**Урок No11. Функции**

1.

# Задание No1

# ● Создайте функцию, которая принимает в качестве параметра -

# натуральное целое число.

# ● Данная функция находит факториал полученного числа

# Например, факториал числа 3 это число 6.

# ● Теперь создайте список факториалов чисел от получившегося ранее

# факториала 6, до 1.

# В итоге, на вход программа получает например число 3, возвращает число 6

# (факториал числа 3) и вам нужно сделать список из факториалов числа 6 в

# убывающем порядке. Находим факториал числа 6 - это 720, затем от числа 5 -

# это 120 и так далее вплоть до 1

# То есть, результирующий список будет выглядеть в нашем примере так:

# [720, 120, 24, 6, 2, 1]

#

# https://github.com/A-l-E-v/PySynergy/blob/main/U-11/factorial.py

#

print()

print('--- Факториалы ---')

print()

# считаем факториал числа рекурсией

def fac(i):

if i == 1:

return 1

return fac(i - 1) \* i

number = int (input('От какого числа посчитаем ряд факториалов? '))

# считаем факториал и он будет началом ряда

start = fac(number)

print(f'Факториал: {number}!={start}')

# инициируем список факториалов

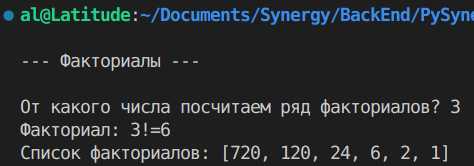
fac\_list = []

# запускаем цикл от старта до 0 с шагом -1

for series in range(start,0,-1):

fac\_list.append (fac(series))

print ('Список факториалов:', fac\_list)

Проверю работу программы от числа 3, как в задании:

2.

# Задание No2

# В Урок No10. Задание No1 вы создавали словарь с информацией о питомце.

# Теперь нам нужна 'настоящая' база данных для ветеринарной клиники.

# Подробный требуемый функционал будет ниже. Пока что справка:

#

# Создайте функцию create

# Создайте функцию read

# Создайте функцию update

# Создайте функцию delete

# Используйте словарь pets, который будет предоставлен ниже, либо

# создайте свой аналогичный

# Функция create:

# Данная функция будет создавать новую запись с информацией о питомце и

# добавлять эту информацию в наш словарь pets

# Функция read

# Данная функция будет отображать информацию о запрашиваемом питомце в

# виде:

# Это желторотый питон по кличке 'Каа'. Возраст питомца: 19 лет. Имя владельца:

# Саша

# Функция update

# Данная функция будет обновлять информацию об указанном питомцеФункция delete

# Данная функция будет удалять запись о существующем питомце

# Структура результирующего словаря pets будет как и в Урок No10. Задание No1,

# но с небольшим видоизменением:

# Словарь pets

# pets = {

# 1:

# {

# 'Мухтар': {

# 'Вид питомца': 'Собака',

# 'Возраст питомца': 9,

# 'Имя владельца': 'Павел'

# },

# },

# 2:

# {

# 'Каа': {

# 'Вид питомца': 'желторотый питон',

# 'Возраст питомца': 19,

# 'Имя владельца': 'Саша'

# },

# },

# # и так далее

# }

# Здесь, 1 и 2 - это идентификаторы наших питомцев. Это уникальные ключи, по

# которым мы сможем обращаться к нашим записям в 'базе данных'

# Суть будущей программы будет заключаться в следующем:

# ● Программа будет работать с помощью цикла while с условием command

# != 'stop', то есть до тех пор, пока на предложение ввести команду,

# пользователь не введёт слово stop● Перед взаимодействием с 'базой данных' запрашивается одна из

# команд в качестве пользовательского ввода. Пусть это будет

# переменная command

# ● Функция create должна добавлять новую информацию таким образом,

# чтобы идентификатор увеличивался на единицу. Чтобы у вас была

# возможность получать последний ключ в словаре воспользуйтесь

# импортом модуля collections. В начале вашей программы пропишите

# строчку: import collection, а в функции create в первых строках пропишите

# следующий код:

# def create():

# last = collections.deque(pets, maxlen=1)[0]

# last в данном случае и будет число последнего ключа (или в нашей

# логике - идентификатора записи). Именно его и необходимо будет

# увеличивать на единицу при добавлении следующей записи.

# Как вам уже известно - суть функций заключается в том, чтобы использовать

# один и тот же код в нескольких местах. В данной задаче вам предстоит

# получать информацию о питомце несколько раз. Чтобы не повторяться в коде,

# вам нужно создать такие функции

# get\_pet(ID):

# def get\_pet(ID):

# # функция, с помощью которой вы получите информацию о питомце в виде

# словаря

# # сделайте проверку, если питомца с таким ID нету в нашей 'базе данных'

# # верните в этом случае False

# # а если питомец всё же есть в 'базе данных' - верните информацию о нём

# # выглядеть это может примерно так:

# return pets[ID] if ID in pets.keys() else False

# get\_suffix(age):

# def get\_suffix(age):

# # функция, с помощью которой можно получить суффикс

# # 'год', 'года', 'лет'

# # реализацию этой функции вам предстоит придумать самостоятельно

# # функция будет возвращать соответствующую строку

# return pets\_list():

# def pets\_list():

# # Эта функция будет создана для удобства отображения всего списка питомцев

# # Информацию по каждому питомцу можно вывести с помощью цикла for

# Обратите внимание, если ID не существует в словаре с питомцами - будет

# возникать ошибка. Вам можно от неё избавиться, если правильно составить

# проверочное условие. Здесь не потребуется использовать такие конструкции,

# как try, except, чтобы обработать возникшую ошибку

#

# https://github.com/A-l-E-v/PySynergy/blob/main/U-11/petsDB.py

#

import collections

from collections import deque #для проверки пустого словаря

# дефолтный словарь питомцев

pets = {

1:

{

'Мухтар': {

'Вид питомца': 'собака',

'Возраст питомца': 9,

'Имя владельца': 'Павел'

},

},

2:

{

'Каа': {

'Вид питомца': 'желторотый питон',

'Возраст питомца': 19,

'Имя владельца': 'Саша'

},

},

}

def list():

# узнаём размер словаря

pets\_length = len(pets)

# проходимся по каждому если есть, если нет, то переходим на ввод первого

if (pets\_length):

for pet\_id in pets:

pet = get\_pet(pet\_id)

if pet:

pet\_name = [k for k in pet.keys()][0]

age = pet[pet\_name]['Возраст питомца']

print (f'Это {pet[pet\_name]['Вид питомца']} по кличке "{pet\_name}". \

Возраст питомца: {age} {get\_suffix(age)}. \

Имя владельца: {pet[pet\_name]['Имя владельца']}')

else:

print ('БД пуста! Создайте первую запись!')

return

# функция постановки год(а)/лет

def get\_suffix(age):

if age % 10 == 1 and age != 11 and age % 100 != 11: return 'год'

elif 1 < age % 10 <= 4 and age != 12 and age != 13 and age != 14: return 'года'

else: return 'лет'

# функция получения информации по ID питомца

def get\_pet(ID):

return pets[ID] if ID in pets.keys() else False

# функция создания записи

def create():

# по умолчанию увеличиваем ID на единицу от последнего

last\_pet = last()

last\_pet += 1

pet\_name = input ('Введите имя питомца: ')

# словарь одного питомца

a\_pet = {pet\_name: dict()}

a\_pet [pet\_name]['Вид питомца'] = input ('Введите вид питомца: ')

a\_pet [pet\_name]['Возраст питомца'] = int(input ('Введите возраст питомца: '))

a\_pet [pet\_name]['Имя владельца'] = input ('Введите имя владельца: ')

# добавляем в словарь нового питомца

pets[last\_pet]=a\_pet

print ('Питомец добавлен!')

# функция чтения записи по ID

def read():

info()

pet\_id=int(input('Введите ID записи: '))

pet = get\_pet(pet\_id)

print()

if (pet):

pet\_name = [k for k in pet.keys()][0]

age = pet[pet\_name]['Возраст питомца']

print (f'Это {pet[pet\_name]['Вид питомца']} по кличке "{pet\_name}". \

Возраст питомца: {age} {get\_suffix(age)}. \

Имя владельца: {pet[pet\_name]['Имя владельца']}')

else:

print('Питомца с таким ID нет в БД!')

info()

# функция обновления записи по ID

def update():

info()

id = int(input('Введите ID питомца для обновления: '))

if (get\_pet(id)):

pet\_name = input ('Введите имя питомца: ')

# словарь одного питомца

a\_pet = {pet\_name: dict()}

a\_pet [pet\_name]['Вид питомца'] = input ('Введите вид питомца: ')

a\_pet [pet\_name]['Возраст питомца'] = int(input ('Введите возраст питомца: '))

a\_pet [pet\_name]['Имя владельца'] = input ('Введите имя владельца: ')

# добавляем в словарь нового питомца

pets[id]=a\_pet

print (f'Питомец {id} обновлён!')

else:

print ('Такой ID в БД не используется')

# функция удаления записи по ID

def delete():

info()

pets\_length = len(pets)

# если есть записи в словаре

if (pets\_length):

id\_allowed = True

while (id\_allowed):

id = int (input('Введите ID для удаления: '))

# if id <= last\_pet:id\_allowed=False

if (get\_pet(id)):id\_allowed=False

else:

print ('Записи с таким ID не существует.')

# print ('ДО:',pets)

del pets[id]

print()

# print ('ПОСЛЕ:',pets)

print (f'Запись под номером {id} была удалена.')

info()

print()

# иначе создаём запись

else: create()

# функция информации о БД

def info():

# запрашиваем длину словаря

pets\_length = len(pets)

if (pets\_length):

print (f'Количество записей в БД: {pets\_length}')

print ('Использующиеся ID: ', end='')

for k in pets.keys():

print (f'{k} ', end='')

print()

else:

print ('БД пуста! Создайте первую запись!')

# функция получения последней записи в словарь или false, если пусто

def last():

if bool(deque(pets)):

records = collections.deque(pets, maxlen=1)[0]

return records

else:

return False

# ------ main () --------

print()

print('--- База данных питомцев ---')

print()

# инициализируем пустую переменную команд

command = ''

# диспетчер команд

while command != 'stop':

print()

print ('Введите одну из команд (create, read, update, delete, info или list)')

command = input('Или stop для выхода: ')

if command == 'create' : create()

elif command == 'read' : read()

elif command == 'update' : update()

elif command == 'delete' : delete()

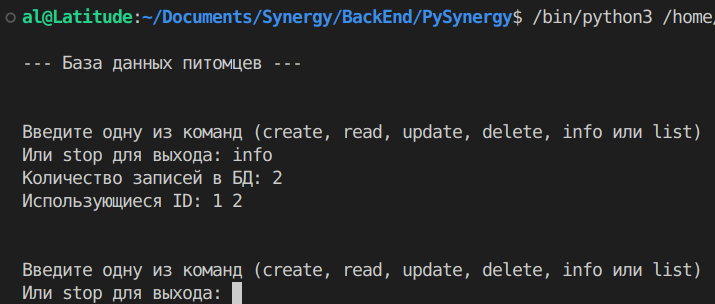
elif command == 'info' : info()

elif command == 'list' : list()

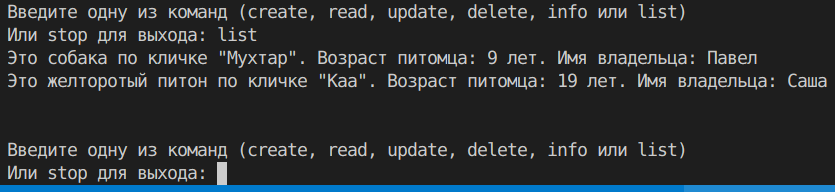
elif command != 'stop': print('Нет такой команды!')

print()

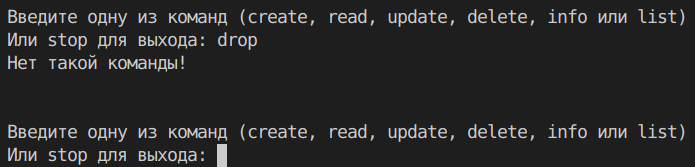
Выведу информацию по БД, введя команду info:



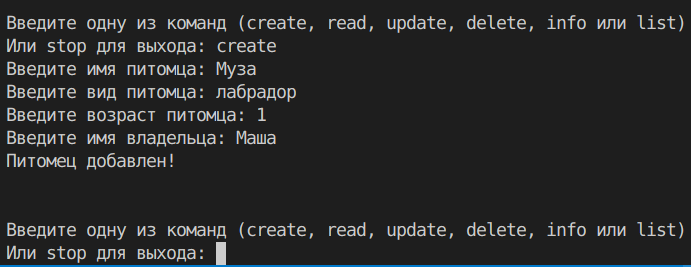
Выведу имеющиеся при записи данные, введя команду list:

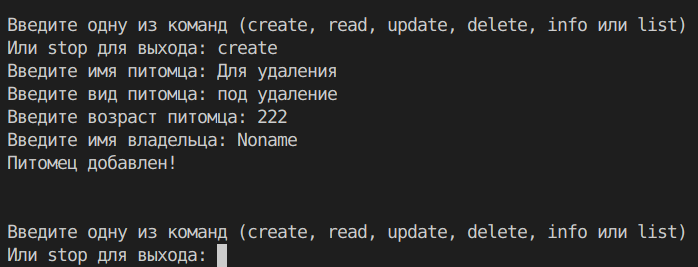


Введу не существующую команду drop:

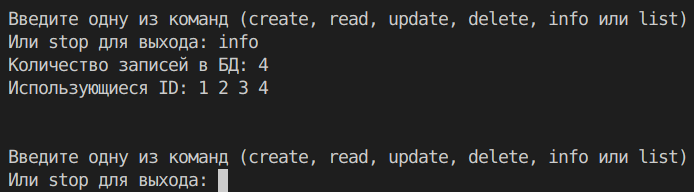


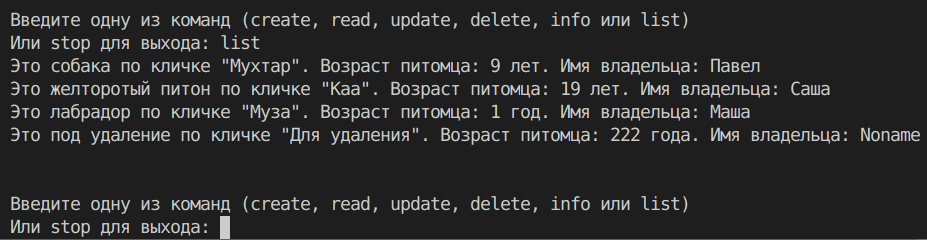
Добавлю двух новых питомцев, введя команду create:



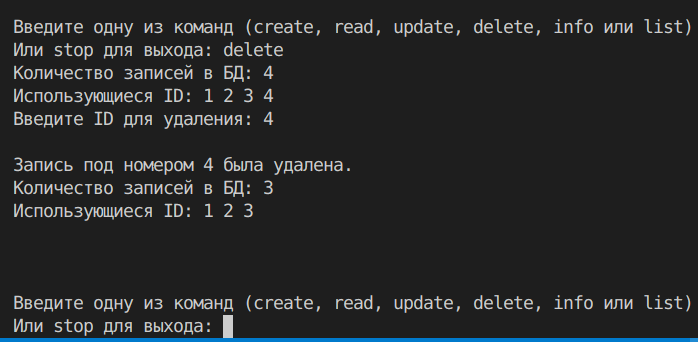


Проверяю информацию по БД и список записей после добавления 2 новых питомцев:

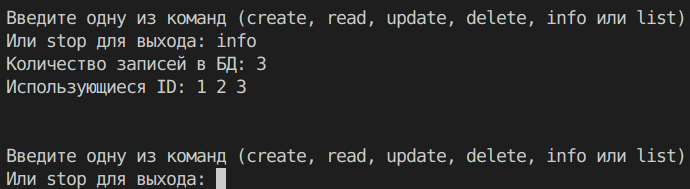


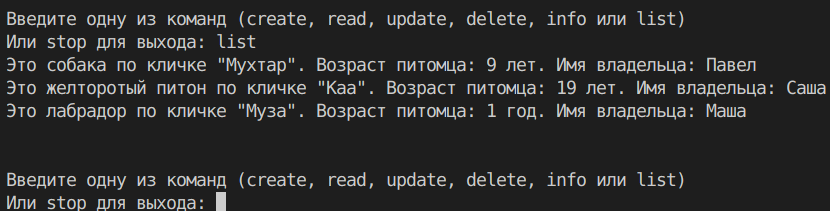


Удалю запись о питомце «под удаление»:

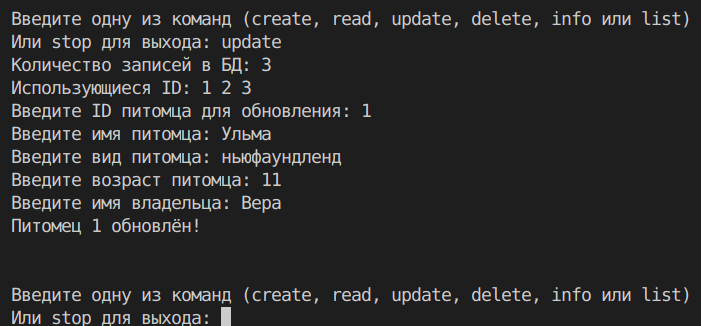


Проверяю состояние БД командами info и list:

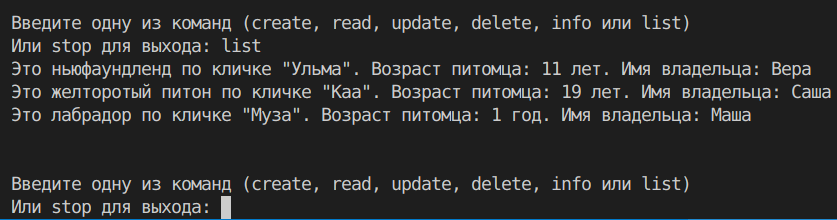




Обновлю запись питомца «Мухтар», введя команду update:



Выведу результирующую записи БД:



Попробуем удалить несуществующую запись:

