

Функции

№ урока: 5 **Курс:** Python Starter

Средства обучения: Python 3.4; интегрированная среда разработки (PyCharm 4 (рекомендуется) или Microsoft Visual Studio 2013 + Python Tools for Visual Studio)

Обзор, цель и назначение урока

В уроке рассматриваются функции.

Изучив материал данного занятия, учащийся сможет:

- Понимать, что такое функции и пользоваться ими
- Создавать собственные функции
- Пользоваться именованными аргументами при вызове функций, задавать аргументы в произвольном порядке
- Пользоваться опциональными параметрами

Содержание урока

1. Понятие функции
2. Именованные параметры при вызове функции
3. Значение аргументов по умолчанию (опциональные параметры)

Резюме

- Функция в программировании — поименованный фрагмент программного кода (подпрограмма), к которому можно обратиться из другого места программы.
- Функции нужны главным образом для избегания повторения одинакового кода.
- Функция может принимать параметры и должна возвращать некоторое значение, возможно пустое. Функции, которые возвращают пустое значение, часто называют процедурами.
- В некоторых языках программирования объявления функций и процедур имеют различный синтаксис, в частности, могут использоваться различные ключевые слова. В Python на уровне языка нет различия между процедурами и функциями.
- Процедуры в Python всегда возвращают значение None. Если функция сама не вернула никакого значения, значение None возвращается автоматически.
- Параметр в программировании — принятый функцией аргумент. Термин «аргумент» подразумевает, что конкретно и какой конкретной функции было передано, а параметр — в каком качестве функция применила это принятое. То есть вызывающий код передает аргумент в параметр, который определен в члене спецификации функции.
- Формальный параметр — аргумент, указываемый при объявлении или определении функции.
- Фактический параметр — аргумент, передаваемый в функцию при ее вызове.
- При вызове функции формальные параметры замещаются фактическими.
- Семантика использования формальных и фактических параметров называется стратегией вычисления. Заданная стратегия вычисления диктует, когда следует вычислять аргументы функции (метода, операции, отношения), и какие значения следует передавать. Существует довольно много разнообразных стратегий вычисления.

- Очень часто понятие стратегии вычисления также называют способом передачи параметров, хотя для некоторых стратегий вычисления (например, «вызов по необходимости») такой термин некорректен.
- Вызов по значению (англ. call-by-value) является наиболее широко распространённой стратегией вычислений, её можно видеть в самых разных языках, от Си до Scheme. При вызове по значению, выражение-аргумент вычисляется, и полученное значение связывается с соответствующим формальным параметром функции (обычно посредством копирования этого значения в новую область памяти). При этом, если язык разрешает функциям присваивать значения своим параметрам, то изменения будут касаться лишь этих локальных копий, но видимые в месте вызова функции значения останутся неизменными по возвращении.
- При вызове по ссылке (англ. call-by-reference), или передаче по ссылке (pass-by-reference), функция неявно получает ссылку на переменную, использованную в качестве аргумента, вместо копии её значения. Обычно это означает, что функция может осуществлять модификацию (то есть изменять состояние) переменной, переданной в качестве параметра, и это будет иметь эффект в вызывающем контексте.
- Вызов по соиспользованию или вызов с разделением ресурсов (англ. call-by-sharing), также вызов по объекту (call-by-object), также вызов по соиспользованию-объекта или вызов с разделяемым объектом (call-by-object-sharing), подразумевает, что значения в языке основаны на объектах, а не на примитивных типах, то есть на том, что все значения являются обёрнутыми (boxed). При вызове по соиспользованию функция получает значение, содержащее копию ссылки на объект. Сам объект не копируется — он оказывается разделяемым, или используемым совместно. Как следствие, присваивание аргументу в теле функции не имеет эффекта в вызывающем её контексте, но присваивание компонентам этого аргумента — имеет.
- Вызов по соиспользованию используется в языках Python, Iota, Java (для ссылок на объекты), Ruby, Scheme, OCaml, AppleScript, и многих других. Однако, терминология в сообществах разных языков различается. Например, в сообществе Python используется термин «вызов по соиспользованию»; в сообществах Java и Visual Basic ту же семантику часто описывают как «вызов по значению, где „значением“ является ссылка на объект»; в сообществе Ruby говорят, что Ruby «использует вызов по ссылке» — несмотря на то, что семантика вызова в этих языках идентична.
- На практике это значит, что при передаче параметров в функции в Python мы не можем привязать фактические параметры к другим объектам или изменить их, если они относятся к неизменяемым типам (см. урок 2), однако можем модифицировать их значения, если они относятся к изменяемым типам, таким как списки, которые будут рассмотрены в уроке 7.
- Не волнуйтесь, если вы не до конца поняли ту часть теории, которая касается стратегий вычисления, так как она приведена лишь для ознакомления и будет подробно разобрана в курсе Python Essential.
- Каждой функции следует выполнять лишь одно логически завершённое действие. Не нужно писать слишком большие функции, лучше разбивать код на небольшие независимые части, которые легко написать, прочитать и протестировать на корректность.
- Объявление функции в Python:


```
def имя_функции(аргумент_1, аргумент_2, аргумент_n):
    операторы
```
- Если функция не принимает никаких значений, пустой список аргументов всё равно выделяется парой круглых скобок.
- Для возврата значения из функции используется оператор return. Его также можно использовать для досрочного завершения работы функции.
- Оператор return без параметров возвращает None.
- Вызов функции: имя_функции(аргумент_1, аргумент_2, аргумент_n)
- При вызове функции фактические параметры замещают формальные в том порядке, в котором они указаны. Можно изменить этот порядок, указав при вызове имена соответствующих формальных параметров: имя_функции(аргумент_2=значение, аргумент_1=значение)

- При вызове функции можно использовать именованные и неименованные аргументы одновременно. В таком случае сначала указываются неименованные фактические параметры, которые замещают формальные в том порядке, в котором в заголовке функции описано соответствующее количество формальных параметров, а затем указываются остальные фактические параметры вместе с именами соответствующих формальных параметров в произвольном порядке.
- Параметры функций можно делать опциональными, то есть необязательными, путём задания им значений по умолчанию:

```
def имя_функции(аргумент_1, аргумент_2=значение):
    операторы
```
- В этом случае, если значение соответствующего параметра не задано при вызове функции, будет использовано стандартное значение.
- В Python также имеется возможность создавать функции от неизвестного количества аргументов, но её использование требует знания и умения работать со словарями и списками, поэтому будет рассмотрено позже.

Закрепление материала

- Что такое функция?
- Что такое процедура?
- Есть ли в Python отличие между функциями и процедурами на уровне языка?
- Зачем нужны функции?
- Что такое формальные параметры функции?
- Что такое фактические параметры функции?
- Как задаются функции в Python?
- Какой оператор используется для возврата значения из функции?
- Каким образом можно задавать фактические параметры в произвольном порядке?
- Что такое опциональные параметры?

Дополнительное задание

Задание

Создайте программу, которая состоит из функции, которая принимает три числа и возвращает их среднее арифметическое, и главного цикла, спрашивающего у пользователя числа и вычисляющего их средние значения при помощи созданной функции.

Самостоятельная деятельность учащегося

Задание 1

Создайте функцию, которая выводит приветствие для пользователя с заданным именем. Если имя не указано, она должна выводить приветствие для пользователя с Вашим именем.

Задание 2

Создайте две функции, вычисляющие значения определённых алгебраических выражений. Выведите на экран таблицу значений этих функций от -5 до 5 с шагом 0.5.

Задание 3

Создайте программу-калькулятор, которая поддерживает четыре операции: сложение, вычитание, умножение, деление. Все данные должны вводиться в цикле, пока пользователь не укажет, что хочет завершить выполнение программы. Каждая операция должна быть реализована в виде отдельной функции. Функция деления должна проверять данные на корректность и выдавать сообщение об ошибке в случае попытки деления на ноль.

Рекомендуемые ресурсы

Документация по Python

https://docs.python.org/3/reference/compound_stmts.html#function-definitions

Статьи в Википедии о ключевых понятиях, рассмотренных на этом уроке

[https://ru.wikipedia.org/wiki/Функция_\(программирование\)](https://ru.wikipedia.org/wiki/Функция_(программирование))

<https://ru.wikipedia.org/wiki/Подпрограмма>

[https://ru.wikipedia.org/wiki/Параметр_\(программирование\)](https://ru.wikipedia.org/wiki/Параметр_(программирование))

https://ru.wikipedia.org/wiki/Стратегия_вычисления