GeekBrains

Погружение в Python

Урок 4
Функции

Содержание урока

План курса

1. Основы Python
2. Простые типы данных
3. Коллекции
4. Функции
5. Итераторы и генераторы
6. Модули
7. Файлы и файловая система
8. Сериализация

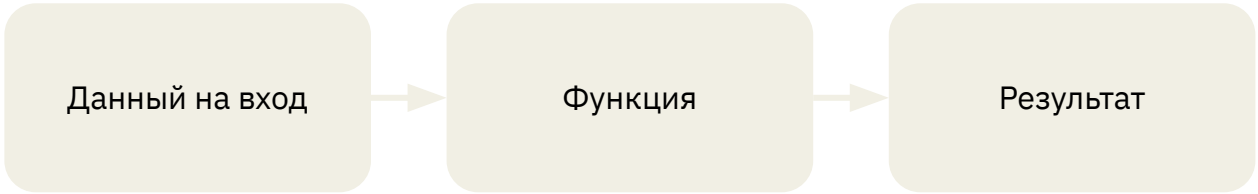
Что будет на уроке сегодня

- 📌 Разберёмся с созданием собственных функций в Python.
- 📌 Изучим работу встроенных функций «из коробки».

Создание своих функции

Функция

Функция — фрагмент программного кода, к которому можно обратиться из другого места программы.



Функция как чёрный ящик

Функция высшего порядка

Функцией высшего порядка называется такая функция, которая принимает функцию-объект как аргумент или возвращает функцию-объект в виде выходного значения.

1

Функция() — вызываем

✔ print(42)

2

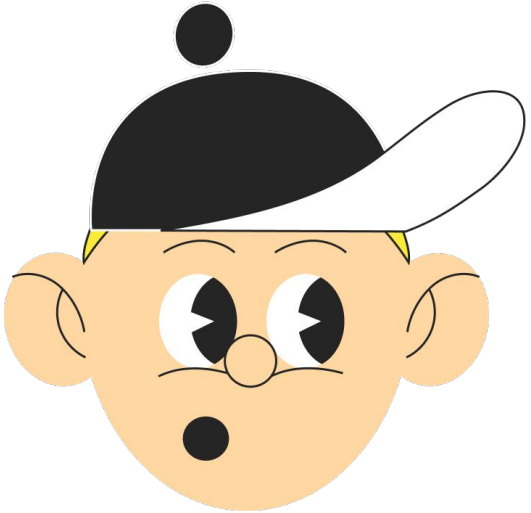
Функция — передаём

✔ func = sum

Определение собственной функции

Зарезервированное слово def

```
def my_func():
    pass
```



Что такое pass?

Зарезервированное слово pass ничего не делает

Плохо

```
if a != 5:
    pass
else:
    a += 1
```

Лучше

```
if a == 5:
    a += 1
else:
    pass
```

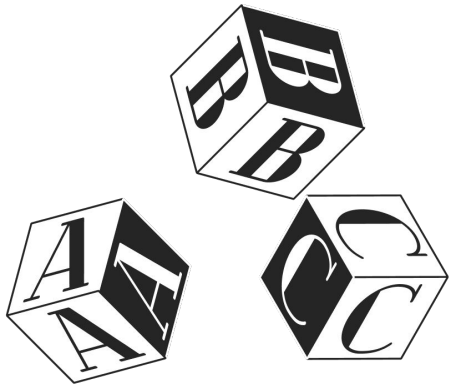
Отлично

```
if a == 5:
    a += 1
```

Аргументы функции

После имени функции в круглых скобках указываются параметры функции через запятую

```
def quadratic_equations(a, b, c):
```



Изменяемые и неизменяемые аргументы

При передаче аргументов в функцию стоит помнить изменяемого типа объект или нет

1

Неизменяемый аргумент

При изменении внутри функции остаётся прежним вне функции

2

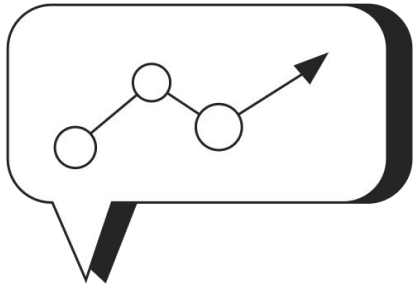
Изменяемый аргумент

При изменении внутри функции изменяется и за её пределами

💡 В Python аргументы передаются внутри функции по ссылке на объект!

Возврат значения из функции, return

- **Если указан один объект после return**
  возвращается именно этот объект
- **Если указано несколько значений через запятую после return**
  возвращается кортеж с перечисленными значениями
- **Если ничего не указано после return**
  возвращается None
- **Если return отсутствует**
  Python «мысленно» дописывает в качестве последней строки функции `return None`

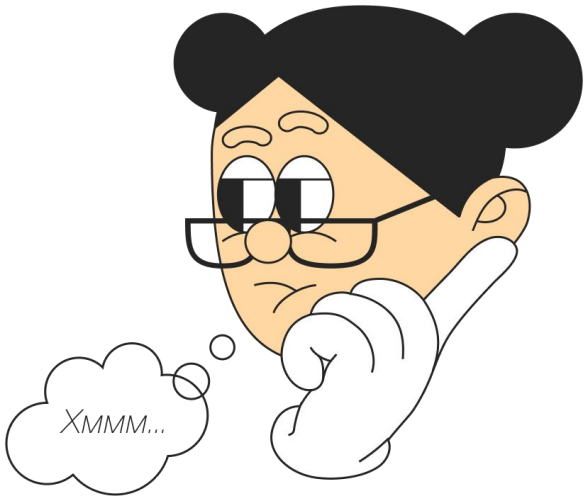


Значения по умолчанию

После параметра указываем значения по умолчанию

```
def quadratic_equations(a, b=0, c=0):
```

💡 В качестве значения по умолчанию нельзя указывать изменяемы типы: списки, словари, множества и т.п.!



Позиционные и ключевые параметры

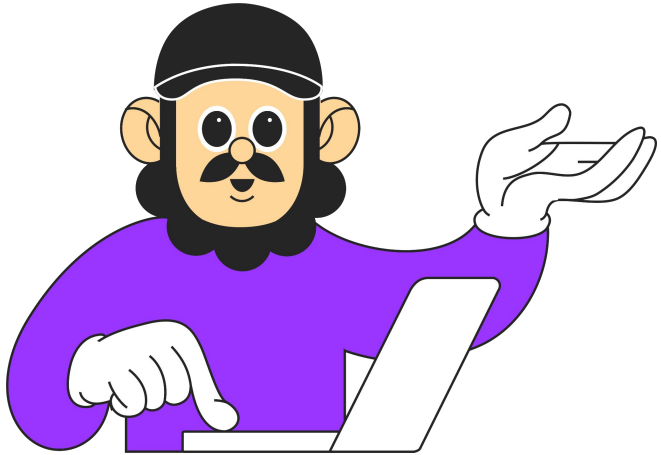
Косая черта / и звёздочка * разделяют позиционные и ключевые параметры

```
def func(positional_only, /, positional_or_keyword, *, keyword_only):
    pass
```

Параметры *args и **kwargs

Имена args и kwargs — общепринятое соглашение

- ✔ def func(*args): — принимает любое число позиционных аргументов
- ✔ def func(**kwargs): — принимает любое число ключевых аргументов
- ✔ def func(*args, **kwargs): — принимает любое число позиционных и ключевых аргументов



Области видимости

💡 **Локальная**
Код внутри самой функции, т.е. переменные заданные в теле функции

💡 **Глобальная, global**
Код модуля, т.е. переменные заданные в файле ру содержащем функцию

💡 **Не локальная, nonlocal**
Код внешней функции, исключающий доступ к глобальным переменным

📌 Доступ к глобальной **константе** из тела функции — нормально!

Анонимная функция lambda

✔ **lambda parameters: action**

Анонимные функции, они же лямбда функции — синтаксический сахар для обычных питоновских функций с рядом ограничений. Они позволяют задать функцию в одну строку кода без использования других ключевых слов.

Документирование кода функций

Пишется сразу после определения функции.
Пояснения к однострочной строке документации:

- ✔ Тройные кавычки используются, даже если строка помещается на одной строке. Это позволяет легко расширить его позже.
- ✔ Закрывающие кавычки находятся на той же строке, что и открывающие. Это выглядит лучше для однострочников.
- ✔ Нет пустой строки ни до, ни после строки документации.
- ✔ Строка документации — это фраза, заканчивающаяся точкой. Он описывает действие функции или метода как команду.
- ✔ Однострочная строка документации не должна повторять параметры функции.

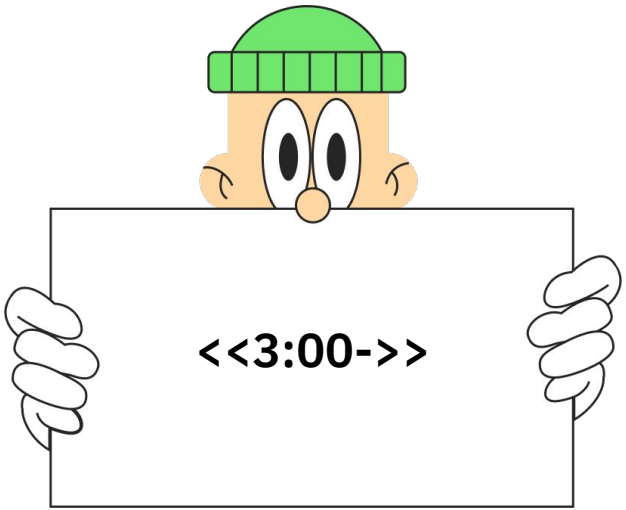
Перед вами функция и её вызов.
Найдите три ошибки и напишите о них в чат.

У вас 3 минуты.

Функции

```
def func(a=0.0, /, b=0.0, *, c=0.0):
    """func(a=0.0: int | float, /, b=0.0: int | float, *, c=0.0: int | float) -> : int | float"""
    if a > c:
        return a
    if a < c:
        return c
    return

print(func(c=1, b=2, a=3))
```



Функции «из коробки»

Встроенные функции

В Python есть ряд встроенных функций. Перечислим их в алфавитном порядке:

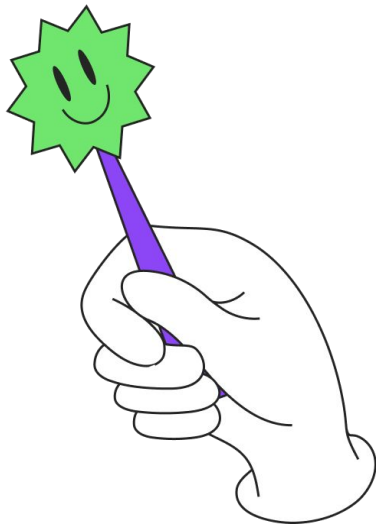
`abs(), aiter(), all(), any(), anext(), ascii(), bin(), bool(), breakpoint(), bytearray(), bytes(), callable(), chr(), classmethod(), compile(), complex(), delattr(), dict(), dir(), divmod(), enumerate(), eval(), exec(), filter(), float(), format(), frozenset(), getattr(), globals(), hasattr(), hash(), help(), hex(), id(), input(), int(), isinstance(), issubclass(), iter(), len(), list(), locals(), map(), max(), memoryview(), min(), next(), object(), oct(), open(), ord(), pow(), print(), property(), range(), repr(), reversed(), round(), set(), setattr(), slice(), sorted(), staticmethod(), str(), sum(), super(), tuple(), type(), vars(), zip().`

Повторяем map(), filter(), zip()

💡 **map(function, iterable)**
Принимает на вход функцию и последовательность. Функция применяется к каждому элементу последовательности и возвращает map итератор.

💡 **filter(function, iterable)**
Принимает на вход функцию и последовательность. Если функция возвращает истину, элемент остаётся в последовательности. Как и map возвращает объект итератор.

💡 **zip(*iterables, strict=False)**
Принимает несколько последовательностей и итерируется по ним параллельно. Если передать ключевой аргумент strict=True, вызовет ошибку ValueError в случае разного числа элементов в каждой из последовательностей.



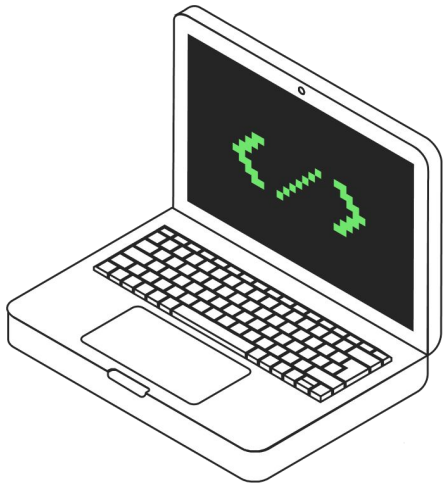
Функции max(), min(), sum()

Разберём на примерах

- `max(iterable, *[, key, default])` или `max(arg1, arg2, *args[, key])` Функция принимает на вход итерируемую последовательность или несколько позиционных элементов и ищет максимальное из них.
- `min(iterable, *[, key, default])` или `min(arg1, arg2, *args[, key])` Функция работает аналогично max, но ищет минимальный элемент.
- `sum(iterable, /, start=0)`
  Функция принимает объект итератор и подсчитывает сумму всех элементов. Ключевой аргумент start задаёт начальное значение для суммирования.

Функции all(), any()

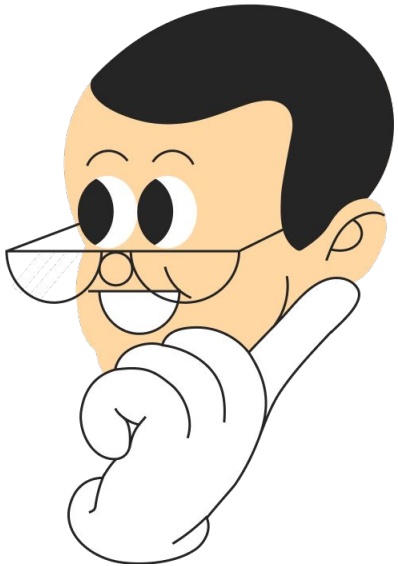
- ✔ `all(iterable)` Функция возвращает истину, если все элементы последовательности являются истиной.
- ✔ `any(iterable)` Функция возвращает истину, если хотя бы один элемент последовательности являются истиной.



Функции chr(), ord()

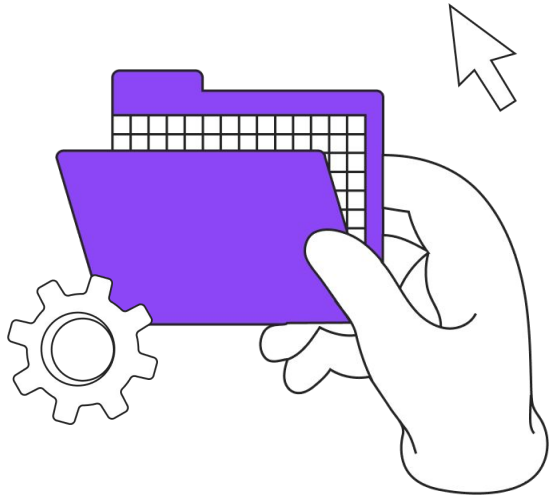
Разберём на примерах

- ✔ `chr(integer)` Функция возвращает строковой символ из таблицы Юникод по его номеру. Номер — целое число от 0 до 1_114_111.
- ✔ `ord(char)` Функция принимает один символ и возвращает его код в таблице Юникод.



Функции locals(), globals(), vars()

- ✔ `locals()` Функция возвращает словарь переменных из локальной области видимости на момент вызова функции.
- ✔ `globals()` Функция возвращает словарь переменных из глобальной области видимости, т.е. из пространства модуля.
- ✔ `vars()` Функция без аргументов работает аналогично функции locals(). Если передать в vars объект, функция возвращает его атрибут __dict__. А если такого атрибута нет у объекта, вызывает ошибку TypeError.

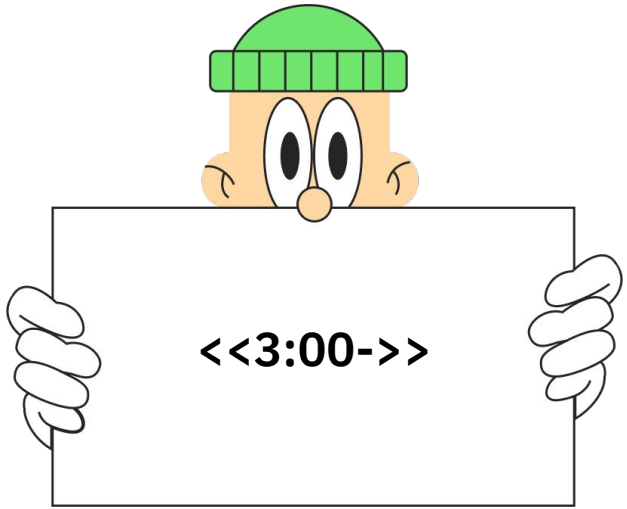


Перед вами несколько строк кода.
Напишите в чат, что они вернут,
не запуская программу.

У вас 3 минуты.

Встроенные функции

```
data = [25, -42, 146, 73, -100, 12]
print(list(map(str, data)))
print(max(data, key=lambda x: -x))
print(*filter(lambda x: not x[0].startswith('__'),
globals().items()))
```



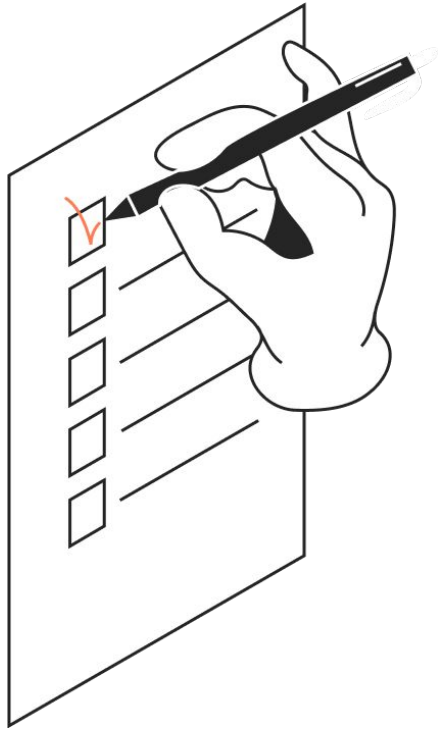
Итоги занятия

На этой лекции мы

- 📌 Разобрались с созданием собственных функций в Python.
- 📌 Изучили работу встроенных функций «из коробки».

Задание

Поработайте со справочной информацией в Python, функцией help(). Попробуйте найти информацию об изученных на уроке функциях «из коробки». Почитайте описание тех функций, которые не рассматривались на уроке.



Спасибо за внимание