Государственное учреждение образования

«Гимназия г. Наровли»

Проект

**Написание программы для автоматического выполнения теста Руфье**

#### Направление «Программирование на языке Python»

Выполнил

Радкевич Станислав Васильевич,

учащийся XI класса

Наровля, 2023

**Оглавление**

Введение…………………………………………………………………………...3

Основная часть…………………………………………………………………….4

Заключение…………………..…………………………………………………...12

Список использованных источников…………………………………………...13

**Введение**

Python –  язык программирования с динамической типизацией и автоматическим управлением памятью. Язык является полностью объектно-ориентированным, а это значит, что всё является объектами. Необычной особенностью языка является выделение блоков кода пробельными отступами. Идея создания языка появилась в конце 1980-х годов, а разработка его реализации началась в 1989 году Гвидо ван Россумом.

По сравнению с другими языками Python намного проще в понимании и написании, так как функции в нём имеют названия от слов на английском. Это самый используемый язык в наше время. На нём пишут алгоритмы программ с машинным обучением и аналитические приложения. С помощью него обслуживают хранилища данных и облачные сервисы. Именно поэтому я выбрал этот язык для написания своего проекта.

Человеку чтобы быть счастливым нужно в первую очередь здоровым. Но как определить насколько ты здоров? Для этого был создан тест Руфье, который предназначен для определения степени физической подготовки. Его суть заключается в том, что нужно замерить пульс после минуты покоя в положении лёжа (), после в течение 45 секунд сделать 30 приседаний, затем на протяжении одной минуты требуется замерять пульс в первые () и последние ( 15 секунд. После проведённых измерений нужно подставить полученные значения в формулу по нахождению индекса Руфье.

Результаты оцениваются по величине индекса от 0 до 15:

* меньше 3 — хорошая работоспособность;
* 3–6 — средняя;
* 7–9 — удовлетворительная;
* 10–14 — плохая (средняя сердечная недостаточность);
* 15 и выше — сердечная недостаточность.

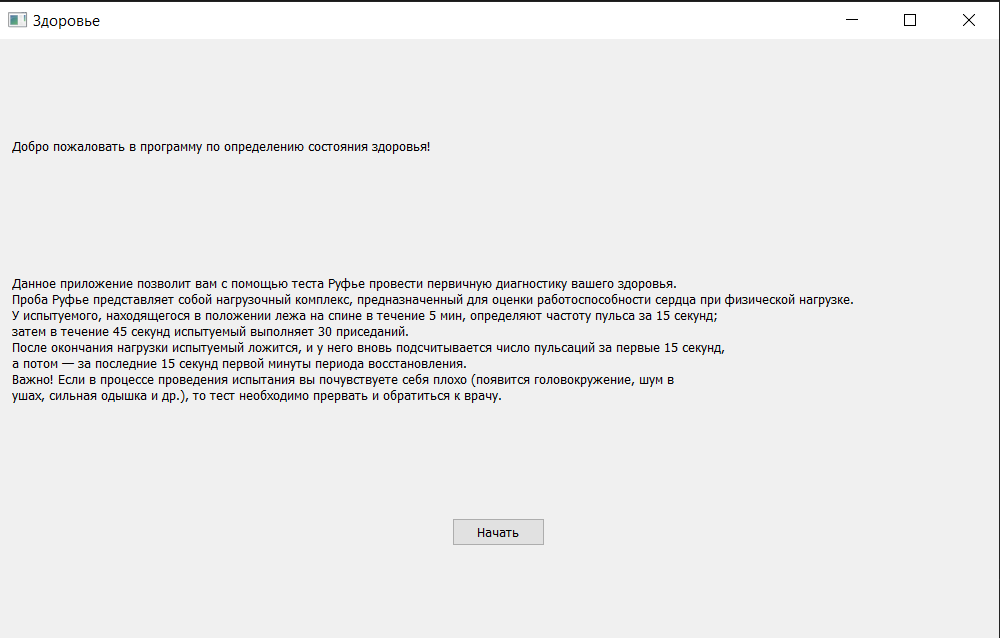
*Цель работы*: написание программы для автовыполнения теста Руфье.

*Задачи*:

* изучить необходимую информацию;
* написать код программы;
* сделать вывод о работоспособности и практичности программы.

*Практическая значимость*: работу можно использовать на  
уроках физической культуры и здоровья и на медкомиссиях для простого и удобного подсчёта индекса физической подготовки.

**Основная часть**

 Моя программа основывается на том, что при запуске открывается окно, в котором описывается что такое тест Руфье. Внизу находится кнопка «Начать». Код этого окна описывается в файле my\_app.py.

В данных строках создаётся первое окно: название, размер и положение.

def set\_appear(self):

self.setWindowTitle(txt\_title)

self.resize(win\_width, win\_height)

self.move(win\_x, win\_y)

В данных строках создаются виджеты: текст и кнопка.

def initUI(self):

    ''' создаёт графические элементы '''

    self.btn\_next = QPushButton(txt\_next)

    self.hello\_text = QLabel(txt\_hello)

    self.instruction = QLabel(txt\_instruction)

    self.layout = QVBoxLayout()

    self.layout.addWidget(self.hello\_text, alignment = Qt.AlignLeft)

    self.layout.addWidget(self.instruction, alignment = Qt.AlignLeft)

    self.layout.addWidget(self.btn\_next, alignment = Qt.AlignCenter)

    self.setLayout(self.layout)

Здесь задаются функции, которые будет выполнять кнопка «Начать», и привязывают эти функции к кнопке.

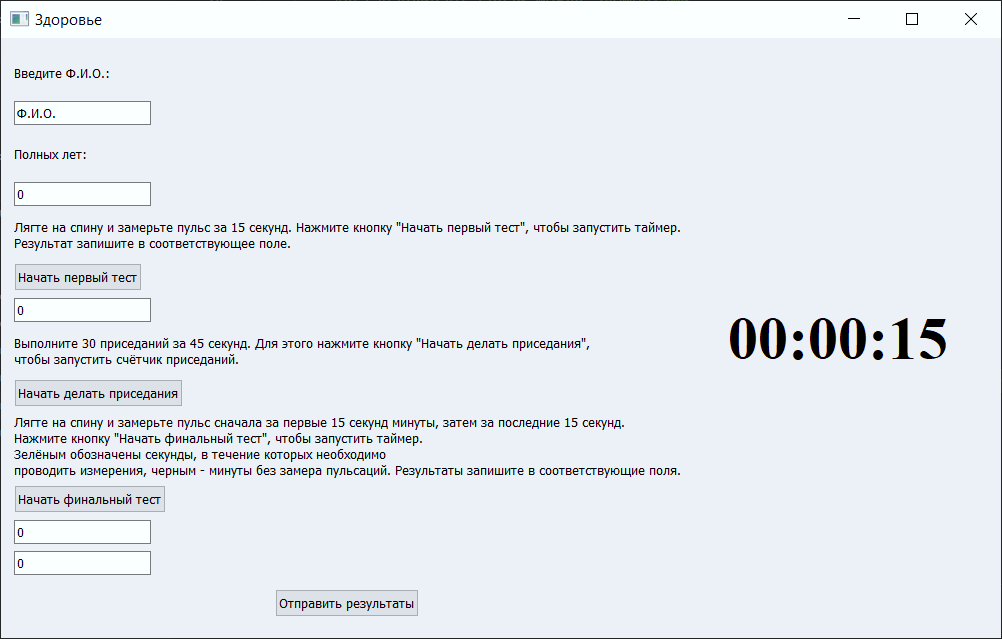
def next\_click(self):

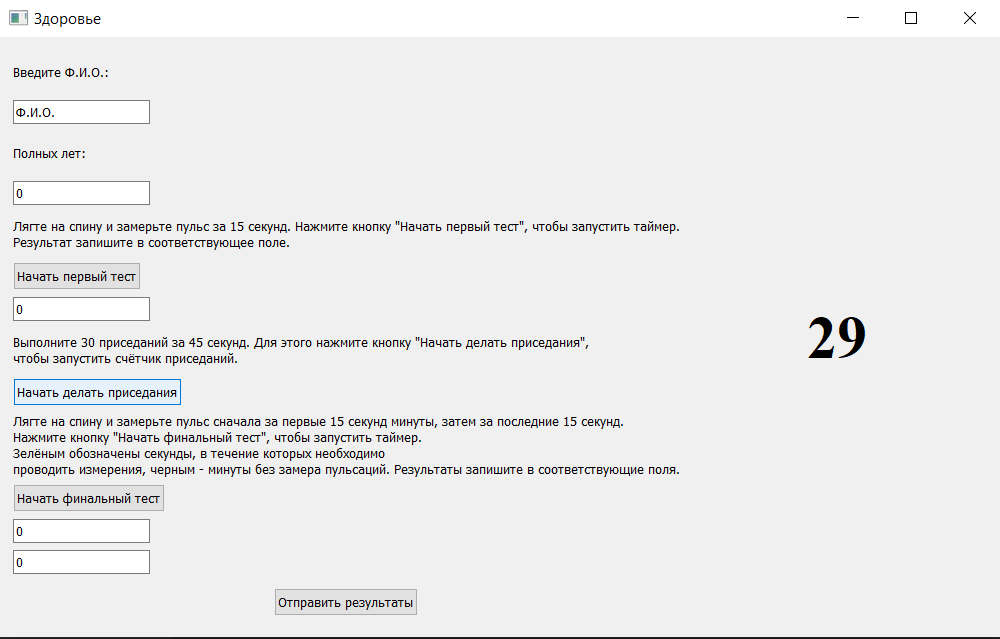
   self.tw = TestWin()

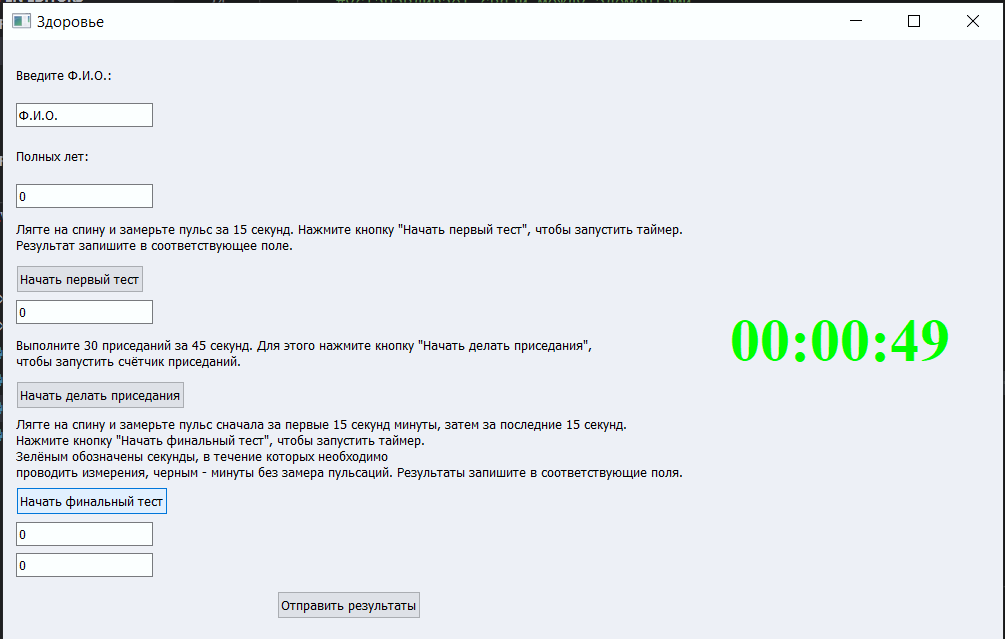
    self.hide()

def connects(self):

    self.btn\_next.clicked.connect(self.next\_click)

 При нажатии кнопки окно меняется на второе. На нём присутствуют описания действий, поля для ввода имени, возраста и значений пульса, и кнопки для начала тестов. Код этого окна находится в файле second\_win.py. Для работы программы надо ввести измерения пульса после 5 минут покоя. Чтобы запустить первый таймер требуется нажать на кнопку «Начать первый тест». Тогда появится обратный отсчёт на 15 секунд. По окончании отсчёта надо полученные данные в соответствуещее поле.

 Для запсука второго таймера нужно нажать на кнопку «Начать делать приседания». Запустится счётчик на 30 приседаний (1 приседание в 1.5 секунды).

 После нажатия третей кнопки «Начать финальный тест» запустится таймер, в котором в первые и последние 15 секунд он будет зелёным, а в остальное время – чёрным. Во время того, как таймер горит зелёным, требуется считать количество биений сердца. Пульс в первые 15 секунд надо записать в первое поле, а пульс за последние 15 секунд нужно написать во второе поле.

В коде программы второе окно создавалось так же как и первое. В данном фрагменте описано создание виджетов второго окна.

def initUI(self):

    ''' создаёт графические элементы '''

    self.btn\_next = QPushButton(txt\_sendresults, self)

    self.btn\_test1 = QPushButton(txt\_starttest1, self)

    self.btn\_test2 = QPushButton(txt\_starttest2, self)

    self.btn\_test3 = QPushButton(txt\_starttest3, self)

    self.text\_name = QLabel(txt\_name)

    self.text\_age = QLabel(txt\_age)

    self.text\_test1 = QLabel(txt\_test1)

    self.text\_test2 = QLabel(txt\_test2)

    self.text\_test3 = QLabel(txt\_test3)

    self.text\_timer = QLabel(txt\_timer)

    self.text\_timer.setFont(QFont("Times", 36, QFont.Bold))

    self.line\_name = QLineEdit(txt\_hintname)

    self.line\_age = QLineEdit(txt\_hintage)

    self.line\_test1 = QLineEdit(txt\_hinttest1)

    self.line\_test2 = QLineEdit(txt\_hinttest2)

    self.line\_test3 = QLineEdit(txt\_hinttest3)

    self.l\_line = QVBoxLayout()

    self.r\_line = QVBoxLayout()

    self.h\_line = QHBoxLayout()

    self.r\_line.addWidget(self.text\_timer, alignment = Qt.AlignCenter)

    self.l\_line.addWidget(self.text\_name, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.line\_name, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.text\_age, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.line\_age, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.text\_test1, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.btn\_test1, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.line\_test1, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.text\_test2, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.btn\_test2, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.text\_test3, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.btn\_test3, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.line\_test2, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.line\_test3, alignment = Qt.AlignLeft)

    self.l\_line.addWidget(self.btn\_next, alignment = Qt.AlignCenter)

    self.h\_line.addLayout(self.l\_line)

    self.h\_line.addLayout(self.r\_line)

    self.setLayout(self.h\_line)

Создание виджетов второго окна.

Здесь задаются функции, выполняемые кнопкой «Отправить результаты». Она переключает второе окно на последнее.

def next\_click(self):

    self.hide()

    self.exp = Experiment(int(self.line\_age.text()), self.line\_test1.text(), self.line\_test2.text(), self.line\_test2.text())

    self.fw = FinalWin(self.exp)

Это функции, запускающие таймер первого теста,

def timer\_test(self):

    global time

    time = QTime(0, 0, 15)

    self.timer = QTimer()

    self.timer.timeout.connect(self.timer1Event)

    self.timer.start(1000)

def timer1Event(self):

    global time

    time = time.addSecs(-1)

    self.text\_timer.setText(time.toString("hh:mm:ss"))

    self.text\_timer.setFont(QFont("Times", 36, QFont.Bold))

    self.text\_timer.setStyleSheet("color: rgb(0,0,0)")

        if time.toString("hh:mm:ss") == "00:00:00":

            self.timer.stop()

Это фунции, запускающие счётчик приседаний.

def timer\_sits(self):

    global time

    time = QTime(0, 0, 30)

    self.timer = QTimer()

    self.timer.timeout.connect(self.timer2Event)

    #одно приседание в 1.5 секунды

    self.timer.start(1500)

def timer2Event(self):

    global time

    time = time.addSecs(-1)

    self.text\_timer.setText(time.toString("hh:mm:ss")[6:8])

    self.text\_timer.setStyleSheet("color: rgb(0,0,0)")

    self.text\_timer.setFont(QFont("Times", 36, QFont.Bold))

    if time.toString("hh:mm:ss") == "00:00:00":

        self.timer.stop()

Это функции, запускающие последний таймер.

def timer\_final(self):

    global time

    time = QTime(0, 1, 0)

    self.timer = QTimer()

    self.timer.timeout.connect(self.timer3Event)

    self.timer.start(1000)

def timer3Event(self):

    global time

    time = time.addSecs(-1)

    self.text\_timer.setText(time.toString("hh:mm:ss"))

    if int(time.toString("hh:mm:ss")[6:8]) >= 45:

        self.text\_timer.setStyleSheet("color: rgb(0,255,0)")

    elif int(time.toString("hh:mm:ss")[6:8]) <= 15:

        self.text\_timer.setStyleSheet("color: rgb(0,255,0)")

    else:

        self.text\_timer.setStyleSheet("color: rgb(0,0,0)")

    self.text\_timer.setFont(QFont("Times", 36, QFont.Bold))

    if time.toString("hh:mm:ss") == "00:00:00":

        self.timer.stop()

Здесь привязываются функции всех кнопок второго экрана к самим виджетам.

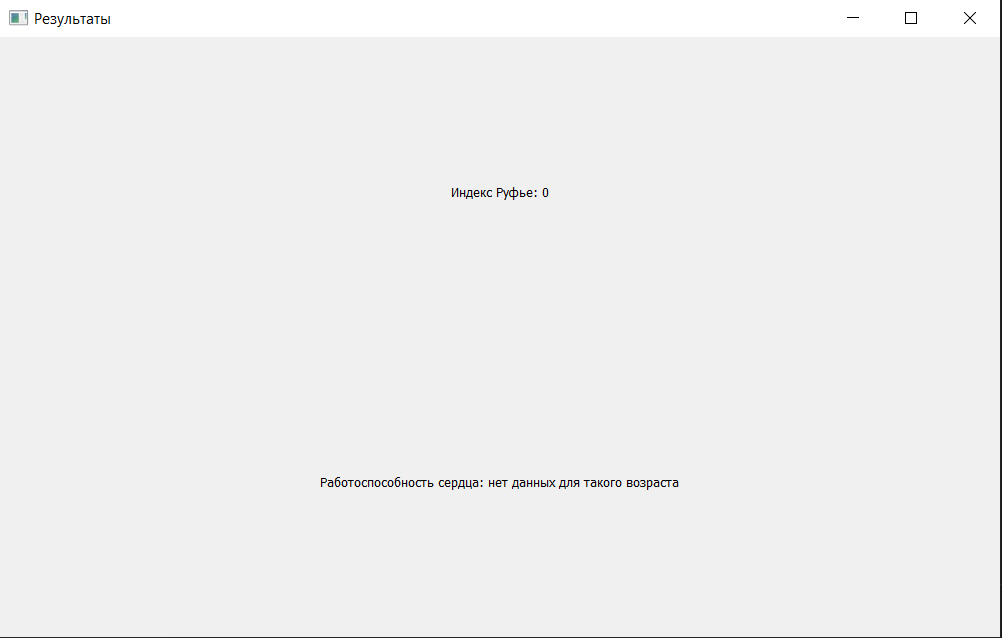
def connects(self):

        self.btn\_next.clicked.connect(self.next\_click)

        self.btn\_test1.clicked.connect(self.timer\_test)

        self.btn\_test2.clicked.connect(self.timer\_sits)

        self.btn\_test3.clicked.connect(self.timer\_final)

 Для вывода результата нажимаем кнопку «Получить результаты». Поя вится последнее третье окно. На нём находится полученный индекс Руфье и вывод о рабоспособности сердца.

В данных строках создаётся третье окно и все элементы на нём.

def set\_appear(self):

    self.setWindowTitle(txt\_finalwin)

    self.resize(win\_width, win\_height)

    self.move(win\_x, win\_y)

def initUI(self):

    ''' создаёт графические элементы '''

    self.work\_text = QLabel(txt\_workheart + self.results())

    self.index\_text = QLabel(txt\_index + str(self.index))

    self.layout\_line = QVBoxLayout()

    self.layout\_line.addWidget(self.index\_text, alignment = Qt.AlignCenter)

    self.layout\_line.addWidget(self.work\_text, alignment = Qt.AlignCenter)

    self.setLayout(self.layout\_line)

В этой функции подсчитывается индекс Руфье и определяется сетепень работоспособности тела относительно введённого возраста.

def results(self):

    if self.exp.age < 7:

        self.index = 0

        return "нет данных для такого возраста"

    self.index = (4 \* (int(self.exp.t1) + int(self.exp.t2) + int(self.exp.t3)) - 200) / 10 "Формула теста Руфье"

    if self.exp.age == 7 or self.exp.age == 8:

        if self.index >= 21:

            return txt\_res1

        elif self.index < 21 and self.index >= 17:

                return txt\_res2

            elif self.index < 17 and self.index >= 12:

                return txt\_res3

            elif self.index < 12 and self.index >= 6.5:

                return txt\_res4

            else:

                return txt\_res5

        if self.exp.age == 9 or self.exp.age == 10:

            if self.index >= 19.5:

                return txt\_res1

            elif self.index < 19.5 and self.index >= 15.5:

                return txt\_res2

            elif self.index < 15.5 and self.index >= 10.5:

                return txt\_res3

            elif self.index < 10.5 and self.index >= 5:

                return txt\_res4

            else:

                return txt\_res5

        if self.exp.age == 11 or self.exp.age == 12:

            if self.index >= 18:

                return txt\_res1

            elif self.index < 18 and self.index >= 14:

                return txt\_res2

            elif self.index < 14 and self.index >= 9:

                return txt\_res3

            elif self.index < 9 and self.index >= 3.5:

                return txt\_res4

            else:

                return txt\_res5

        if self.exp.age == 13 or self.exp.age == 14:

            if self.index >= 16.5:

                return txt\_res1

            elif self.index < 16.5 and self.index >= 12.5:

                return txt\_res2

            elif self.index < 12.5 and self.index >= 7.5:

                return txt\_res3

            elif self.index < 7.5 and self.index >= 2:

                return txt\_res4

            else:

                return txt\_res5

        if self.exp.age >= 15:

            if self.index >= 15:

                return txt\_res1

            elif self.index < 15 and self.index >= 11:

                return txt\_res2

            elif self.index < 11 and self.index >= 6:

                return txt\_res3

            elif self.index < 6 and self.index >= 0.5:

                return txt\_res4

            else:

                return txt\_res5

Многие переменные, использовавшиеся в программе, описаны в файле instr.py. Для написания кода использовался Visual Studio Code

*Примечание*: для работы программы требуется установить библиотеку PyQt5.

**Заключение**

В наше время значительное развитие получают новые технологии, которые помогают человеку облегчить свою жизнь. Но есть и обратная сторона этого развития. Люди всё меньше следят за своим здоровьем, находясь днями у себя дома.

Такие программы, как моя, могут проявить интерес и желание у людей к своему здоровью. Они увидят тест «Насколько ты здоров». И они захотят это проверить. А если увидят плохой результат, то могут задуматься: “Может надо начать заниматься спортом?”. И тем самым помогут своему организму.

**Список использованных источников**

1. Проба Руфье-Диксона https://ru.wikipedia.org/wiki/Проба\_Руфье\_ —\_Диксона - Дата доступа 19.09.2023.
2. Образовательный курс «Алгоритмика».
3. Ответы на различные вопросы находились на сайте Stack OwerFlow <https://stackoverflow.com/questions> - Дата доступа 19.09.2023.