# Урок №11. Функции

**Задание №1**

- 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]

**Задание №2**

В Урок №10. Задание №1 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 питомце

**Функция delete**

Данная функция будет удалять запись о существующем питомце

---

Структруа результирующего словаря pets будет как и в Урок №10. Задание №1, 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

Вместе с файлом, отправляемом на проверку домашнего задания, комментарием укажите ссылку на репозиторий GitHub, где хранится ваша программа.