Испытания

# Python: Словарь с псевдонимами

Python: Введение в ООП  4 сообщения

Обновлено: 31 марта, 15:47

91
Студент

83%
Завершения

Начать испытание

Как вы уже знаете, в Python многое работает на основе протоколов. Обращение по индексу или ключу — через те самые квадратные скобки — тоже работает с любыми объектами, реализующими **"subscription protocol"**.

Этот протокол требует, чтобы у объекта имелись нужного вида методы `__getitem__()` и `__setitem__()`, первый из которых позволяет получить значение по ключу или индексу, а второй сохранить значение по указанному ключу. Вот так эти методы работают для словаря:

```
d = {}
d = {'a': 42}
d.__getitem__('a')       # то есть d['a']
# 42
d.__setitem__('a', 100)  # то есть d['a'] = 100
print(d)  # => {'a': 100}
d.__setitem__('a', d.__getitem__('a') + 5)
# а это уже d['a'] += 5
```

Подробнее о методах можно почитать здесь:

- getitem()
- setitem()

## src/solution.py

Вам нужно реализовать класс `MultiKeyDict`, который должен уметь работать по протоколу `__getitem__()` / `__setitem__()`

протоколу `__getitem__()`/`__setitem__()`.

Класс принимает именованные параметры, которые становятся первыми ключами и значениями. В этом класс максимально похож на `dict`. А вот метод `alias()` уже является отличием: вызывая этот метод с параметрами "новый*ключ='старый*ключ'", вы создаёте *псевдонимы* для существующих ключей.

Обращение по созданному псевдониму ничем не отличается от обращения по оригинальному ключу, то есть с момента создания псевдонима у значения становится *два ключа* (или больше, конечно же).

Важно, что с помощью метода `alias()` должно быть возможно "перекинуть" старые ключи на новые значения *без потери* этих самых значений, если останется хотя бы один псевдоним, всё ещё ссылающийся на значение.

> При перебрасывании последнего ключа, некие значения могут остаться вообще без ключа. По условиям задачи вам не нужно производить чистку таких значений, но вы можете попробовать реализовать и такую функциональность!

```
mkd = MultiKeyDict(x=100, y=[10, 20])
mkd.alias(z='x')  # 'z' теперь означает то же, что и 'x'
print(mkd['z'])
# => 100
mkd['z'] += 1  # Можно даже менять значение через присваивание,
print(mkd['x'])       # что затронет и оригинальный ключ.
# => 101
mkd.alias(z='y')  # Теперь 'z' уже равнозначен 'y'
mkd['z'] += [30]
print(mkd['y'])
# => [10, 20, 30]
```

## Последние решения

| Автор | Дата обновления | Версий | |
|---|---|---|---|
| noboribetsu | 04 июля, 10:32 | 1 | ~~Смотреть~~ |
| dmitrii_morozov | 27 июня, 16:15 | 1 | ~~Смотреть~~ |
| odhako | 19 июня, 13:07 | 1 | ~~Смотр~~ |
| user_fa40288a25ddf4a1 | 12 июня, 20:03 | 1 | ~~Смотреть~~ |

| | | | |
|---|---|---|---|
| user-1a40288a25dd14a1 | 12 июня, 20:03 | 1 | ~~Смотреть~~ |
| rezajkee | 01 июня, 11:44 | 1 | ~~Смотреть~~ |




8 800 100 22 47 бесплатно по РФ

+7 495 085 28 38 бесплатно по Москве

Hexlet Ltd.
Itälahdenkatu 22 A,
00210 Helsinki, Finland
VAT ID: FI26641607

Подписаться

**Помощь**

Справка

Вопросы и ответы

support@hexlet.io

Улучшить Хекслет

**Наши проекты**

Хекслет Колледж

Code Basics

Codebattle

Hexlet Guides

Хекслет-резюме