Python: Возврат значений

В этом уроке подробнее разберем, как работать с созданными функциями, чтобы они были полезны.

Когда мы определяем функцию, она печатает на экран какие-то данные:

```
def greeting():
    print('Hello, Hexlet!')
```

Пользы от таких функций немного, так как результатом нельзя воспользоваться внутри программы. Рассмотрим на примере.

Возьмем задачу обработки электронной почты. Когда пользователь регистрируется на сайте, то он может ввести email любым способом:

Если мы сохраним его в таком виде в базу данных, то пользователь не войдет на сайт. Чтобы этого не произошло, email нужно подготовить к записи в базу: привести его к нижнему регистру и обрезать пробелы по краям строки. Такая задача решается в пару строчек:

```
def save_email():
    # Email приходит из формы
    email = '  SuppORT@hexlet.IO'
```

```
    # Обрезаем пробельные символы
    trimmed_email = email.strip()
    prepared_email = trimmed_email.lower()
    print(prepared_email)
    # Здесь будет запись в базу данных
```

Этот код стал возможен благодаря тому, что значение вернулось. Методы `strip()` и `lower()` ничего не печатают на экран, они **возвращают** результат своей работы. Поэтому мы можем записать его в переменные. Если бы они печатали на экран, мы бы не могли присвоить результат переменной. Например, так мы не можем сделать с функцией `greeting()`:

```
message = greeting()
# в действительности, функция print() возвращает None
# None — специальный объект, используемый для представления
print(message) # => None
```

Теперь изменим функцию `greeting()` так, чтобы она возвращала данные. Для этого выполним возврат вместо печати на экран:

```
def greeting():
    return 'Hello, Hexlet!'
```

`return` — это инструкция. Она берет записанное справа выражение и отдает его тому коду, который

вызвал метод. Здесь выполнение функции завершается.

```
# Теперь мы можем использовать результат работы функции
message = greeting()
print(message) # => Hello, Hexlet!
# И даже выполнить какие-то действия над результатом
print(message.upper()) # => HELLO, HEXLET!
```

Любой код после `return` не выполняется:

```
def greeting_with_code_after_return():
    return 'Hello, Hexlet!'
    print('Я никогда не выполнюсь')
```

Даже если функция возвращает данные, это не ограничивает ее в том, что она печатает. Кроме возврата данных мы можем и печатать:

```
def greeting_with_return_and_printing():
    print('Я появлюсь в консоли')
    return 'Hello, Hexlet!'


# И напечатает текст на экран, и вернет значение
message = greeting_with_return_and_printing()
```

Возвращать можно не только конкретное значение. Так как `return` работает с выражениями, то справа от

него может быть что угодно. Здесь нужно руководствоваться принципами читаемости кода:

```
def greeting():
    message = 'Hello, Hexlet!'
    return message
```

Здесь мы не возвращаем переменную. Возвращается всегда значение, которое находится в этой переменной. Ниже пример с вычислениями:

```
def double_five():
    # или return 5 + 5
    result = 5 + 5
    return result
```

Определить функцию мало. Еще важно, чтобы она была полезна, и результатом можно было воспользоваться. А теперь подумайте, что вернет вызов, определенной ниже функции `run()`?

```
# Определение
def run():
    return 5
    return 10


# Что будет выведено на экран?
print(run())
```

https://replit.com/@hexlet/python-basics-define-functions-return

Задание

Реализуйте функцию `say_hurray_three_times()`, которая возвращает строку 'hurray! hurray! hurray!'.

```
hurray = say_hurray_three_times()
print(hurray) # => hurray! hurray! hurray!
```

- ▶ Упражнение не проходит проверку — что делать? 😶
- ▶ В моей среде код работает, а здесь нет 😐
- ▶ Мой код отличается от решения учителя 🤔
- ▶ Прочитал урок — ничего не понятно 🙄

Полезное

- return