# Функции (часть 2)

| **№ урока:** | 6 | **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 рассматривается более продвинутое использование функций, рассматривается понятие рекурсии, даётся обзор некоторых стандартных функций языка Python.

## Изучив материал данного занятия, учащийся сможет:

- Создавать особый вид комментариев – документационные строки
- Использовать стандартные функции языка Python
- Понимать разницу между локальными и глобальными переменными
- Использовать рекурсию

## Содержание урока

1. Документационные строки (docstrings)
2. Встроенные функции
3. Локальные и глобальные переменные
4. Рекурсия

## Резюме

- Строка, стоящая в самом начале функции (а также модуля, класса или метода), играет роль особого вида комментариев – документационной строки (docstring).
- В отличие от обычных комментариев, к документационным строкам можно получить доступ во время выполнения программы.
- Напрямую доступ к документационным строкам осуществляется путём обращения к полю __doc__ соответствующих объектов:

  имя_функции.__doc__
- Обратите внимание на отсутствие круглых скобок после имени функции. Круглые скобки – это операция вызова функции, но при их отсутствии сама функция рассматривается как некий объект.
- При работе с интерпретатором в интерактивном режиме удобно использовать функцию help.
- Стандартная библиотека Python содержит огромное количество различных модулей и библиотек. Но есть и особо важные стандартные функции, которые встроены в сам язык и доступны всегда без импортирования.
- Некоторые из них:
  - abs(x) – модуль числа x
  - bin(x) – представление числа x в двоичной системе счисления
  - bool(x) – создать значение типа bool из x
  - callable(f) – проверить, можно ли f вызывать как функцию
  - chr(code) – символ с кодом code
  - complex(real, imag) – создать комплексное число
  - dir(obj) – вывести список полей и методов объекта obj
  - float(x) – создать вещественное число из x



CyberBionic systematics

CyberBionic Systematics ®
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

    - format(x, fmt) – возвращает строку, представляющую значение x, отформатированное согласно форматной строке fmt
    - help(obj) – выводит справку по объекту obj
    - hex(x) – шестнадцатеричное представление числа x
    - id(obj) – значение, уникальное для каждого объекта (в CPython – адрес объекта в памяти)
    - input() – ввод данных
    - int(x) – создать целое число
    - len(s) – длина строки или любой другой последовательности
    - max(arg1, arg2, …) – максимальное число среди заданных
    - min(arg1, arg2, …) – минимальное число среди заданных
    - oct(x) – представление числа x в восьмеричной системе счисления
    - ord(c) – код символа c
    - pow(x, n) – число x в степени n
    - print() – вывод данных
    - range() – последовательность целых чисел (см. урок про циклы)
    - repr(obj) – строковое представление объекта
    - reversed(iterable) – обход последовательности в обратном порядке
    - round(number, ndigits) – округление числа
    - sorted(iterable) – сортирует последовательность в порядке возрастания или убывания значений
    - str(x) – создание строки из x
- Полный список можно найти в документации в главе Built-in Functions раздела Library Reference.
- В Python можно объявлять функции внутри функций.
- Область видимости (англ. scope) обозначает область программы, в пределах которой идентификатор (имя) некоторой переменной продолжает быть связанным с этой переменной и возвращать её значение. За пределами области видимости тот же самый идентификатор может быть связан с другой переменной, либо быть свободным (не связанным ни с какой из них).
- В языках, поддерживающих структурное программирование, переменные обычно разделяются на два типа по области видимости:
    - локальные переменные — объявляются внутри функции и недоступны снаружи неё;
    - глобальные переменные — объявляются вне всех функций и доступны отовсюду.
- Использование глобальных переменных имеет недостатки: глобальная переменная может быть изменена в любой точке программы, что может повлиять на работу других частей программы. По этой причине глобальные переменные имеют неограниченный потенциал для создания взаимных зависимостей, что приводит к усложнению программы.
- Переменные локальной области видимости используются, чтобы избежать проблем с побочными эффектами, которые могут произойти с глобальными переменными.
- Глобальные переменные широко используются для передачи данных между секциями кода, которые не участвуют в отношениях вызовов, такие как параллельные нити исполнения или обработчики сигналов.
- В Python областью видимости локальной переменной является функция. В некоторых языках любой блок кода может иметь свои локальные переменные.
- Операция присвоения в функции создаёт локальную переменную. Если необходимо изменить значение переменной из другой области видимости, следует воспользоваться операторами global или nonlocal.
- Переменные, указанные в операторе global, рассматриваются как глобальные.
- Переменные, указанные в операторе nonlocal, рассматриваются как переменные из ближайшей области видимости (внешняя функция в случае вложенных функций или глобальная область видимости, если сама функция является глобальной).
- Память под локальные переменные выделяется при каждом вызове функции. Это делает возможным рекурсию.



CyberBionic systematics

CyberBionic Systematics ®
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Рекурсия — вызов функции из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная или косвенная рекурсия), например, функция A вызывает функцию B, а функция B — функцию A.
- Количество вложенных вызовов функции или процедуры называется глубиной рекурсии.
- Рекурсивная программа позволяет описать повторяющееся или даже потенциально бесконечное вычисление, причём без явных повторений частей программы и использования циклов.
- Реализация рекурсивных вызовов функций в практически применяемых языках, как правило, опирается на механизм стека вызовов — адрес возврата и локальные переменные функции, записываются в стек, благодаря чему каждый следующий рекурсивный вызов этой функции пользуется своим набором локальных переменных и за счёт этого работает корректно. Оборотной стороной этого довольно простого по структуре механизма является то, что на каждый рекурсивный вызов требуется некоторое количество оперативной памяти компьютера, и при чрезмерно большой глубине рекурсии может наступить переполнение стека вызовов.
- Вопрос о желательности использования рекурсивных функций в программировании неоднозначен: с одной стороны, рекурсивная форма может быть структурно проще и нагляднее, в особенности, когда сам реализуемый алгоритм по сути рекурсивен. С другой стороны, обычно рекомендуется избегать рекурсивных программ, которые приводят (или в некоторых условиях могут приводить) к слишком большой глубине рекурсии.
- В рекурсивных функциях должно быть условие выхода из рекурсии, то есть как минимум одна из ветвей кода обязана возвращать результат, а не вызывать функцию рекурсивно.

## Закрепление материала

- Что такое документационная строка?
- Можно ли в Python объявлять функции внутри других функций?
- Назовите несколько стандартных функций языка Python.
- Что такое область видимости?
- Что такое локальная переменная?
- Что такое глобальная переменная?
- Что такое рекурсия?
- Что необходимо для того, чтобы рекурсивная функция выполнилась за конечное время?

## Дополнительное задание

Задание
Напишите рекурсивную функцию, которая вычисляет сумму натуральных чисел, которые входят в заданный промежуток.

## Самостоятельная деятельность учащегося

Задание 1
Прочитайте в документации по языку Python информацию о перечисленных в резюме данного урока стандартных функциях. Проверьте их на практике.

Задание 2
Создайте программу, которая проверяет, является ли палиндромом введённая фраза.

Задание 3
Пусть на каждую ступеньку лестницы можно стать с предыдущей или переступив через одну. Определите, сколькими способами можно подняться на заданную ступеньку.



CyberBionic systematics

CyberBionic Systematics ®
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

## Рекомендуемые ресурсы

Документация по Python
https://docs.python.org/3/library/functions.html
https://docs.python.org/3/reference/simple_stmts.html#the-global-statement
https://docs.python.org/3/reference/simple_stmts.html#the-nonlocal-statement

Статьи в Википедии о ключевых понятиях, рассмотренных на этом уроке
https://ru.wikipedia.org/wiki/Область_видимости
https://ru.wikipedia.org/wiki/Локальная_переменная
https://ru.wikipedia.org/wiki/Глобальная_переменная
https://ru.wikipedia.org/wiki/Рекурсия



CyberBionic systematics

CyberBionic Systematics ®
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8937
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com