится текст на русском в utf8, откройте его в этой кодировке:

```
russian_file = open("russian.txt", "r", encoding="utf8")
```

Как только файл был открыт и у вас появился файловый объект, вы можете получить следующую информацию о нем(Табл.2.):

Таблица 2. Информация о файде

Атрибут	Значение
file.closed	Возвращает True если файл был закрыт.

file.mode	Возвращает режим доступа, с которым был открыт файл.
file.name	Возвращает имя файла.

У получаемого объекта есть несколько полезных методов, рассмотрим их.

• *метод read(n)* позволяет прочитать следующие n символов файла. Замечу, что можно представить, что в нашем объекте файла есть указатель на текущую читаемую позицию. При открытии файла, она ставится в самое начало. По мере чтения, этот указатель сдвигается. Таким образом, если выполнять read(n) несколько раз подряд, мы будем получать не первые n символов, а каждый раз новые, n символов.

Если n явно не указать, то считается весь файл целиком (указатель окажется в самом конце файла). Для использования метода read, файл должен быть открыт в режиме для чтения Примечание: чтобы узнать текущее положение указателя внутри файла, можно воспользоваться методом tell(), а чтобы установить указатель в нужное положение pos, используется метод seek(pos)

```
  file = open("russian.txt", "r", encoding="utf8")
#открыли файл, file.tell() == 0, т.е указатель
стоит в самом начале
  text = file.read()                    #считали
весь файл
```

• если файл был открыт в режиме для записи, можно воспользоваться *методом python: write(buffer)* - записывает содержимое буфера в файл. Работа указателя при записи аналогична - он указывает на текущее обрабатываемое место.

Примечание: при записи содержимого в файл, не гарантируется, что все запишется в файл здесь и сейчас, сразу после выполнения

команды. Записываемая информация буферизуется (накапливается) и записывается при накоплении некоторого критического количества. Двоичные файлы буферизуются кусками фиксированного размера. Этот размер определяется эвристикой, пытающейся определить размер блока устройства, на котором находится файл, в случае неудачи использует io.DEFAULT_BUFFER_SIZE. Для многих систем буфер равен 4096 или 8192 байт. Содержимое принудительно записывается в файл при его закрытии. Также для принудительной записи в файл можно воспользоваться методом flush() - он просто записывает содержимое текущего буффера в файл здесь и сейчас.

Следует сказать, что открытый в любом режиме файл после его использования нужно обязательно закрывать. Делается это методом close(). После его выполнения работа с файлом будет корректно завершена, но с нашим объектом файла работать уже тоже будет нельзя - при необходимости повторной работы с файлом нужно снова его открывать при помощи open.

```
file = open("some_data.txt")
text = file.read()
file.close()
#дальше работаем с text, если надо
```

Но вдруг в процессе выполнения нашей программы произойдет критическая ошибка и программа завершит свое выполнение, а мы, например, записывали в файл какую-то информацию? Верно, вполне возможно, что последняя добавленная информация в файл так и не запишется. Чтобы избежать такой ситуации, ну и чтобы просто не забывать вовремя вызывать close() используется конструкция with:

```
with open("text.txt", "w") as out:  #в out теперь находится ссылка на наш объект файла, как если бы было просто out = open("text.txt", "w")
      for i in range(100):
```

```
        out.write("А я запишу все эти строки в
влюбом случае\n") #записываем 100 одинаковых
строчек
```

Конструкция with используется для того, чтобы гарантировать, что критические действия будут выполнены в любом случае, ее можно использовать и в некоторых других случаях, но в контексте открытия файлов она используется чаще всего.

Через конструкцию with можно открывать сразу несколько файлов:

```
  with open("input.txt", "r") as input,
open("output.txt", "w") as output:
      output.write(input.read())      #скопировали
содержимое input в output
```

- Чтобы считать из файла целую строку, используется метод *readline(max_len)*. Если указать параметр max_len, то будут считаны максимум max_len символов

```
  with open("text.txt", "r") as file:
      print(file.readline())  #считали и вывели
первую строку файла
```

На самом деле у нашего объекта файла есть итератор, поэтому перебирать строки внутри файла можно с его помощью:

```
  with open("text.txt", "r") as file:
      for line in file:
          print(line)
```

Такой способ чтения наиболее удобен для построчного чтения

Работа с файловой системой

Взаимодействие с файлами не ограничивается только самими файлами, нам часто приходится работать и с папками. Для этого используются библиотеки os и os.path. Они связаны с операционной

системой компьютера и позволяют взаимодействовать с файловой системой.

Все папки директории

os.listdir(dir) перечисляет файлы и папки в указанной директории dir. Если вызвать эту функцию без аргументов, она вернет файлы и папки текущей рабочей директории.

Текущая папка

Относительные пути строятся относительно текущей папки. Чтобы получить абсолютный путь файла из относительного, используется функция os.path.abspath(file_path). Чтобы узнать, какая папка является текущей, можно вызвать функцию os.getcwd(). Для смены текущей папки используется os.chdir(new_dir).

Проверка существования файла или папки и определение, является ли имя файлом или папкой

os.path.exists(file_name) проверяет, существует ли указанный файл (или директория) file_name.

Чтобы проверить, является ли данное имя name файлом или папкой, можно воспользоваться функциями os.isdir(name) или os.isfile(name), которые возвращают True или False.

Рекурсивный обход папок

Одной из самых интересных и мощных функций является функция os.walk(dir) - она позволяет рекурсивно пройтись по всем папкам, подпапкам, их подпапкам и так далее. На самом деле она возвращает генератор (последовательность элементов). Каждый элемент представляет собой кортеж из 3х элементов. Первый элемент - строковое представление директории текущей директории, которую просматривает функция. Вторым элементом - список всех подпапок данной директории, а третьим - список всех файлов этой директории.

for current_dir, dirs, files in os.walk("."): #передаем в качестве аргумента текущую директорию - означает именно ее)

print(current_dir, dirs, files) #выведем, что получается

Копирование файлов

Копировать файлы можно при помощи функции copy из модуля shutil

```
shutil.copy("input.txt", "output.txt")
```

Копировать папки можно с помощью copytree из того же модуля:

```
shutil.copytree("test", "test/test2") #Скопирует папку test внутрь неё самой же в подпапку test2
```

Многие другие функции для работы с файлами и папками вы сможете найти в модулях os и shutil. Теперь вы знаете, где искать нужный функционал.

Распространенные форматы текстовых данных

csv

csv является табличным форматом. В нем содержатся значения, разделенные запятой (Comma-Separated Values). Например,

first name,last name,module1,module2,module3
Nikolay,Neznaev,0,20,10
Stepan,Sharyashiy,100,99.5,100

Для работы с csv файлами можно воспользоваться библиотекой csv:

```
import csv
with open("example.csv", "r") as file:
    reader = csv.reader(file) #На основе открытого файла получаем объект из библиотеки csv
    for row in reader:
        print(row) #Каждая строка - список значений
```

В csv.reader параметром delimeter можно передать разделитель значений, таким образом разделяющим символом в файле csv может быть не только запятая.

Для изолирования некоторых значений можно пользоваться двойными кавычками. Библиотека csv учитывает различные мелочи, такие как строки с содержащимися в ней запятыми и переносами строки, различные разделители, поэтому ее использование целесообразнее splita по разделителю.

Для записи значений в csv формате используется csv.writer:

```
import csv
students = [
            ["Greg", "Lebovskiy", 70, 80, 90,
"Good job, Greg!"],
            ["Nick", "Shalopaev", 10, 50, 45,
"Shalopaev, you should study better!"]
            ]
with open("example.csv", "a") as file:
    writer = csv.writer(file)               #На
основе открытого файла получаем объект из
библиотеки csv
    for student in students:
        writer.writerow(student)
#Записываем строку
    #Вместо цикла выше мы могли сразу записать
все через writer.writerows(students)
```

JSON

JSON (JavaScript Object Notation) - простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Впервые он был придуман и использован в JavaScript для хранения структур и классов, но быстро обрел свою популярность и вышел за пределы своего родителя.

JSON основан на двух структурах данных:

1. Коллекция пар ключ/значение. В разных языках, эта концепция реализована как объект, запись, структура, словарь, хэш, именованный список или ассоциативный массив.

2. Упорядоченный список значений. В большинстве языков это реализовано как массив, вектор, список или последовательность.

Это универсальные структуры данных. Почти все современные языки программирования поддерживают их в какой-либо форме. Логично предположить, что формат данных, независимый от языка программирования, должен быть основан на этих структурах.

Объекты в формате SJON хранятся как словари в Python, но с некоторыми деталями: во первых, ключом в json-объекте может быть только строка, значения True и False пишутся с маленькой буквы, значению None соответствует значение null, строки хранятся только внутри двойных кавычек.

Для удобной работы с json файлами в языке python можно использовать библиотеку json

Например:

```
import json

student1 = {
    "full_name" : "Greg Martin",
    "scores" : [100, 85, 94],
    "certificate" : True,
    "comment": "Great job, Greg!"
}

student2 = {
    "full_name" : "John Price",
    "scores" : [0, 10, 0],
    "certificate" : False,
```

```
      "comment": "Guns aren't gonna help you here,
captain!"
  }

  data = [student1, student2]

  print(json.dumps(data,                    indent=4,
sort_keys=True))  #Делаем  отступы  в  4  пробела,
сортируем ключи в алфавитном порядке
```

Для получения строкового представления объекта в формате json можно использовать json.dumps(data, **parrams) с различными вспомогательными настройками (пробелы, сортировка и др.)

Для записи в файл можно воспользоваться json.dump(data, file_obj, **params):

```
  with open("output.json", "w") as out:
      json.dump(data,        out,        indent=4,
sort_keys=True)
```

Для получения объекта python на основе его срокового представления можно воспользоваться функцией json.loads или json.load для считывания из файла:

```
  json_str    =    json.dumps(data,    indent=4,
sort_keys=True)       #получение       строкового
представления json
  data_again  =  json.loads(json_str)   #получаем
объект python
  print(sum(data_again[0]["scores"])) #убедимся в
кореектном считывании: посчитаем сумму баллов у
первого студента

  with open("output.json") as file:
     data_from_file  =  json.load(file)   #считаем
объект из файла
```

```
    print(sum(data_from_file[0]["scores"]))
#аналогично посчитаем сумму баллов
```

Работа с xml файлами в python

Импорт модуля ElementTree

```
import xml.etree.ElementTree as ET
```

Функция Element () используется для создания элементов XML

```
p=ET.Element('parent')
```

Функция SubElement (), используемая для создания вложенных элементов в элементе give

```
c = ET.SubElement(p, 'child1')
```

Функция dump() используется для вывода элементов xml.

```
ET.dump(p)

# <parent><child1 /></parent>
```

Если вы хотите сохранить в файл, создайте дерево XML с функцией ElementTree() и сохраните в файл, используя метод write()

```
tree = ET.ElementTree(p)
tree.write("sample.xml")
```

Функция Comment() используется для вставки комментариев в XML-файл.

```
comment = ET.Comment('user comment')
p.append(comment)   #этот   комментарий   будет
добавлен к родительскому элементу
```

```
ET.dump(p)

# <parent><child1 /><!--user comment--></parent>
```

Изменение файла XML

Импортируйте модуль ElementTree и откройте файл XML, получите элемент XML

```
import xml.etree.ElementTree as ET
tree = ET.parse('sample.xml')
root=tree.getroot()
element = root[0] #получите первого ребенка родительского корня
element

# <Element 'child1' at 0x7fa5806a8b90>
```

Элементом объекта можно управлять, изменяя его поля, добавляя и изменяя атрибуты, добавляя и удаляя дочерние элементы

```
element.set('attribute_name', 'attribute_value') #установите артрибут xml элементу
element.text="string_text"
```

Если вы хотите удалить элемент, используйте метод Element.remove()

```
root.remove(element)
```

Метод ElementTree.write(), используемый для вывода объекта XML в файлы XML.

```
tree.write('sample.xml')
```

Открытие и чтение больших файлов XML с помощью iterparse (инкрементальный анализ)

Иногда мы не хотим загружать весь XML-файл, чтобы получить необходимую нам информацию. В этих случаях полезно постепенно загружать соответствующие разделы и затем удалять их, когда мы закончим. С помощью функции iterparse вы можете редактировать дерево элементов, которое хранится при разборе XML.

Импортируйте объект ElementTree:

```
import xml.etree.ElementTree as ET
```

Откройте файл .xml и переберите все элементы:

```
for event, elem in ET.iterparse("yourXMLfile.xml"):

# ... сделайте что-нибудь ...
```

Кроме того, мы можем искать только определенные события, такие как начальный / конечный теги или пространства имен. Если эта опция не указана (как указано выше), возвращаются только события «end»:

```
events=("start", "end", "start-ns", "end-ns")
for event, elem in ET.iterparse("yourXMLfile.xml", events=events):

# ... сделайте что-нибудь ...
```

Вот полный пример, показывающий, как очистить элементы из дерева в памяти, когда мы закончим с ними:

```
  for           event,           elem           in
ET.iterparse("yourXMLfile.xml",
events=("start","end")):
      if elem.tag == "record_tag" and event ==
"end":
          print elem.text
          elem.clear()

  #    ... сделайте что-нибудь другое ...
```

Открытие и чтение с помощью ElementTree

Импортируйте объект ElementTree, откройте соответствующий XML-файл и получите корневой тег:

```
import xml.etree.ElementTree as ET
tree = ET.parse("yourXMLfile.xml")
root = tree.getroot()
```

Есть несколько способов поиска по дереву. Сначала по итерации:

```
for child in root:
   print(child.tag, child.attrib)
```

В противном случае вы можете ссылаться на определенные места, такие как список:

```
print(root[0][1].text)
```

Для поиска конкретных тегов по имени, используйте .find или .findall:

```
print(root.findall("myTag"))
print(root[0].find("myOtherTag"))
```

Поиск в XML с помощью XPath

Начиная с версии 2.7 ElementTree имеет лучшую поддержку XPath запросов. XPath - это синтаксис, позволяющий вам перемещаться по XML, как SQL используется для поиска в базе данных. Как find и

findall функции поддержки XPath. Xml ниже будет использоваться для этого примера

```
  <Catalog>
      <Books>
          <Book id="1" price="7.95">
              <Title>Мечтают ли андроиды об
электроовцах?</Title>
              <Author>Philip K. Dick</Author>
          </Book>
          <Book id="5" price="5.95">
              <Title>The Colour of Magic</Title>
              <Author>Terry Pratchett</Author>
          </Book>
          <Book id="7" price="6.95">
              <Title>The Eye of The World</Title>
              <Author>Robert Jordan</Author>
          </Book>
      </Books>
  </Catalog>
```

Поиск всех книг:

```
  import xml.etree.cElementTree as ET
  tree = ET.parse('sample.xml')
  tree.findall('Books/Book')
```

Поиск книги с названием «Цвет магии»:

```
  tree.find("Books/Book[Title='The Colour of
Magic']")
  # всегда используйте '' в правой стороне
сравнения
```

```
  Поиск книги с id = 5:
  tree.find("Books/Book[@id='5']")
  # поиск с xml атрибутами должен иметь '@' перед
именем
```

Поиск второй книги:

```
  tree.find("Books/Book[2]")
  # индексы начинаются с 1, не с 0
```

Поиск последней книги:

```
  tree.find("Books/Book[last()]")
  # 'last' единственная xpath функция позволенная
в ElementTree
```

Поиск всех авторов:

```
  tree.findall(".//Author")
  # поиск с // должен использовать родственный
путь
```

ЗАДАНИЕ НА ДОМАШНЮЮ РАБОТУ

Задание №1 выполняется по варианту.

Для задания №2 создать реализацию репозитория для работы с файлами в формате csv, json или xml в зависимости от варианта.

В каждом задании обработать ошибки, которые могут возникнуть при использовании программы.

Вариант 1

Задача 1.

Дана в архиве файловая структура, состоящая из директорий и файлов.

Необходимо распаковать этот архив (средствами языка python), и затем найти в данной в файловой структуре все директории, в которых есть хотя бы один файл с расширением “.py”.

Ответом на данную задачу будет являться файл со списком таких директорий, отсортированных в лексикографическом порядке.

Задача 2.

Создать файл в формате csv, содержащий сведения о книгах в библиотеке. Структура записи: шифр книги, автор, названия, год издания, местоположения (номер стеллажа, полка).

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по шифру XXX);

2 - добавить новую запись;

3 - изменить запись (по шифру XXX);

4 - получить информацию о книге с шифром XXX.

5 – отсортировать книги по:

- Году издания
- Названию (по алфавиту)
- Автору (по алфавиту)

6 – сохранить в отдельный файл и вывести на экран все книги, которые находятся на заданном стеллаже и полке.

Вариант 2

Задача 1.

Выберите любую папку на своем компьютере, имеющую вложенные директории.

Выведите на печать в терминал ее содержимое, как и всех подкаталогов

Задача 2.

Создать файл в формате csv, содержащий сведения о студентах. Структура записи: номер студенческого, ФИО студента, год поступления, специальность, курс.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по номеру студенческого);

2 - добавить новую запись;

3 - изменить запись (по номеру студенческого);

4 - получить информацию.

5 – отсортировать студентов по:

- Году поступления
- ФИО (по алфавиту)

– Специальности
– Курсу

6 – сохранить в отдельный файл и вывести на экран всех студентов, обучающихся на заданной специальности в алфавитном порядке.

Вариант 3

Задача 1.

Дана в архиве файловая структура, состоящая из директорий и файлов.

Необходимо распаковать этот архив (средствами языка python), и затем найти в данной в файловой структуре все файлы формата “txt”.

Ответом на данную задачу будет являться файл со списком таких файлов, отсортированных в лексикографическом порядке.

Задача 2.

Создать файл в формате xml, содержащий сведения о ноутбуках. Структура записи: модель, видеокарта, процессор, оперативная память.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по модели);

2 - добавить новую запись;

3 - изменить запись (по модели);

4 - получить информацию (по модели).

5 – отсортировать ноутбуки по:

– Видеокарте
– Оперативной памяти

- процессору

6 – сохранить в отдельный файл и вывести на экран все ноутбуки, у которых размер оперативной памяти выше заданной

Вариант 4

Задача 1.

Дана в архиве файловая структура, состоящая из директорий и файлов.

Необходимо распаковать этот архив (средствами языка python), и затем найти в данной в файловой структуре все файлы в которых в названии присутствует слово “file”.

Ответом на данную задачу будет являться файл со списком таких файлов, отсортированных в лексикографическом порядке.

Задача 2.

Создать файл в формате json, содержащий сведения о домашних животных. Структура записи: тип животного, кличка, год рождения, специальность, паспорт.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по паспорту);

2 - добавить новую запись;

3 - изменить запись (по паспорту);

4 - получить информацию (по паспорту или типу животного).

5 – отсортировать животных по:

- Году рождения
- Кличке (по алфавиту)
- Типу животного

6 – сохранить в отдельный файл и вывести на экран всех животных, которые родились раньше заданного года рождения.

Вариант 5

Задача 1.

Выберите любую папку на своем компьютере, имеющую вложенные директории.

Выведите на печать в терминал и сохраните в файл txt названия всех файлов начинающихся на букву а. Заархивируйте данную папку средствами python.

Задача 2.

Создать файл в формате xml, содержащий сведения о автомобилях. Структура записи: гос. номер, марка, цвет, ФИО владельца.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по гос. номеру);

2 - добавить новую запись;

3 - изменить запись (по гос. номеру);

4 - получить информацию (по гос. номеру).

5 – отсортировать автомобили по:

- Марке автомобиля
- Цвету
- ФИО владельца

6 – сохранить в отдельный файл и вывести на экран все автомобили, которыми владеет заданный человек по ФИО.

Вариант 6

Задача 1.

Выберите любую папку на своем компьютере, имеющую вложенные директории.

Выведите на печать в терминал и сохраните в файл txt названия всех файлов с расширением «.csv». Заархивируйте данную папку средствами python.

Задача 2.

Создать файл в формате json, содержащий сведения осотрудниках. Структура записи: id работника, ФИО, должности, размер заработной платы, год рождения.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по id);

2 - добавить новую запись;

3 - изменить запись (по id);

4 - получить информацию (по id).

5 – отсортировать сотрудников по:

- ФИО
- Id
- Должности
- Заработной плате

6 – сохранить в отдельный файл и вывести на консоль сотрудников с заданной должностью и зарплатой выше заданной.

Вариант 7

Задача 1.

Дана в архиве файловая структура, состоящая из директорий и файлов.

Необходимо распаковать этот архив (средствами языка python), и затем найти в данной в файловой структуре все файлы с расширением «.py» и начинающиеся на букву “m”.

Ответом на данную задачу будет являться файл со списком таких файлов, отсортированных в лексикографическом порядке.

Задача 2.

Создать файл в формате csv, содержащий сведения о товарах в интернет магазине. Структура записи: id товара, название, количество товаров на складе, дата поступления.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по id);

2 - добавить новую запись;

3 - изменить запись (по id);

4 - получить информацию (по id).

5 – отсортировать товары по:

- Id
- Названию товара
- Количеству на складе
- Дате поступления

6 – сохранить в отдельный файл и вывести на консоль товары, которые поступили на склад раньше заданной даты.

Вариант 8

Задача 1.

Выберите любую папку на своем компьютере, имеющую вложенные директории.

Выведите на печать в терминал и сохраните в файл txt названия всех файлов с названием «main». Заархивируйте данную папку средствами python.

Задача 2.

Создать файл в формате xml, содержащий сведения о видеокартах. Структура записи: название, объем видеопамяти, тип видеопамяти, частота графического процессора, частота памяти и ширина шины.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по названию);

2 - добавить новую запись;

3 - изменить запись (по названию);

4 - получить информацию (по названию)

5 – отсортировать видеокарты по:

- Названию
- Объему видеопамяти
- Частоте графического процессора
- Типу видеопамяти

6 – сохранить в отдельный файл и вывести на консоль видеокарты, у которых объем видеокарты больше заданно.

Вариант 9

Задача 1.

Дана в архиве файловая структура, состоящая из директорий и файлов.

Необходимо распаковать этот архив (средствами языка python), и затем найти в данной в файловой структуре все файлы с расширением «.py» и начинающиеся на букву “m”.

Ответом на данную задачу будет являться файл со списком таких файлов, отсортированных в лексикографическом порядке.

Задача 2.

Создать файл в формате json, содержащий сведения о странах. Структура записи: название, форма правления, размер территории, население, ввп, официальный язык.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по названию);

2 - добавить новую запись;

3 - изменить запись (по названию);

4 - получить информацию (по названию)

5 – отсортировать страны по:

- Названию
- Форме правления
- Размеру территории
- ВВП

6 – сохранить в отдельный файл и вывести на консоль страну с максимальным ВВП и страну с самой большой территорией.

Вариант 10

Задача 1.

Выберите любую папку на своем компьютере, имеющую вложенные директории.

Выведите на печать в терминал и сохраните в файл txt названия всех файлов, которые содержат введенное с консоли слово. Заархивируйте данную папку средствами python.

Задача 2.

Создать файл в формате csv, содержащий сведения о покупателях в интернет магазине. Структура: id, ФИО, дата рождения, общая сумма покупок, количество заказов..

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по id);

2 - добавить новую запись;

3 - изменить запись (по id);

4 - получить информацию (по id)

5 – отсортировать покупателей по:

- ФИО
- Дате рождения
- Общей сумме покупок
- Количеству заказов

6 – сохранить в отдельный файл и вывести на консоль покупателя с максимальной суммой покупок и покупателя с максимальным количеством заказов.

Вариант 11

Задача 1.

Дана в архиве файловая структура, состоящая из директорий и файлов.

Необходимо распаковать этот архив (средствами языка python), и затем найти в данной файловой структуре все директории, в которых есть хотя бы один файл с заданным с консоли расширением.

Ответом на данную задачу будет являться файл со списком таких директорий, отсортированных в лексикографическом порядке.

Задача 2.

Создать файл в формате xml, содержащий сведения о книгах в библиотеке. Структура записи: шифр книги, автор, названия, год издания, местоположения (номер стеллажа, полка).

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по шифру XXX);

2 - добавить новую запись;

3 - изменить запись (по шифру XXX);

4 - получить информацию о книге с шифром XXX.

5 – отсортировать книги по:

- Году издания
- Названию (по алфавиту)
- Автору (по алфавиту)

6 – сохранить в отдельный файл и вывести на экран все книги, которые находятся на заданном стеллаже и полке.

Вариант 12

Задача 1.

Выберите любую папку на своем компьютере, имеющую вложенные директории.

Выведите на печать в терминал ее содержимое, как и всех подкаталогов

Заархивируйте данную папку средствами python.

Задача 2.

Создать файл в формате xml, содержащий сведения о студентах. Структура записи: номер студенческого, ФИО студента, год поступления, специальность, курс.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по номеру студенческого);

2 - добавить новую запись;

3 - изменить запись (по номеру студенческого);

4 - получить информацию.

5 – отсортировать студентов по:

- Году поступления
- ФИО (по алфавиту)
- Специальности
- Курсу

6 – сохранить в отдельный файл и вывести на экран всех студентов, обучающихся на заданной специальности в алфавитном порядке.

Вариант 13

Задача 1.

Дана в архиве файловая структура, состоящая из директорий и файлов.

Необходимо распаковать этот архив (средствами языка python), и затем найти в данной в файловой структуре все файлы с названием “text”.

Ответом на данную задачу будет являться файл со списком таких файлов, отсортированных в лексикографическом порядке.

Задача 2.

Создать файл в формате json, содержащий сведения о ноутбуках. Структура записи: модель, видеокарта, процессор, оперативная память.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по модели);

2 - добавить новую запись;

3 - изменить запись (по модели);

4 - получить информацию (по модели).

5 – отсортировать ноутбуки по:

- Видеокарте
- Оперативной памяти
- процессору

6 – сохранить в отдельный файл и вывести на экран все ноутбуки, которые имеют заданный процессор.

Вариант 14

Задача 1.

Дана в архиве файловая структура, состоящая из директорий и файлов.

Необходимо распаковать этот архив (средствами языка python), и затем найти в данной в файловой структуре все файлы в которых в названии присутствует слово “cat ”.

Ответом на данную задачу будет являться файл со списком таких файлов, отсортированных в лексикографическом порядке.

Задача 2.

Создать файл в формате csv, содержащий сведения о домашних животных. Структура записи: тип животного, кличка, год рождения, специальность, паспорт.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по паспорту);

2 - добавить новую запись;

3 - изменить запись (по паспорту);

4 - получить информацию (по паспорту или типу животного).

5 – отсортировать животных по:

- Году рождения
- Кличке (по алфавиту)
- Типу животного

6 – сохранить в отдельный файл и вывести на экран всех животных, которые родились позже заданного года рождения.

Вариант 15

Задача 1.

Выберите любую папку на своем компьютере, имеющую вложенные директории.

Выведите на печать в терминал и сохраните в файл txt названия всех файлов начинающихся на букву «s». Заархивируйте данную папку средствами python.

Задача 2.

Создать файл в формате json, содержащий сведения о автомобилях. Структура записи: гос. номер, марка, цвет, ФИО владельца.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по гос. номеру);

2 - добавить новую запись;

3 - изменить запись (по гос. номеру);

4 - получить информацию (по гос. номеру).

5 – отсортировать автомобили по:

- Марке автомобиля
- Цвету
- ФИО владельца

6 – сохранить в отдельный файл и вывести на экран все автомобили с заданным цветом

Вариант 16

Задача 1.

Выберите любую папку на своем компьютере, имеющую вложенные директории.

Выведите на печать в терминал и сохраните в файл txt названия всех файлов с расширением «.csv». Заархивируйте данную папку средствами python.

Задача 2.

Создать файл в формате csv, содержащий сведения о сотрудниках. Структура записи: id работника, ФИО, должности, размер заработной платы, год рождения.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по id);

2 - добавить новую запись;

3 - изменить запись (по id);

4 - получить информацию (по id).

5 – отсортировать сотрудников по:

- ФИО
- Id
- Должности
- Заработной плате

6 – сохранить в отдельный файл и вывести на консоль сотрудников с возрастом выше заданного.

Вариант 17

Задача 1.

Дана в архиве файловая структура, состоящая из директорий и файлов.

Необходимо распаковать этот архив (средствами языка python), и затем найти в данной в файловой структуре все файлы с расширением «.txt» и начинающиеся на букву “l”.

Ответом на данную задачу будет являться файл со списком таких файлов, отсортированных в лексикографическом порядке.

Задача 2.

Создать файл в формате csv, содержащий сведения о товарах в интернет магазине. Структура записи: id товара, название, количество товаров на складе, дата поступления.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по id);

2 - добавить новую запись;

3 - изменить запись (по id);

4 - получить информацию (по id).

5 – отсортировать товары по:

- Id
- Названию товара
- Количеству на складе
- Дате поступления

6 – сохранить в отдельный файл и вывести на консоль товары, которые поступили на склад позже заданной даты.

Вариант 18

Задача 1.

Выберите любую папку на своем компьютере, имеющую вложенные директории.

Выведите на печать в терминал и сохраните в файл txt названия всех файлов с названием «main.py». Заархивируйте данную папку средствами python.

Задача 2.

Создать файл в формате xml, содержащий сведения о видеокартах. Структура записи: название, объем видеопамяти, тип видеопамяти, частота графического процессора, частота памяти и ширина шины.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по названию);

2 - добавить новую запись;

3 - изменить запись (по названию);

4 - получить информацию (по названию)

5 – отсортировать видеокарты по:

- Названию
- Объему видеопамяти
- Частоте графического процессора
- Типу видеопамяти

6 – сохранить в отдельный файл и вывести на консоль видеокарты, у которых частота памяти видеокарты ниже заданной

Вариант 19

Задача 1.

Дана в архиве файловая структура, состоящая из директорий и файлов.

Необходимо распаковать этот архив (средствами языка python), и затем найти в данной файловой структуре все файлы, которые содержат введенное с консоли слово..

Ответом на данную задачу будет являться файл со списком таких файлов, отсортированных в лексикографическом порядке.

Задача 2.

Создать файл в формате json, содержащий сведения о странах. Структура записи: название, форма правления, размер территории, население, ввп, официальный язык.

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по названию);

2 - добавить новую запись;

3 - изменить запись (по названию);

4 - получить информацию (по названию)

5 – отсортировать страны по:

- Названию
- Форме правления
- Размеру территории
- ВВП

6 – сохранить в отдельный файл и вывести на консоль страну с минимальным ВВП и страну с самой большой территорией.

Вариант 20

Задача 1.

Выберите любую папку на своем компьютере, имеющую вложенные директории.

Выведите на печать в терминал и сохраните в файл txt названия всех файлов, которые содержат введенное с консоли слово. Заархивируйте данную папку средствами python.

Задача 2.

Создать файл в формате csv, содержащий сведения о покупателях в интернет магазине. Структура: id, ФИО, дата рождения, общая сумма покупок, количество заказов..

Предусмотреть возможность корректировки файла по вводимому коду корректировки, например:

1 - удалить запись (по id);

2 - добавить новую запись;

3 - изменить запись (по id);

4 - получить информацию (по id)

5 – отсортировать покупателей по:

- ФИО
- Дате рождения
- Общей сумме покупок
- Количеству заказов

6 – сохранить в отдельный файл и вывести на консоль покупателя с минимальной суммой покупок и покупателя с минимальным количеством заказов.

КОНТРОЛЬНЫЕ ВОПРОСЫ И ЗАДАНИЯ

1. Расскажите о режимах открытия файла
2. Опишите команды с помощью которых происходит чтение и запись в файл
3. Опишите основные команды для работы с файловой системой
4. Расскажите преимущества открытия файла через конструкцию with
5. Приведите пример файла csv
6. Приведите пример файла xml
7. Приведите пример файла json
8. Опишите команды с помощью которых происходит чтение и запись в файл json
9. Опишите команды с помощью которых происходит чтение и запись в файл xml
10. Опишите команды с помощью которых происходит чтение и запись в файл csv
11. Приведите пример чтения и записи из файла формата “.txt”
12. Опишите процесс архивирования директории с помощью python
13. Опишите процесс разархивирования директории с помощью python

ФОРМА ОТЧЕТА ПО ДОМАШНЕЙ РАБОТЕ

На выполнение домашней работы отводится 21 академических часа: 20 часов на выполнение и сдачу домашней работы и 1 час на подготовку отчета. Отчет на защиту предоставляется в печатном виде.

Порядок выполнения:

8. Изучить теоретический материал.
9. Получить вариант у преподавателя.
10. Разработать графический интерфейс согласно варианту.
11. Выполнить тестирование программы.
12. Продемонстрировать работу программы преподавателю.
13. Оформить отчет.
14. Защитить выполненную работу у преподавателя.

СОДЕРЖАНИЕ ОТЧЁТА

1. Титульный лист.
2. Цель, задачи работы.
3. Формулировка задания.
4. UML-диаграмма классов, используемых в программе.
5. Листинг программы.
6. Результаты выполнения программы.
7. Выводы по работе в целом.

ОСНОВНАЯ ЛИТЕРАТУРА

1. Васильев А.Н. Python на примерах [Электронный ресурс]: практический курс по программированию/ Васильев А.Н.— Электрон. текстовые данные.— СПб.: Наука и Техника, 2017.— 432 с.— Режим доступа: http://www.iprbookshop.ru/73043.html

2. Буйначев С.К. Основы программирования на языке Python [Электронный ресурс]: учебное пособие/ Буйначев С.К., Боклаг Н.Ю.— Электрон. текстовые данные.— Екатеринбург: Уральский федеральный университет, ЭБС АСВ, 2014.— 92 с.— Режим доступа: http://www.iprbookshop.ru/66183.html

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

4. Хахаев, И.А. Практикум по алгоритмизации и программированию на Python [Электронный ресурс] : учебное пособие / И.А. Хахаев. — Электрон. дан. — Москва : , 2016. — 178 с. — Режим доступа: https://e.lanbook.com/book/100377

5. Сузи Р.А. Язык программирования Python [Электронный ресурс]/ Сузи Р.А.— Электрон. текстовые данные.— М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016.— 350 с.— Режим доступа: http://www.iprbookshop.ru/52211.html

Электронные ресурсы:

5. Электронно-библиотечная система http://www.iprbookshop.ru

6. Электронно-библиотечная система http://e.lanbook.com

7. Руководство по языку программирования Java [Электронный ресурс]//Сайт о программировании “METANIT”: сайт – Режим доступа: https://metanit.com/python/tutorial/1.1.php