# Функции

## Видео 1. Функции в Python

## Цели занятия

- Познакомиться с понятием функции и узнать, как она помогает писать код в Python правильно и понятно.
- На практике увидеть, как прописывают функции в реальном коде.
- Разобрать свойства функции и на практике увидеть, как они позволяют использовать одни и те же переменные в разных частях кода.
- Рассмотреть приёмы, упрощающие написание кода в реальных задачах.
- Узнать, как задавать неизвестное количество аргументов в функции.

## Что такое функции

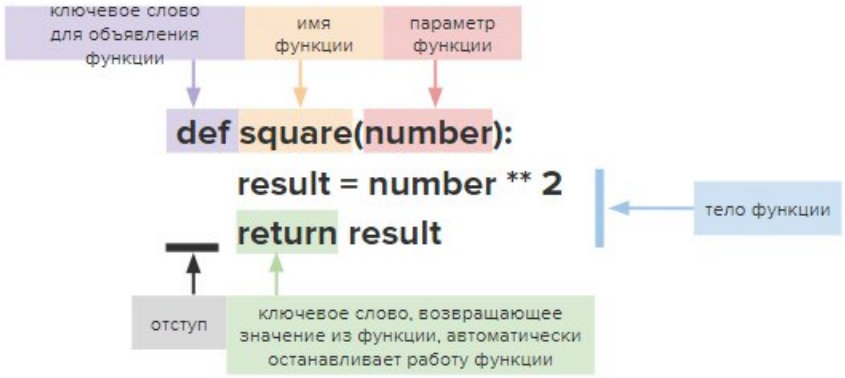
В математике:
- соответствие между элементами;
- то, как значение одной величины определяет значение другой.

В программировании:
- обособленный участок кода, который можно вызывать, обратившись к нему по имени (подпрограмма);
- объект, принимающий аргументы и возвращающий значение.

Функции помогают избежать дублирования кода, улучшить его структуру и читаемость.

## Объявление функции в Python

Функция определяется оператором def. Блок кода внутри каждой функции начинается с двоеточия (:) и должен иметь отступ (пробел). Любые аргументы или входные параметры помещаются в круглые скобки. После объявления функции перед кодом должен быть хотя бы один отступ.




- **def** — обязательный служебный оператор для написания функции.

- **square** — имя функции. Мы можем придумать его сами. Главное, чтобы оно отражало суть.
- **(number)** — параметр функции; всегда находится в круглых скобках (это входной параметр, который мы задаём).
- _ **(отступ обязателен)** — прописываем тело функции.
- **return** — ключевое слово. Определяет, какой результат на выходе будет у функции.

## Вспомогательные полезные функции

**Функция help () вызывает справку о нужной функции.**

**Docstring** (*documentation string*, строка документации) — встроенное средство документирования модулей, функций, классов и методов. Сразу после определения указывают строковое значение, которое и будет **docstring**.

## Параметры функций

Функция принимает на входе какой-либо параметр и на выходе даёт результат использования этого параметра. Параметр может быть из внешнего кода, текущего времени или тот, который мы ей задаём.

- Функция может принимать более одного параметра, а может не принимать параметры вообще.
- Для всех параметров функций можно указывать значения по умолчанию — это даёт возможность вызвать функцию с меньшим числом параметров.

## Тип данных None

**None** — специальный тип данных, который означает отсутствие значения.

Если в функции нет ***return*** или он пустой, она возвращает ***None***.

## Вывод

Функции помогают избежать дублирования кода, улучшить его структуру и читаемость. У каждой функции есть структура написания. В функции можно вводить либо не вводить дополнительные параметры — в зависимости от того результата, который мы хотим получить.

# Видео 2. Области видимости

## Что такое области видимости

**Область видимости (scope)** определяет контекст объекта, в рамках которого его можно использовать.

Типы области видимости:

- глобальная область видимости — переменная является глобальной, она определена вне любой из функций и доступна любой функции в программе;
- локальная — определяется внутри функции и доступна только из этой функции, то есть имеет локальную область видимости.

Если Python не может найти нужную переменную в локальной области видимости, тогда он будет искать её в области видимости уровня выше с помощью оператора global. Этот оператор можно использовать только в крайнем случае (если по-другому не получается), чтобы не сделать ошибки в коде.

## Анонимные функции

Анонимные функции создают при помощи инструкции ***lambda*** и используют для более краткой записи функций с одним выражением. Выполняются быстрее обычных и не требуют инструкции ***return***:
***lambda x, pow: x**pow.***

## Методы объекта

В Python это функции, которые принадлежат определённому объекту. У каждого объекта есть свои методы.

Примеры методов **списков**:

- *.index(),*
- *.count(),*
- *.append(),*
- *.remove(),*
- *.reverse().*

Примеры методов **строк**:

- *.capitalize(),*
- *.upper(),*
- *.lower(),*
- *.replace(),*
- *.count().*

Примеры методов **словарей**:

- *.keys(),*
- *.values(),*
- *.items().*

## Вывод

В Python можно использовать одни и те же переменные, находящиеся как в глобальном коде, так и локально внутри функции. Работа в Python максимально безопасна для внешних переменных, т. к. используемые внутри функции переменные без оператора global не влияют на внешний код. Если есть простые преобразования, можно написать код без полноценного объявления функций с помощью метода lambda.

# Видео 3. Аргументы в функциях

## Args и kwargs

1. Позиционные переменные — это аргументы, которые идут по порядку.
2. * args — это сокращение от «arguments» (аргументы).
3. **kwargs — сокращение от «keyword arguments» (именованные аргументы).
4. Каждая из этих конструкций используется для распаковки аргументов соответствующего типа, позволяя вызывать функции со списком аргументов переменной длины.
5. Оператор ***** (звёздочка) позволяет распаковывать объекты, внутри которых хранятся некие элементы.
6. «args» — это всего лишь набор символов, которым принято обозначать аргументы. Главное здесь — это оператор *****. А то, что именно идёт после него, особой роли не играет. Благодаря использованию ***** мы создали список позиционных аргументов на основе того, что было передано функции при вызове.
7. **kwargs — это два символа ******. Благодаря им создаётся словарь, в котором содержатся именованные аргументы, переданные функции при её вызове.

## Вывод

Если мы заранее не можем знать, какое количество параметров надо передать, можно использовать аргументы args или kwargs — в зависимости от того, идут данные просто по порядку или им нужно задать имя.

## Общие итоги

1. Познакомились с понятием функции и узнали, как она помогает писать код в Python правильно и понятно.

2. На практике увидели, как прописывать функции в реальном коде.

3. Разобрали свойства функции — области видимости данных и анонимные функции. На практике увидели, как они позволяют использовать одни и те же переменные в разных частях кода.

4. Рассмотрели приёмы, упрощающие написание кода в реальных задачах.

5. Узнали, как задавать неизвестное количество аргументов в функции.