

ФГАОУ ВО "МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"

Лабораторная работа №7

GUI, классы, модуль Tkinter

Вариант №12

по дисциплине:

Основы программирования

Выполнил

студент 1 курса
группы 211-321
Журавлев Д.А.

Проверил

Никишина И.Н.

МОСКВА 2021

Постановка задачи

Задание 1.

Описать запись с именем Note, содержащую поля имя, фамилия, номер телефона, день рождения.

Написать программу выполняющую следующие действия:

- Ввод элементов Note с клавиатуры. Элементы Note упорядочены по дате рождения.
- Вывод информации о человеке, номер телефона которого введен с клавиатуры.
- Запись данных в файл с заданным именем.

Задание 2.

Текст помощи некоторой программы организован как линейный список.

Составить программу, которая обеспечивает:

- Начальное формирование текста помощи
- Вывод текста помощи
- Вывод поясняющего текста для заданного термина.

Теоретическая часть

Библиотека Tkinter установлена в Python в качестве стандартного модуля, поэтому нам не нужно устанавливать что-либо для его использования. Tkinter — очень мощная библиотека. Если вы уже установили Python, можете использовать IDLE, который является интегрированной IDE, поставляемой в Python, эта IDE написана с использованием Tkinter. Звучит круто!

Для начала, следует импортировать Tkinter и создать окно, в котором мы зададим его название:

```
from tkinter import *  
  
window = Tk()  
  
window.title("Добро пожаловать в приложение PythonRu")  
  
window.mainloop()
```

Последняя строка вызывает функцию mainloop. Эта функция вызывает бесконечный цикл окна, поэтому окно будет ждать любого взаимодействия с пользователем, пока не будет закрыто

В случае, если вы забудете вызвать функцию mainloop, для пользователя ничего не отобразится.

Создание виджета Label

Чтобы добавить текст в наш предыдущий пример, мы создадим lbl, с помощью класса Label, например:

```
lbl = Label(window, text="Привет")
```

Затем мы установим позицию в окне с помощью функции grid и укажем ее следующим образом:

```
lbl.grid(column=0, row=0)
```

Настройка размера и шрифта текста

Вы можете задать шрифт текста и размер. Также можно изменить стиль шрифта. Для этого передайте параметр `font` таким образом:

```
lbl = Label(window, text="Привет", font=("Arial Bold", 50))
```

Настройка размеров окна приложения

Мы можем установить размер окна по умолчанию, используя функцию `geometry` следующим образом:

```
window.geometry('400x250')
```

В приведенной выше строке устанавливается окно шириной до 400 пикселей и высотой до 250 пикселей.

Добавление виджета Button

Начнем с добавления кнопки в окно. Кнопка создается и добавляется в окно так же, как и метка:

```
btn = Button(window, text="Не нажимать!")
```

```
btn.grid(column=1, row=0)
```

Все остальные виджеты из библиотеки добавляются похожим образом.

Создание Класса

```
class Vehicle(object):  
    """docstring"""  
  
    def __init__(self):  
        """Constructor"""  
        pass
```

Этот класс не делает ничего конкретного, тем не менее, это очень хороший инструмент для изучения. Например, чтобы создать класс, мы используем ключевое слово `class`, за которым следует наименование класса. В Пайтоне, конвенция указывает на то, что наименование класса должно начинаться с заглавной буквы. Далее нам нужно открыть круглые скобки, за которыми следует слово `object` и закрытые скобки. «`object`» — то, на чем основан класс, или наследуется от него. Это называется базовым классом или родительским классом. Большая часть классов в Пайтоне основаны на объекте. У классов есть особый метод, под названием `__init__`.

Этот метод вызывается всякий раз, когда вы создаете (или создаете экземпляр) объект на основе этого класса. Метод `__init__` вызывается единожды, и не может быть вызван снова внутри программы. Обратите внимание на то, что каждый метод должен иметь как минимум один аргумент, что в случае с обычной функцией уже не вяжется. В Python 3 нам не нужно прямо указывать, что мы наследуем у объекта. Вместо этого, мы можем написать это следующим образом:

```
class Vehicle:  
    """docstring"""  
  
    def __init__(self):  
        """Constructor"""  
        pass
```

Обратите внимание на то, что единственная разница в том, что круглые скобки нам больше не нужны, когда мы основываем наш класс на объекте. Давайте немного расширим наше определение класса и дадим ему некоторые атрибуты и методы.

```
class Vehicle(object):
    """docstring"""

    def __init__(self, color, doors, tires):
        """Constructor"""
        self.color = color
        self.doors = doors
        self.tires = tires

    def brake(self):
        """
        Stop the car
        """
        return "Braking"

    def drive(self):
        """
        Drive the car
        """
        return "I'm driving!"
```

Задание 1

Описание программы

Программа написана на алгоритмическом языке Python 3.9.1, реализована в среде OS Windows 10 и состоит из частей, отвечающих за ввод данных, их преобразования к численному формату, вычисления и представления итоговых данных на экране монитора.

Описание алгоритма

1. Считывание данных из data-файлов, приведение их к необходимым структурам данных.
2. Действия в зависимости от выбора пользователя, рендер необходимых частей интерфейса, изменение данных.
3. Выгрузка измененных данным в data-файлы

Описание входных и выходных данных

Входные данные поступают с клавиатуры, а выходные – выводятся на монитор для просмотра.

Листинг программы

```
from tkinter import *
from ex00_note import Note

class Note:

    def init_from_str(string: str):
        params = string.split("&")
        date = params[3].split(".")
        return Note(params[1], params[0], params[2], date)

    def __init__(self,
                 surname: str,
                 name: str,
                 tel,
                 birth: list[int]) -> None:
        self.surname = surname.capitalize()
        self.name = name.capitalize()
        self.tel = tel
        self.birth = birth

    def __str__(self):
        return "{}&{}&{}&{}.{}.{}". \
            format(self.name, self.surname, self.tel, self.birth[0],
                  self.birth[1], self.birth[2])

    def pretty(self):
        return "{} {} Phone: {} Date Of Birth: {}.{}.{}". \
            format(self.name, self.surname, self.tel, self.birth[0],
                  self.birth[1], self.birth[2])

    def to_string(self):
        return self.__str__()

    def days_past(self):
        return int(self.birth[2]) * 365 + int(self.birth[1]) * 30 +
        int(self.birth[0])
```

```

def list_notes(fr: Frame) -> None:
    for widget in fr.wininfo_children():
        widget.destroy()
    notes.sort(key=lambda x: x.days_past())
    for note in notes:
        lb = Label(fr, text=note.pretty())
        lb.pack()

def clear_notes() -> None:
    fr = notes_frame
    notes.clear()
    for widget in fr.wininfo_children():
        widget.destroy()

def clear() -> None:
    fr = notes_frame
    list_notes(fr)

def make_file():
    fr = notes_frame
    path = fr.wininfo_children()[0].get()
    for widget in fr.wininfo_children():
        widget.destroy()
    try:
        with open(path, 'w', encoding='utf-8') as g:
            for note in notes:
                g.write(note.pretty() + '\n')
            lb = Label(fr, text="Sucesss")
            lb.pack()
    except:
        lb = Label(fr, text="Falid to write a text")
        lb.pack()

def write_file():
    fr = notes_frame
    for widget in fr.wininfo_children():
        widget.destroy()

    path = Entry(fr)
    path.insert(0, './output.txt')
    path.pack(pady=5)

    write_btn = Button(fr, text='Write', command=make_file, width=10, height=1)
    write_btn.pack(padx=3)

def find_notes():
    fr = notes_frame
    tel = fr.wininfo_children()[0].get()
    for widget in fr.wininfo_children():
        widget.destroy()
    for note in notes:
        if tel == note.tel:
            lb = Label(fr, text=note.pretty())
            lb.pack()
    if len(fr.wininfo_children()) == 0:
        lb = Label(fr, text="No notes with same phone number")
        lb.pack()

def serch():
    fr = notes_frame
    for widget in fr.wininfo_children():
        widget.destroy()

    t1 = Entry(fr)

```

```

tl.insert(0, 'Tel')
tl.pack(pady=5)

sch_btn = Button(fr, text='Search', command=find_notes, width=10, height=1)
sch_btn.pack(padx=3)

def add_note():
    fr = notes_frame
    vl = []
    for widget in fr.winfo_children():
        if widget.winfo_class() == 'Entry':
            vl.append(widget.get())
        if widget.winfo_class() == 'Frame':
            date = []
            for item in widget.winfo_children():
                date.append(item.get())
            vl.append(date)
    new_note = Note(vl[1], vl[0], vl[2], vl[3])
    notes.append(new_note)
    list_notes(notes_frame)

def add():
    fr = notes_frame
    for widget in fr.winfo_children():
        widget.destroy()

    nm = Entry(fr)
    nm.insert(0, 'Name')
    nm.pack(pady=5)

    sr = Entry(fr)
    sr.insert(0, 'Surname')
    sr.pack(pady=5)

    tl = Entry(fr)
    tl.insert(0, 'Tel')
    tl.pack(pady=5)

    bd_frame = Frame(fr, width=230, height=30)
    bd_frame.pack(pady=5)

    day = Spinbox(bd_frame, from_=1, to=31, width=5)
    day.pack(side='left', padx=3)

    month = Spinbox(bd_frame, from_=1, to=12, width=5)
    month.pack(side='left', padx=3)

    year = Spinbox(bd_frame, from_=1920, to=2021, width=5)
    year.pack(side='left', padx=3)

    add_btn = Button(fr, text='Save', command=add_note, width=10, height=1)
    add_btn.pack(padx=3)

root = Tk()
notes = []
try:
    with open('ex00_data.txt ', 'r', encoding='utf-8') as g:
        while True:
            line = g.readline()
            if not line:
                break
            new_note = Note.init_from_str(line[:-1])
            notes.append(new_note)
except:

```

```

pass

root.geometry('600x300')

canvass = Canvas(root, bg="#263D42")
canvass.place(relwidth=1, relheight=1)

notes_frame = Frame(root, bg="lightgrey")
notes_frame.place(relx=0.05, rely=0.05, relwidth=0.9, relheight=0.6)

buttons_frame = Frame(root, bg="#263D42")
buttons_frame.place(relx=0.05, rely=0.75, relwidth=0.9, relheight=0.15)

list_notes(notes_frame)

add_button = Button(buttons_frame, text='Add note', command=add)
add_button.pack(side='left', padx=5, expand=1)

clear_button = Button(buttons_frame, text='Del notes', command=clear_notes)
clear_button.pack(side='left', padx=5, expand=1)

clear_button = Button(buttons_frame, text='Search', command=serch)
clear_button.pack(side='left', padx=5, expand=1)

write_button = Button(buttons_frame, text='Write to file', command=write_file)
write_button.pack(side='left', padx=5, expand=1)

clear_button = Button(buttons_frame, text='Clear', command=clear)
clear_button.pack(side='left', padx=5, expand=1)

root.mainloop()

with open('ex00_data.txt ', 'w', encoding='utf-8') as g:
    for note in notes:
        g.write(note.to_string() + '\n')

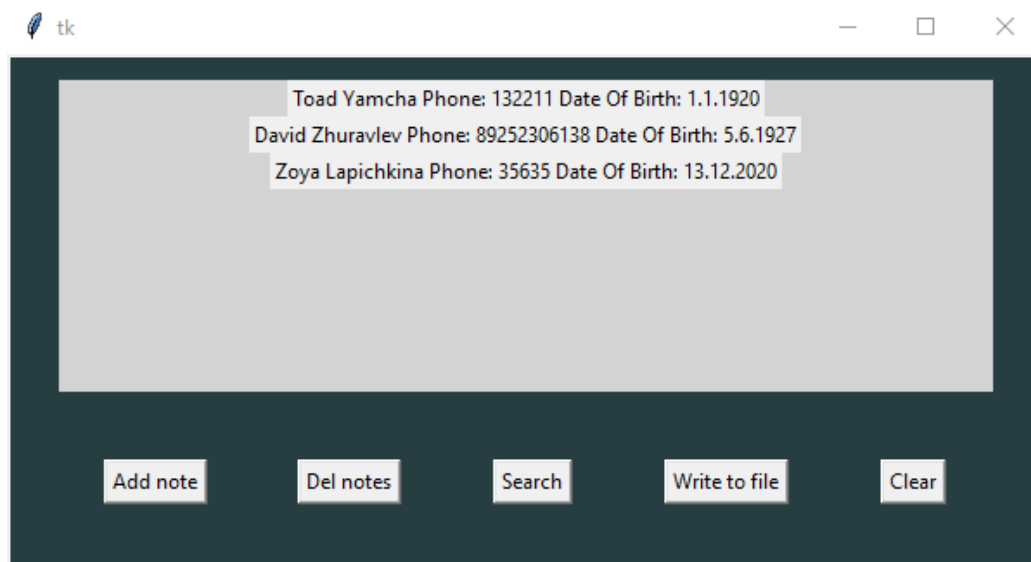
```

Результат работы программы

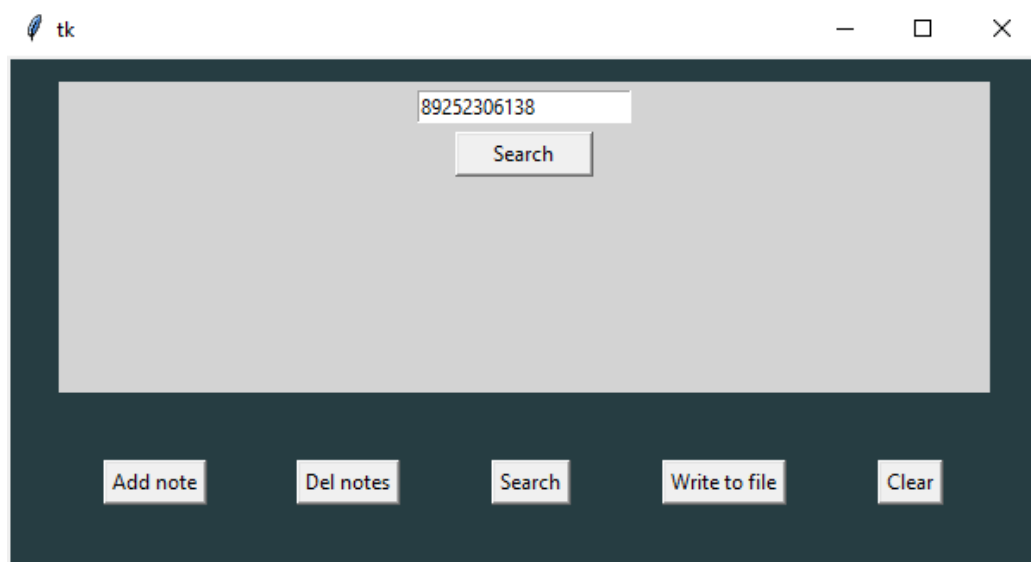
Форма ввода нового элемента Note

The screenshot shows a Tkinter window titled "tk" with a dark blue border and standard window controls (minimize, maximize, close). The main content area has a light grey background. In the center, there is a form for adding a new note. The form consists of three text input fields labeled "Name", "Surname", and "Tel". Below these fields are three spin boxes with values 1, 1, and 1920. A "Save" button is positioned below the spin boxes. At the bottom of the window, there is a dark blue bar containing five buttons: "Add note", "Del notes", "Search", "Write to file", and "Clear".

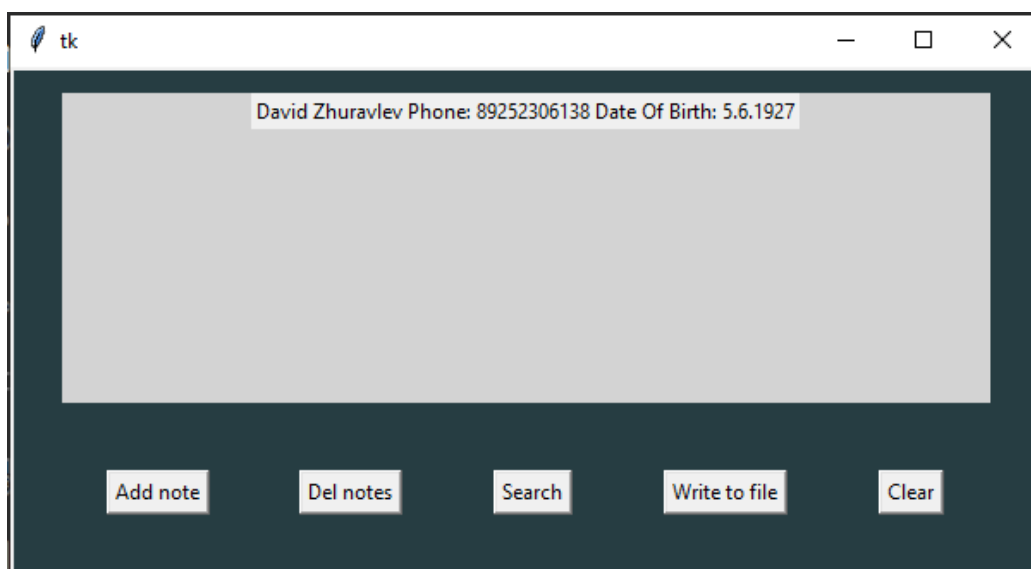
Список существующих элементов Note

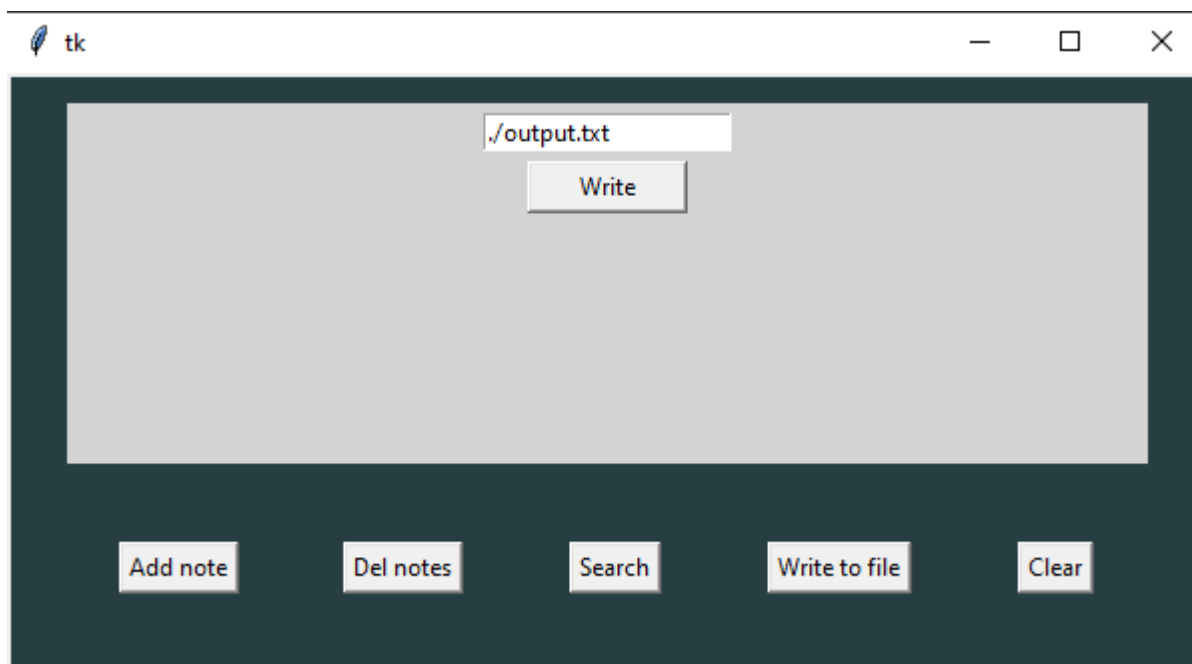


Окно поиска записи по номеру телефона



Результат поиска





Задание 2

Описание программы

Программа написана на алгоритмическом языке Python 3.9.1, реализована в среде OS Windows 10 и состоит из частей, отвечающих за ввод данных, их преобразования к численному формату, вычисления и представления итоговых данных на экране монитора.

Описание алгоритма

1. Считывание данных из data-файлов, приведение их к необходимым структурам данных.
2. Действия в зависимости от выбора пользователя, рендер необходимых частей интерфейса, изменение данных.
3. Выгрузка измененных данным в data-файлы

Описание входных и выходных данных

Входные данные поступают с клавиатуры, а выходные – выводятся на монитор для просмотра.

Листинг программы

```
from tkinter import *
from tkinter import scrolledtext
from ex00_note import Note

def list_text(fr: Frame) -> None:
    for widget in fr.winfo_children():
        widget.destroy()
    txt = scrolledtext.ScrolledText(fr)
    txt.insert(INSERT, " ".join(text))
    txt.place( relwidth=1, relheight=0.9)
    save_button = Button(fr, text='Save text', command=lambda: save_text(fr,
text))
    save_button.place(relx=0.4, rely=0.9, relwidth=0.2, relheight=0.1)
```

```

def save_text(fr: Frame, text) -> None:
    temp = fr.winfo_children()[0]
    res = temp.winfo_children()[1].get(1.0, END).split(' ')
    text.clear()
    for word in res:
        text.append(word)

def add():
    pass

def add_descr(fr: Frame) -> None:
    for widget in fr.winfo_children():
        widget.destroy()

    word = Entry(fr)
    word.insert(0, 'Слово')
    word.place(relx=0.1, rely=0.1, relwidth=0.5, relheight=0.1)

    txt = scrolledtext.ScrolledText(fr)
    txt.insert(INSERT, "Описание")
    txt.place(relx=0.1, rely=0.3, relwidth=0.8, relheight=0.5)

    save_button = Button(fr, text='Save description', command=lambda:
save_descr(fr))
    save_button.place(relx=0.3, rely=0.9, relwidth=0.4, relheight=0.1)

def save_descr(fr):
    word = fr.winfo_children()[0].get()
    temp = fr.winfo_children()[1]
    d = temp.winfo_children()[1].get(1.0, END)

    for widget in fr.winfo_children():
        widget.destroy()
    descr[word] = d
    list_text(fr)

def search_word(fr: Frame) -> None:
    for widget in fr.winfo_children():
        widget.destroy()

    word = Entry(fr)
    word.insert(0, 'Слово')
    word.place(relx=0.1, rely=0.1, relwidth=0.5, relheight=0.1)

    search_button = Button(fr, text='Search', command=lambda: search(fr))
    search_button.place(relx=0.1, rely=0.3, relwidth=0.4, relheight=0.1)

def search(fr: Frame) -> None:
    word = fr.winfo_children()[0].get()

    for widget in fr.winfo_children():
        widget.destroy()

    if word in descr:
        txt = scrolledtext.ScrolledText(fr)
        txt.insert(INSERT, word + '\n\n')
        txt.insert(END, descr[word])
        txt.place(relx=0, rely=0, relwidth=1, relheight=1)

def list_all_words(fr: Frame) -> None:
    for widget in fr.winfo_children():
        widget.destroy()

    txt = scrolledtext.ScrolledText(fr)

```

```

        for word in descr.keys():
            txt.insert(END, word + '\n\n')
            txt.insert(END, descr[word])
            txt.insert(END, "-----\n")
        txt.place(relx=0, rely=0, relwidth=1, relheight=1)

root = Tk()
text = []
descr = {}

try:
    with open('ex01_text_data.txt', 'r', encoding='utf-8') as g:
        while True:
            line = g.readline()
            if not line:
                break
            words = line.split(' ')
            for word in words:
                text.append(word)
except:
    pass

try:
    with open('ex01_descr_data.txt', 'r', encoding='utf-8') as g:
        while True:
            line = g.readline()
            if not line:
                break
            words = line.split('|')
            descr[words[0]] = words[1].replace('###n', '\n')
except:
    pass

root.geometry('600x600')

c canvass = Canvas(root, bg="#263D42")
c canvass.place(relwidth=1, relheight=1)

text_frame = Frame(root, bg="lightgrey")
text_frame .place(relx=0.05, rely=0.05, relwidth=0.9, relheight=0.8)

buttons_frame = Frame(root, bg="#263D42")
buttons_frame.place(relx=0.05, rely=0.85, relwidth=0.9, relheight=0.1)

list_text(text_frame)

open_button = Button(buttons_frame, text='Open manual', command=lambda:
list_text(text_frame))
open_button.pack(side='left', padx=5, expand=1)

search_button = Button(buttons_frame, text='Search word', command=lambda:
search_word(text_frame))
search_button.pack(side='left', padx=5, expand=1)

descr_button = Button(buttons_frame, text='Add description', command=lambda:
add_descr(text_frame))
descr_button.pack(side='left', padx=5, expand=1)

all_button = Button(buttons_frame, text='All words', command=lambda:
list_all_words(text_frame))
all_button.pack(side='left', padx=5, expand=1)

root.mainloop()

```

```

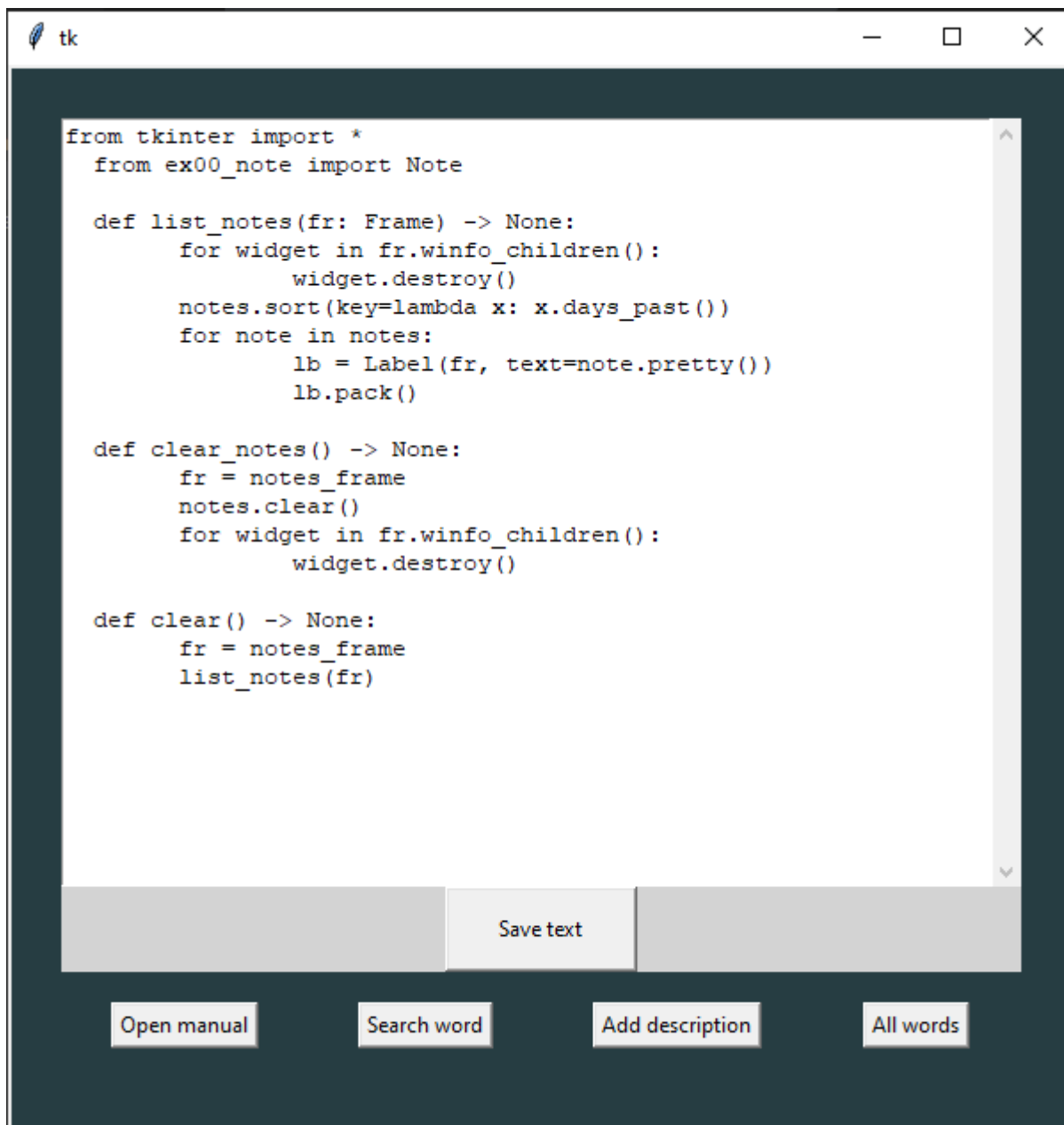
with open('ex01_text_data.txt', 'w', encoding='utf-8') as g:
    g.write(' '.join(text).replace(' ', ''))

with open('ex01_descr_data.txt', 'w', encoding='utf-8') as g:
    for key in descr.keys():
        g.write("{}|{}".format(key, descr[key].replace('\n', '###n') + '\n'))

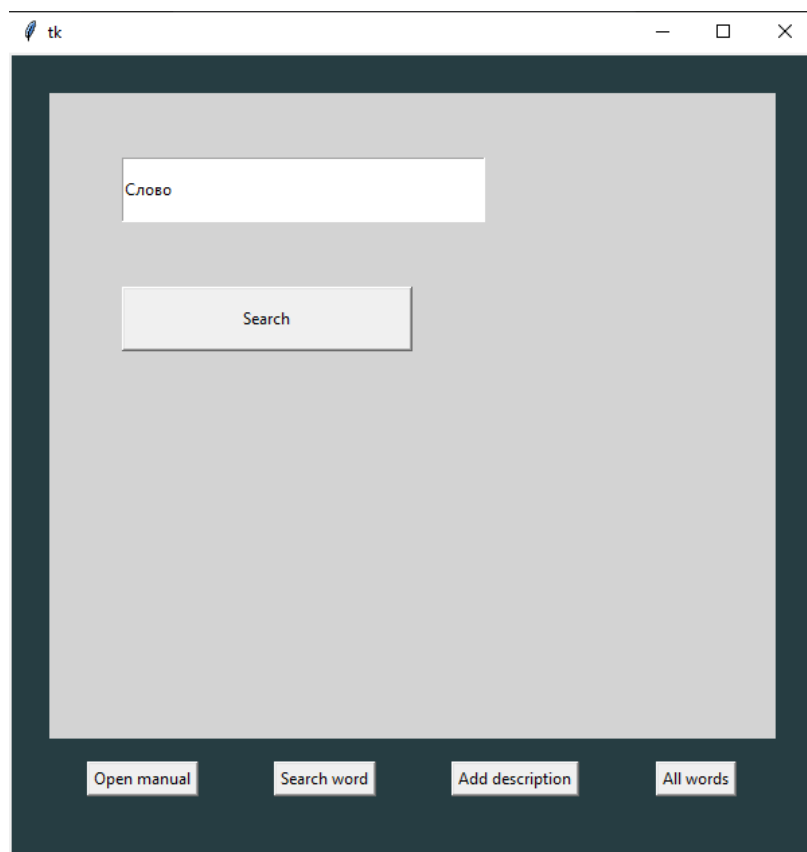
```

Результат работы программы

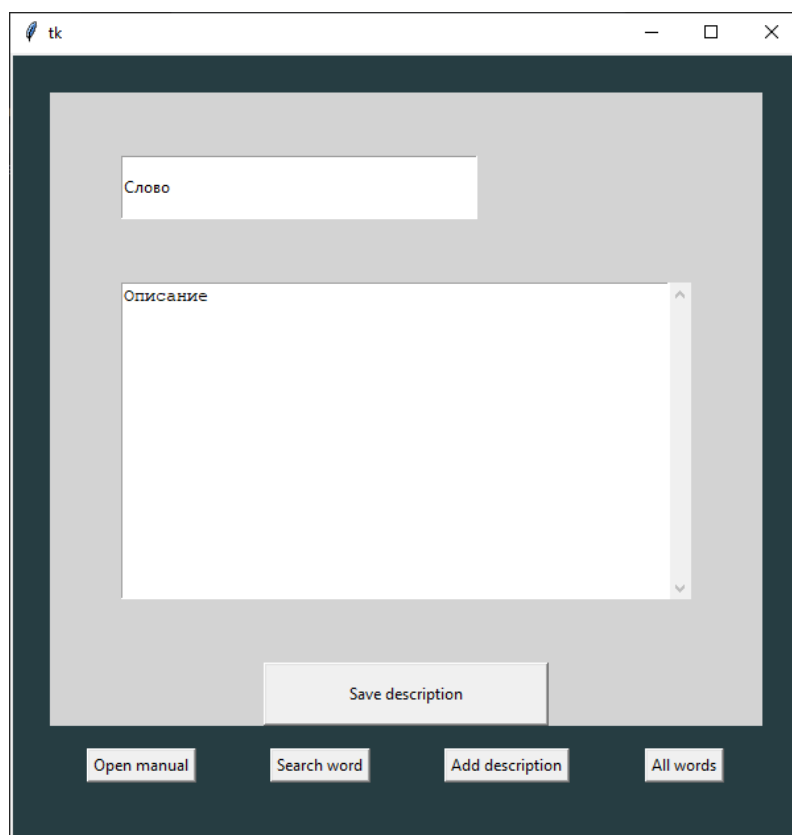
Вывод основного текста. В этом же окне его можно изменить и сохранить изменения.



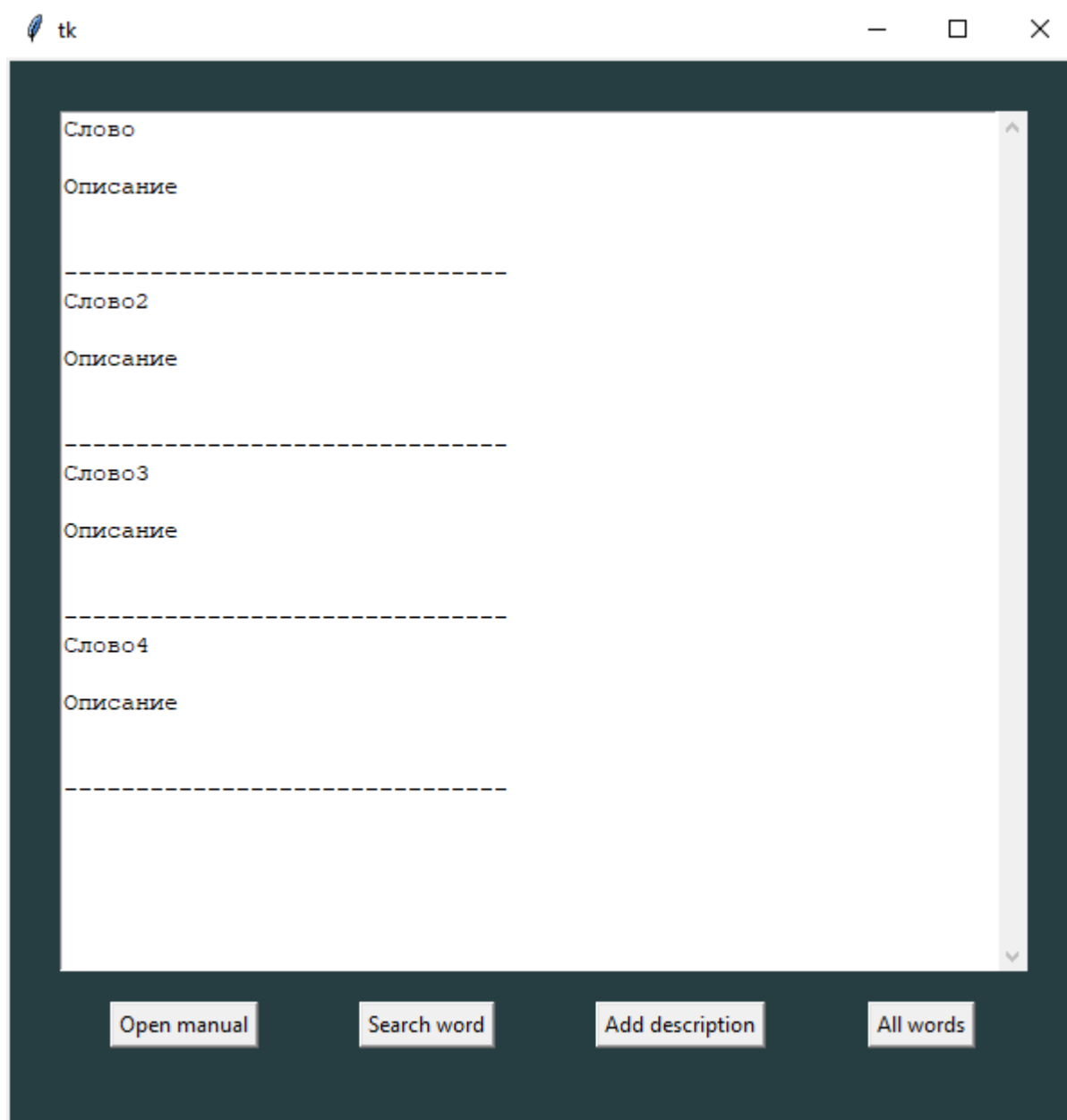
Окно поиска слова в списке слов с пояснениями



Добавление нового описания



Все доступные описания



tk

Слово

Описание

Слово2

Описание

Слово3

Описание

Слово4

Описание

Open manual Search word Add description All words

Список используемой литературы

1. В.П. Рядченко, Методическое пособие по выполнению лабораторных работ
2. <https://pythonworld.ru/>