import sys

import sqlite3

def print\_menu():

print ('\nPlease select from the following items:')

print('1. Add new contact')

print('2. Display contacts')

print('3. Edit contacts')

print('4. Delete contacts')

print('5. Find contacts')

print('0. Exit the program')

def addcontact():

while True:

name = input("What is the person's first name?: ")

if len(name) != 0:

break

else:

print("Please enter the contact's name")

while True:

surname = input("What is the person's surname name?: ")

if len(surname) != 0:

break

else:

print("Please enter the surname")

while True:

num = input("What is the person's phone number? (only digits allowed): ")

if not num.isdigit():

print("Please enter only digits")

continue

elif len(num) != 10:

print("Please enter 10-digits phone number with no comma, no spaces, no punctuation")

continue

else:

break

cursor.execute('''INSERT INTO phonebook (name, surname, phone\_number) VALUES (?,?,?)''',

(name, surname, num))

conn.commit()

print("New contact " + surname + ' ' + name + " was added to the phonebook table")

def displaybook():

cursor.execute("SELECT surname, name, phone\_number FROM phonebook ORDER BY surname")

results = cursor.fetchall()

print(results)

def key\_pair\_reception(str):

print ("\nPlease select the key field for " + str + " (from 1 to 3)")

print('1. Name')

print('2. Surname')

print('3. Phone\_number')

print('0. Return to the Main menu')

n = int(input('Your choice: '))

if n == 1:

field = "name"

elif n == 2:

field = "surname"

elif n == 3:

field = "phone\_number"

else:

return None

keyword = input("\nPlease enter the key value: " + field + " = ")

keypair = field + "='" + keyword + "'"

return keypair

def editcontacts():

s = key\_pair\_reception('searching')

u = key\_pair\_reception('updating')

if s != None:

sql = "UPDATE phonebook SET " + u + " WHERE " + s

cursor.execute(sql)

conn.commit()

print("The records with " + s + " are deleted.\n")

def deletecontacts():

s = key\_pair\_reception('searching')

if s != None:

sql = 'DELETE FROM phonebook WHERE ' + s

cursor.execute(sql)

conn.commit()

print("The records with " + s + " are deleted.\n")

def findcontacts():

s = key\_pair\_reception('searching')

if s != None:

sql = 'SELECT surname, name, phone\_number FROM phonebook WHERE ' + s + ' ORDER BY surname'

cursor.execute(sql)

results = cursor.fetchall()

print(results)

# Основная программа

print ('\nWELCOME TO THE PHONE DIRECTORY')

conn = sqlite3.connect('my.db')

cursor = conn.cursor()

cursor.execute('''CREATE TABLE IF NOT EXISTS phonebook (

id integer PRIMARY KEY,

name text NOT NULL,

surname text,

phone\_number text)''');

m = -1

while m != 0:

print\_menu()

m = int(input('Your choice: '))

if m == 1:

addcontact()

continue

elif m == 2:

displaybook()

continue

elif m == 3:

editcontacts()

continue

elif m == 4:

deletecontacts()

continue

elif m == 5:

findcontacts()

continue

elif m == 0:

print('The program is completed.\n')

conn.close()

sys.exit(0)

else:

print('Please follow instructions')

Домашнее задание

def read\_data():

pass

def calculate(data):

pass

def print\_data():

pass

def my\_func():

data = read\_data()

answer = calculate(data)

print\_data(answer)

def cool\_func():

pass

if \_\_name\_\_ == '\_\_main\_\_':

my\_func()

кодирование

*# файл \*.pyw (расширение .pyw необходимо для подавления окна DOS)*

**from** tkinter **import** \*                              *# импорт объектов для графики*

**import** shelve                                      *# импорт модуля для работы с базой данных*

**class** PhoneBook:

*# класс Телефонная книга*

**def** \_\_init\_\_(self, nameBook, dicRec={}):

*# метод инициализации атрибутов экземпляров класса  и передача в метод значений для атрибутов*

        self.nameBook = nameBook                   *# инициализация атрибута Имя книги*

        self.dicRec = dicRec                       *# перечень записей в виде словаря, по умолчанию пустой*

**def** loadBook(self):

*# загрузка записей из базы данных*

        db = shelve.open(self.nameBook)            *# открытие базы данных (имя берется из атрибута nameBook класса)*

        self.dicRec = dict(db.items())             *# загрузка записей из базы данных в атрибут dicRec класса (в словарь класса)*

        db.close()                                 *# закрытие базы данных*

**def** saveBook(self):

*# сохранение записей в базе данных*

        db = shelve.open(self.nameBook)            *# открытие базы данных*

**for** (key, record) **in** self.dicRec.items():  *# запись содержимого из*

            db[key] = record                       *# словаря класса в базу данных*

        db.close()                                 *# закрытие базы данных*

**class** PhoneRec:

*# класс Запись*

**def** \_\_init\_\_(self, keyRec, char, label, phone, familyName, comment, delR=''):

*# инициализация атрибутов экземпляров класса*

        self.keyRec = keyRec                       *# ключ записи*

        self.char = char                           *# буква, на странице которой находится запись*

        self.label = label                         *# метка, к чему относится запись*

        self.phone = phone                         *# телефон*

        self.familyName = familyName               *# Ф.И.О.*

        self.comment = comment                     *# комментарий*

        self.delR = delR                           *# служебное поле для пометки записи:*

*# ''- видимая, 'с' - скрытая, 'у' - удаляемая*

fieldnamesRec = ('keyRec', 'char', 'label', 'phone', 'familyName', 'comment', 'delR')  *# кортеж имен полей в записи*

activCh = 'А'                                      *# буква, актмвная на текущий момент*

typeRec = ''                                       *# тип выводимых на экран записей, '' - открытые, "с" - скрытые*

dicRem = {}                                        *# словарь оставшихся не выведенными записей*

**def** makeWidgets():

*# создание графической формы*

**global** entriesRec, entRec, lab1                *# перечень глобальных переменных, которые будут использоваться и за пределами функции*

    entRec = {}                                    *# словарь, в который будут заносится объекты ввода entFind (поиск) и entKeyRec (ключ)*

    window = Tk()                                  *# создание главного окна*

    window.title('Телефоны')                       *# заголовок окна*

    window.geometry('1260x600+0+0')                *# размеры окна*

    form1 = Frame(window)                          *# создание внутри окна window контейнера form1*

    form1.pack()

    lab1 = Label(form1, text=activCh, fg="#eee", bg="#333", width=5)  *# метка, показывающая*

    lab1.pack(side=LEFT)                                              *# активную букву*

    Label(form1, text='  ', width=30).pack(side=LEFT)                 *# вспомагательная пустая метка для улучшения расположения*

    alph = ["А", "Б", "В", "Г", "Д", "Е", "Ж", "З", "И", "К", "Л", "М", "Н", "О", "П", "Р", "С", "Т",

            "У", "Ф", "Х", "Ц", "Ч", "Ш", "Щ", "Э", "Ю", "Я"]         *# список букв*

**for** i **in** range(len(alph)):                     *# создание кнопок с буквами*

        Button(form1, text=alph[i], command=(**lambda** x=alph[i]: fetchChr(x))).pack(side=LEFT)

    ent = Entry(form1, width=27)                   *# поле ввода для поиска*

    ent.pack(side=LEFT)

    entRec['entFind'] = ent                        *# поместить объект поля ввода в словарь entRec*

    Button(form1, text="Поиск", command=fetchFind).pack(side=LEFT)     *# создание кнопки Поиск*

    form2 = Frame(window)                          *# создание внутри окна window контейнера form2*

    form2.pack()

    entriesRec = {}                *# словарь, для занесения в него объектов Entry ячеек таблицы ввода записей*

**for** (ix, label) **in** enumerate(fieldnamesRec): *# создание надписей заголовков столбцов таблицы*

        lab = Label(form2, text=label)

        lab.grid(row=2, column=ix)

**for** i **in** range(1, 26):         *# создаются ячейки таблицы для ввода записей*

**for** (ix, label) **in** enumerate(fieldnamesRec):

**if** label == 'keyRec' **or** label == 'char' **or** label == 'delR':  *# выделяются столбцы, которые потом будут иметь особый режим доступа*

                ent = Entry(form2, state='normal', width=6)

**else**:

                ent = Entry(form2, width=40)

            ent.grid(row=i+2, column=ix)

            entriesRec[label+str(i)] = ent   *# объекты ячеек таблицы заносятся в словарь, причем к имени*

*# столбца ячейки добавляется номер строки,*

*# тем самым однозначно определяются координаты ячейки,*

*# что бы к ней обращаться*

    form3 = Frame(window)                            *# создание внутри окна window контейнера form3*

    form3.pack()

    Button(window, text="Следующая страница", command=fetchNext).pack()  *# кнопка Следующая страница*

    Label(window, text='      ', width=10).pack(side=LEFT)       *# вспомогательная пустая метка*

    labKeyRec = Label(window, text='keyRec').pack(side=LEFT)     *# надпись перед полем ввода номера ключа*

    ent = Entry(window, width=10)                                *# поле ввода номера ключа*

    ent.pack(side=LEFT)

    entRec['entKeyRec'] = ent             *# занесение объекта поле ввода номера ключа в словарь entRec*

    Button(window, text="Скрыть", command=hideRec).pack(side=LEFT)             *# кнопка Скрыть (запись)*

    Button(window, text="Показать скрытые", command=fetchHide).pack(side=LEFT) *# кнопка Показать скрытые*

    Button(window, text="Открыть", command=openRec).pack(side=LEFT)            *# кнопка Открыть (запись)*

    Label(window, text=' ', width=5).pack(side=LEFT)                           *# вспомогательная пустая метка*

    Button(window, text="Удалить", command=delKeyRec).pack(side=LEFT)          *# кнопка удалить (запись)*

    Label(window, text='      ', width=30).pack(side=LEFT)                     *# вспомогательная пустая метка*

    btns = Button(window, text="Сохранить", command=interSave).pack(side=LEFT)   *# кнопка Сохранить (страницу)*

    Label(window, text='      ', width=20).pack(side=LEFT)                     *# вспомогательная пустая метка*

    Button(window, text="Выход", command=fin).pack(side=LEFT)                  *# кнопка Выход (из программы)*

**return** window                                              *# функция makeWidgets возвращает окно window*

**def** clear\_sheet():

**for** i **in** range(1, 26):

**for** field **in** fieldnamesRec:

**if** field == 'keyRec' **or** field == 'delR':

                entriesRec[field+str(i)].config(state='normal')

                entriesRec[field+str(i)].delete(0, END)

                entriesRec[field+str(i)].config(state='readonly')

**else**:

                entriesRec[field+str(i)].delete(0, END)

**def** fetchChr(ch):

**global** activCh, typeRec, lab1

    saveRec()

    typeRec = ''

    activCh = ch

    lab1.config(text=activCh)

    dicRecChr = {}

**for** key **in** t1.dicRec.keys():

**if** t1.dicRec[key].char == ch:

            dicRecChr[key] = t1.dicRec[key]

    fetch(dicRecChr)

**def** interSave():

    fetchChr(activCh)

**def** fetchHide():

**global** typeRec, lab1

    saveRec()

    lab1.config(text='скр')

    typeRec = 'с'

    fetch(t1.dicRec)

**def** fetch(dicR):

**global** dicRem

    clear\_sheet()

    count = 1

    dicRe = dicR.copy()

**while** count <= 25 **and** len(dicRe):

**for** key **in** dicR.keys():

**if** dicR[key].delR == typeRec:

                record = dicR[key]

**for** field **in** fieldnamesRec:

**if** field == 'keyRec' **or** field == 'delR':

                        entriesRec[field+str(count)].config(state='normal')

                        entriesRec[field+str(count)].insert(0, getattr(record, field))

                        entriesRec[field+str(count)].config(state='readonly')

**else**:

                        entriesRec[field+str(count)].insert(0, getattr(record, field))

                count += 1

                dicRe.pop(key)

**if** count > 25:

**break**

**else**:

                dicRe.pop(key)

    dicRem = dicRe.copy()

**def** fetchNext():

    saveRec()

    fetch(dicRem)

**def** delKeyRec():

*# физическое удаление из базы данных*

    key = entRec['entKeyRec'].get()

**del** t1.dicRec[key]

    db = shelve.open(t1.nameBook)

**del** db[key]

    db.close()

**for** i **in** range(1, 26):

**if** entriesRec['keyRec'+str(i)].get() == key:

            entriesRec['delR'+str(i)].config(state='normal')

            entriesRec['delR'+str(i)].insert(0, 'у')

            entriesRec['delR'+str(i)].config(state='readonly')

    entRec['entKeyRec'].delete(0, END)

**def** hideRec():

*# пометить как скрытую*

    key = entRec['entKeyRec'].get()

**for** i **in** range(1, 26):

**if** entriesRec['keyRec'+str(i)].get() == key:

            entriesRec['delR'+str(i)].config(state='normal')

            entriesRec['delR'+str(i)].insert(0, 'с')

            entriesRec['delR'+str(i)].config(state='readonly')

    entRec['entKeyRec'].delete(0, END)

**def** openRec():

    key = entRec['entKeyRec'].get()

**for** i **in** range(1, 26):

**if** entriesRec['keyRec'+str(i)].get() == key:

            entriesRec['delR'+str(i)].config(state='normal')

            entriesRec['delR'+str(i)].delete(0, END)

            entriesRec['delR'+str(i)].insert(0, '')

            entriesRec['delR'+str(i)].config(state='readonly')

    entRec['entKeyRec'].delete(0, END)

**def** fetchFind():

**global** lab1

    saveRec()

    clear\_sheet()

    lab1.config(text='поиск')

    strF = entRec['entFind'].get()

    dicFind = {}

**for** key **in** t1.dicRec.keys():

        record = t1.dicRec[key]

**for** field **in** fieldnamesRec:

**if** (field != 'keyRec' **and** field != 'char' **and** field != 'delR' **and**

                getattr(record, field).find(strF) != -1):

                dicFind[key] = record

**break**

    fetch(dicFind)

**def** saveRec():

**global** typeRec

**for** i **in** range(1, 26):

        key = entriesRec['keyRec'+str(i)].get()

**if** entriesRec['delR'+str(i)].get() == 'у':

**continue**

**elif** key:

            record = t1.dicRec[key]

**for** field **in** fieldnamesRec:

                setattr(record, field, entriesRec[field+str(i)].get())

            t1.dicRec[key] = record

**else**:

            existRec = False

**for** field **in** fieldnamesRec:

**if** entriesRec[field+str(i)].get(): existRec = True *# Если существует запись в поле на этой строке*

**if** existRec:

**if** entriesRec['char'+str(i)].get():

                    char = entriesRec['char'+str(i)].get()

**else**:

                    char = activCh

                label = entriesRec['label'+str(i)].get()

                phone = entriesRec['phone'+str(i)].get()

                familyName = entriesRec['familyName'+str(i)].get()

                comment = entriesRec['comment'+str(i)].get()

**if** len(t1.dicRec)>0:

                    L = sorted(t1.dicRec.items(), key=**lambda** item: int(item[0]))

                    keyRec = str(int(L[-1][0]) + 1)

**else**:

                    keyRec = "1"

                record = PhoneRec(keyRec, char, label, phone, familyName, comment)

                t1.dicRec[keyRec] = record

    t1.saveBook()

**def** fin():

    saveRec()

    window.destroy()

**if** \_\_name\_\_ == '\_\_main\_\_':

    t1 = PhoneBook("Телефоны")

    t1.loadBook()

    window = makeWidgets()

    fetchChr('А')

    window.mainloop()