МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение   
высшего образования

**«МОСКОВСКИЙ АВТОМОБИЛЬНО–ДОРОЖНЫЙ   
ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»**

Факультет «Управление»

Кафедра «Автоматизированные системы управления»

09.04.01 – «Интегрированные автоматизированные системы управления в отраслях транспортно‑дорожного комплекса»

Дисциплина: «Технология разработки программного обеспечения*»*

**Отчет**

по лабораторной работе №3  
на тему:

«**Подготовка и проведение тестирования ПО»**

Выполнил:

студент группы 1мБД1 ФИО

Проверил:

Доцент, к.т.н. Баринов К.А.

**Москва  
2023**

Оглавление

[1. Создание и запуск модульных тестов для управляемого кода 3](#_Toc153461915)

[1.1. Создание проекта для тестирования 3](#_Toc153461916)

[1.2. Создание проекта модульного теста, тестового класса и метода теста 3](#_Toc153461917)

[1.3. Сборка и запуск тестов 4](#_Toc153461918)

[1.4. Анализ результатов модульного теста и тестируемого кода 4](#_Toc153461919)

[2. Создание, изменение и обслуживание закодированного теста пользовательского интерфейса 5](#_Toc153461920)

[2.1. Создание приложения tkinter 5](#_Toc153461921)

[2.2. Создание ярлыка для приложения tkinter и закодированные тесты пользовательского интерфейса 6](#_Toc153461922)

[Приложения 8](#_Toc153461923)

[Приложение А. Листинг приложения LR3.py 8](#_Toc153461924)

# Создание и запуск модульных тестов для управляемого кода

## Создание проекта для тестирования

Для проведения тестирования создаю каталог с названием LR3 и размещаю в нём два файла проекта с названиями «**hello\_function.py**» (рис. 1) и «**start.py**» (рис. 2).

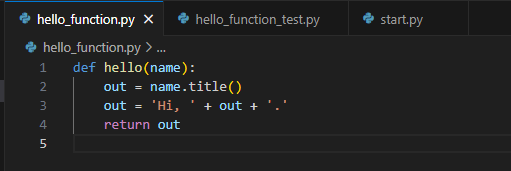


Рисунок . Тестируемая функция hello\_function.py

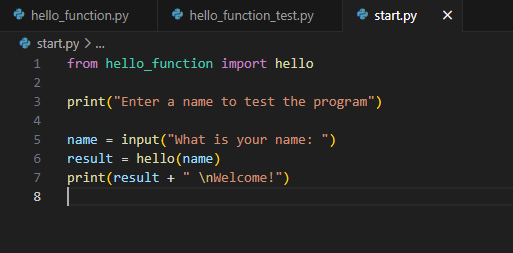


Рисунок . Тестируемый метод

Таким образом получился проект с методами для тестирования. Тестирование будет проводиться на методе «**hello**»

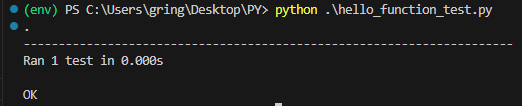
## Создание проекта модульного теста, тестового класса и метода теста

Для реализации тестирования в проект был добавлен файл с названием «**hello\_function\_test.py**» (рис. 3).

  
Рисунок 3. Модульный тест для основного проекта

## Сборка и запуск тестов

Подготовив основной проект и тесты методов проекта, необходимо собрать проект и запустить тестирование. В случае если в основном коде допущена ошибка, то тесты выведут покажут место с ошибкой. Если в основном коде отсутствуют ошибки, то тесты будут пройдены без предупреждений и ошибок (рис. 4).

  
Рисунок 4. Результат тестирования

## Анализ результатов модульного теста и тестируемого кода

Исходя из того, что тест был выполнен успешно, можно сделать вывод что тестируемый метод был написан корректно.

# Создание, изменение и обслуживание закодированного теста пользовательского интерфейса

## Создание приложения tkinter

В уже существующем решение создаю ещё один проект и назову его «LR3.py» (рис. 9, «приложение А»).

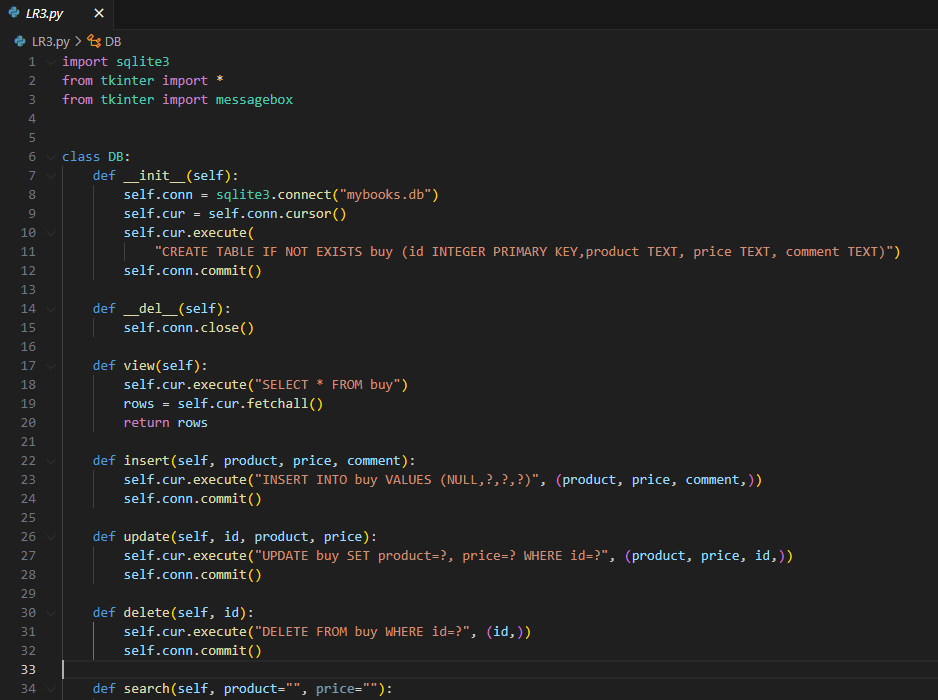
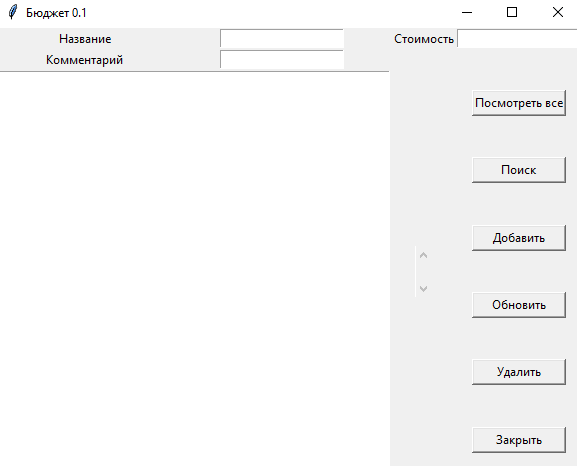


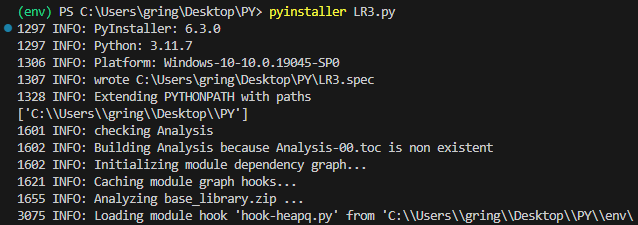
Рисунок . Подготовка пользовательского приложения

Пользовательский интерфейс будет состоять из одного диалогового окна и 6 кнопок (рис. 6).

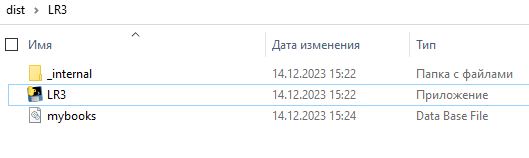
  
Рисунок 6. Пользовательский интерфейс

## Создание ярлыка для приложения tkinter и закодированные тесты пользовательского интерфейса

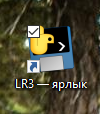
Для создания ярлыка применю команду **pyinstaller LR3.py** (рис. 7).

  
Рисунок 7. Процесс выполнения команды pyinstaller LR3.py

В каталоге с проектом перейду по пути **LR3 → dist → LR3** (рис. 8).

  
Рисунок 8. Переход в дирректорию с приложением

Нажму на файл с именем LR3 правой кнопкой мыши и выберу пункт создать ярлы. Ярлык размещу на рабочем столе (рис. 9).

  
Рисунок 9. Ярлык приложения

# Приложения

## Приложение А. Листинг приложения LR3.py

import sqlite3

from tkinter import \*

from tkinter import messagebox

class DB:

def \_\_init\_\_(self):

self.conn = sqlite3.connect("mybooks.db")

self.cur = self.conn.cursor()

self.cur.execute(

"CREATE TABLE IF NOT EXISTS buy (id INTEGER PRIMARY KEY,product TEXT, price TEXT, comment TEXT)")

self.conn.commit()

def \_\_del\_\_(self):

self.conn.close()

def view(self):

self.cur.execute("SELECT \* FROM buy")

rows = self.cur.fetchall()

return rows

def insert(self, product, price, comment):

self.cur.execute("INSERT INTO buy VALUES (NULL,?,?,?)", (product, price, comment,))

self.conn.commit()

def update(self, id, product, price):

self.cur.execute("UPDATE buy SET product=?, price=? WHERE id=?", (product, price, id,))

self.conn.commit()

def delete(self, id):

self.cur.execute("DELETE FROM buy WHERE id=?", (id,))

self.conn.commit()

def search(self, product="", price=""):

self.cur.execute("SELECT \* FROM buy WHERE product=?", (product,))

rows = self.cur.fetchall()

return rows

db = DB()

def get\_selected\_row(event):

global selected\_tuple

index = list1.curselection()[0]

selected\_tuple = list1.get(index)

e1.delete(0, END)

e1.insert(END, selected\_tuple[1])

e2.delete(0, END)

e2.insert(END, selected\_tuple[2])

e3.delete(0, END)

e3.insert(END, selected\_tuple[3])

def view\_command():

list1.delete(0, END)

for row in db.view():

list1.insert(END, row)

def search\_command():

list1.delete(0, END)

for row in db.search(product\_text.get()):

list1.insert(END, row)

def add\_command():

db.insert(product\_text.get(), price\_text.get(), comment\_text.get())

view\_command()

def delete\_command():

db.delete(selected\_tuple[0])

view\_command()

def update\_command():

db.update(selected\_tuple[0], product\_text.get(), price\_text.get())

view\_command()

window = Tk()

window.title("Бюджет 0.1")

def on\_closing():

if messagebox.askokcancel("", "Закрыть программу?"):

window.destroy()

window.protocol("WM\_DELETE\_WINDOW", on\_closing)

l1 = Label(window, text="Название")

l1.grid(row=0, column=0)

l2 = Label(window, text="Стоимость")

l2.grid(row=0, column=2)

l3 = Label(window, text="Комментарий")

l3.grid(row=1, column=0)

product\_text = StringVar()

e1 = Entry(window, textvariable=product\_text)

e1.grid(row=0, column=1)

price\_text = StringVar()

e2 = Entry(window, textvariable=price\_text)

e2.grid(row=0, column=3)

comment\_text = StringVar()

e3 = Entry(window, textvariable=comment\_text)

e3.grid(row=1, column=1)

list1 = Listbox(window, height=25, width=65)

list1.grid(row=2, column=0, rowspan=6, columnspan=2)

sb1 = Scrollbar(window)

sb1.grid(row=2, column=2, rowspan=6)

list1.configure(yscrollcommand=sb1.set)

sb1.configure(command=list1.yview)

list1.bind('<<ListboxSelect>>', get\_selected\_row)

b1 = Button(window, text="Посмотреть все", width=12, command=view\_command)

b1.grid(row=2, column=3)

b2 = Button(window, text="Поиск", width=12, command=search\_command)

b2.grid(row=3, column=3)

b3 = Button(window, text="Добавить", width=12, command=add\_command)

b3.grid(row=4, column=3)

b4 = Button(window, text="Обновить", width=12, command=update\_command)

b4.grid(row=5, column=3)

b5 = Button(window, text="Удалить", width=12, command=delete\_command)

b5.grid(row=6, column=3)

b6 = Button(window, text="Закрыть", width=12, command=on\_closing)

b6.grid(row=7, column=3)

view\_command()

window.mainloop()