# Лабораторная работа №4

# «ВИДЫ ФУНКЦИЙ В PYTHON. РЕАЛИЗАЦИЯ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ на языке Python»

**Цель работы:**

* Получить практические навыки реализации всех видов функций.
* Получить практические навыки написание регулярных выражений.

**Краткие теоретические сведения**

Функции выполняют три вещи:

1. Они присваивают имена фрагментам кода так, как переменные име­нуют строки и числа.
2. Они принимают аргументы.
3. Учитывая пункты 1 и 2, вы можете создавать собственные «мини-сце­нарии» или «крошечные команды»

Чтобы создать функцию в Python, необходимо воспользоваться зарезервированным словом «def».

Строка объявления функции выглядит следующим образом:

[def] [название функции] ([входные параметры]):

Например: def calc(a, b, operation):.

**Функция как процедура**

Ниже (рис.1) представлены примеры функций с различным количеством входных параметров.

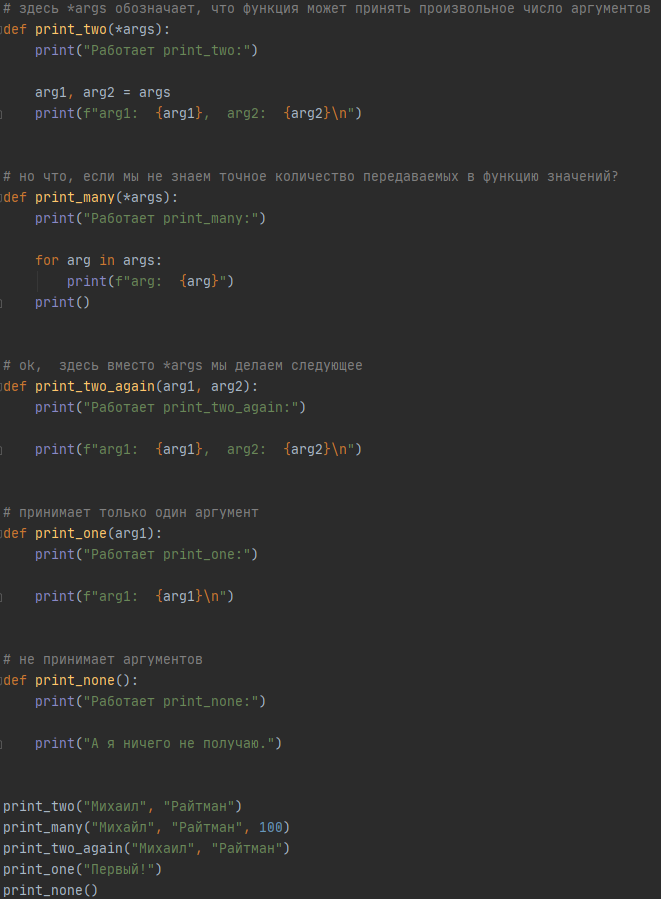
Результат работы данных функций приведен на рис.2.

Рисунок 1. Пример работы с параметрами функции

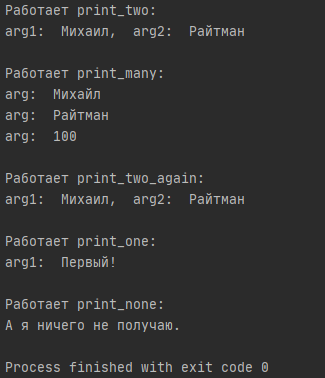


Рисунок 2. Результат работы функций

После двоеточия все строки кода, имеющие отступ в четыре пробела, будут привязаны к указанному имени и будут представлять собой тело функции.

**Функция как функция**

Чтобы изменить поведение функции таким образом, чтобы она возвращала результат выполнения, достаточно в качестве последней строки тела функции указать зарезервированное слово «return» и возвращаемое значение.

В примере на рис.3 приведены функции, возвращающие результат выполнения.

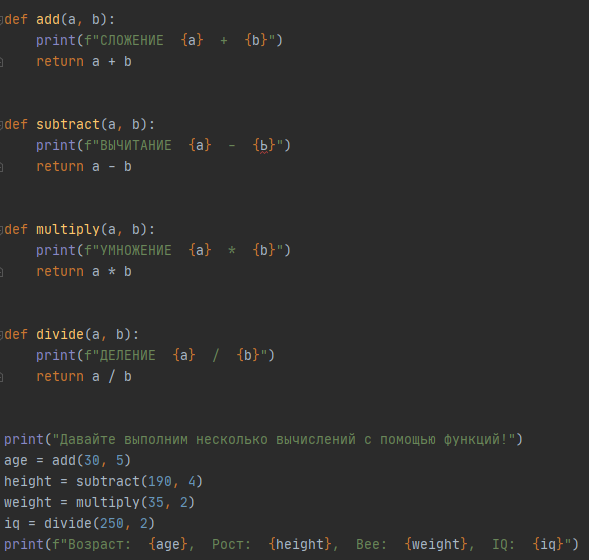
 В приведенном примере (рис.3) в функции add происходит следующее:

Рисунок 3. Функции, возвращающие результат выполнения

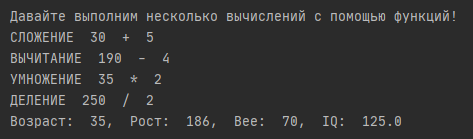
1. Наша функция вызывается с двумя аргументами: а и b.
2. С помощью команды print мы выводим результат выполнения функции, в данном случае, операции сложения.
3. Затем мы сообщаем Python, что нужно вернуть: мы возвращаем результат сложения а + b.
4. Python складывает два числа. Затем, когда функция завершается, результат сложения присваивается в качестве значения переменной и может быть использован далее в коде.

Рисунок 4. Результат работы функций с возвратом значений

На рис. 4 представлен результат выполнения кода из рис.3.

**Анонимные функции**

Помимо обычных, язык Python позволяет использовать анонимные функции, которые также называются лямбда-функциями. Анонимная функция описывается с помощью ключевого слова lambda. После этого могут быть указаны входные параметры для функции. Далее, после двоеточия, указывается тело функции.

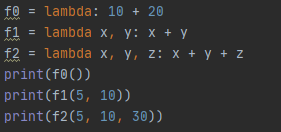
 Например:

Рисунок 5. Анонимные функции

На рис. 6. представлен результат выполнения анонимных функций.

Рисунок 6. Результат выполнения анонимных функций

**Вложенные функции**

Одну функцию можно вложить в другую функцию, причем уровень вложенности не ограничен. При этом вложенная функция получает свою собственную локальную область видимости и имеет доступ к переменным, объявленным внутри функции, в которую она вложена.

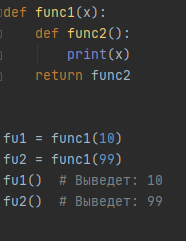
Например:

Рисунок 7. Пример работы вложенных функций

Здесь мы определили функцию func1(), принимающую один параметр. Внутри функции func1() определена вложенная функция func2(). Результатом выполнения функции func1() будет ссылка на эту вложенную функцию. Внутри функции func2() мы производим вывод значения переменной х, которая является локальной в функции func1(). Таким образом, помимо локальной, глобальной и встроенной областей видимости, добавляется вложенная область видимости. При этом поиск идентификаторов вначале производится внутри вложенной функции, затем внутри функции-родителя, далее в функциях более высокого уровня и лишь потом в глобальной и встроенных областях видимости. В нашем примере переменная х будет найдена в области видимости функции func1().

**Глобальные и локальные переменные**

Глобальные переменные – это переменные, объявленные в программе вне функции. В Python глобальные переменные видны в любой части модуля, включая функции.

Если имя локальной переменной совпадает с именем глобальной переменной, то все операции внутри функции осуществляются с локальной переменной, а значение глобальной переменной не изменяется. Локальные переменные видны только внутри тела функции.

Для того чтобы значение глобальной переменной можно было изменить внутри функции, необходимо объявить переменную глобальной с помощью ключевого слова global.

Например:

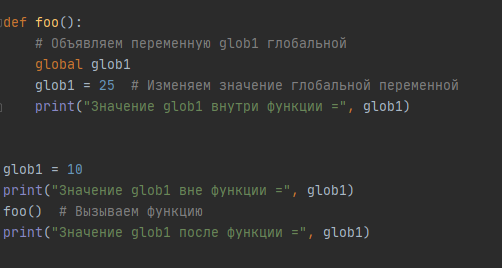


Рисунок . Изменение значения глобальной переменной внутри функции

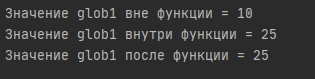
 Результат работы кода приведенного на рис. 8 приведен ниже (рис.9).

Рисунок . Результат изменения

Таким образом, поиск идентификатора, используемого внутри функции, будет производиться в следующем порядке:

1. Поиск во встроенной области видимости (встроенные функции, классы и т. д.).
2. Поиск объявления идентификатора внутри функции (в локальной области видимости).
3. Поиск объявления идентификатора в глобальной области.

**Документирование функции**

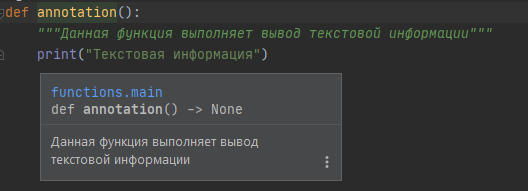
****Документация к функции указывается первой строкой тела функции в тройных кавычках (рис.10).

Рисунок 10. Документирование функции

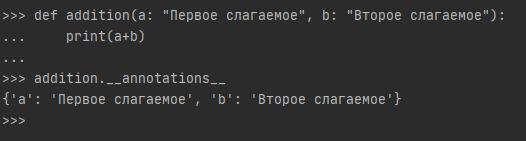
В Python 3 функции могут содержать кроме строк документации, еще и аннотации, которые вводят новый способ документирования. Теперь в заголовке функции допускается указывать предназначение каждого параметра, данные какого типа он может принимать, а также тип возвращаемого функцией значения. Аннотации имеют следующий формат:

Рисунок 11. Пример использования аннотации входных параметров

**Регулярные выражения**

Регулярные выражения предназначены для выполнения в строке сложного поиска или замены. В языке Python использовать регулярные выражения позволяет модуль re. Прежде чем задействовать функции из этого модуля, необходимо подключить модуль с помощью инструкции: import re.

Создать откомпилированный шаблон регулярного выражения позволяет функция compile (). Функция имеет следующий формат:

<Шаблон> = re.compile ( <Регуляр.ное выражение> [, <Модификатор>])

В параметре <Модификатор> могут быть указаны следующие флаги (или их комбинация через оператор |):

* L или LOCALE -учитываются настройки текущей локали;
* I или IGNORECASE - поиск без учета регистра.

Пример:

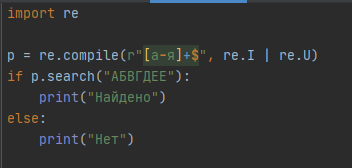


Рисунок . Пример регулярного выражения

Пример использования вложенных функций для создания универсального метода поиска шаблона регулярного выражения в исходной строке представлен на рис. 13

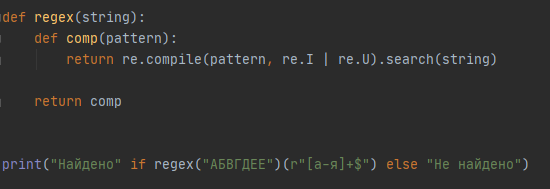


Рисунок 13. Создание универсальной вложенной функции для передачи ей шаблона и исходной строки

**Порядок выполнения работы**

1. Вызов отдельных заданий на выполнение оформить в виде меню
2. Оформить отчет
3. Подготовить ответы на контрольные вопросы

**Задания на лабораторную работу**

Лабораторная работа включает решение следующих типов заданий:

1. Реализация одного задания из лабораторной работы №3 в виде рекурсивной функции;
2. Обработка строки символов;
3. Обработка текста, состоящего из отдельных слов;
4. Обработка регулярных выражения для поиска и преобразования текста;
5. Реализация заданий связанных со строками в виде вложенных функций

Задания для второй части лабораторной работы брать из лабораторной работы №7 курса Java.

**Контрольные вопросы**

1. Какие имена функций допустимы?
2. Как передать в функцию список входных параметров?
3. Для чего нужны функции и какую пользу приносят программисту?
4. Для чего нужна команда return? Приведите пример использования.
5. Какие виды функции вы знаете?
6. В каком случае значение глобальной переменной может измениться в функции?
7. Какой модуль отвечает за работу регулярных выражений?
8. Для каких операций используются регулярные выражения?