


ХОЧУ ПОМОЧЬ  
ПРОЕКТУ



practicum.yandex.ru

Станьте специалистом по Data Science на курсе от Яндекса

5,0 Рейтинг организации

Узнать больше

РЕКЛАМА • 16+

Страница 3. / Модуль python-benedict в Python

Служба поддержки

Модуль python-benedict

методов модификации и извлечения данных

неофициальная документация

Описание

является собой обертку [стандартного словаря Python](#), со множеством дополнительных встроенных методов для создания словаря из 'сырых данных', быстрой модификации и извлечения данных по мере необходимости, а также возможности преобразования словаря в различные форматы и т.д.

**python-benedict:**

- 100% обратно совместимый, можно безопасно обернуть существующие словари.
- [Поддержка KeyAttr](#) для получения/установки значений по ключам `benedict.x.y.z`.
- [Поддержка KeyPath](#) для получения/установки значений по ключам `benedict['x.y.z']`.
- [Поддержка KeyList](#) для получения/установки значений по ключам `benedict['x', 'y', 'z']`.
- [Поддержка индекса списка ключей](#) (также отрицательный) с использованием стандартного суффикса `[n]`.
- Создание словаря и конвертация данных из словаря с наиболее распространенными форматами: `base64`, `csv`, `ini`, `json`, `pickle`, `plist`, `query-string`, `toml`, `xls`, `xml`, `yaml`.
- Серверные операции ввода/вывода: файловая система (чтение/запись), `url` (только чтение), `s3` (чтение/запись).
- Содержит [множество служебных методов для изменения/извлечения данных](#) по мере необходимости.

Срок акции с 01.09.2023 по 30.09.2023. Подробнее на [tinkoff.ru](#). АО «Тинькофф Банк», лицензия № 2673.

Установка python-benedict в виртуальное окружение:

```
# создаем виртуальное окружение, если нет
$ python3 -m venv .venv --prompt VirtualEnv
# активируем виртуальное окружение
$ source .venv/bin/activate
# обновляем `pip`
(VirtualEnv):~$ python3 -m pip install -U pip
# можно установить только основной модуль `python-benedict`:
(VirtualEnv):~$ python3 -m pip install python-benedict
# Если необходимо установить все возможности модуля
(VirtualEnv):~$ python3 -m pip install python-benedict[all]
```

В зависимости от того с какими данными предстоит работать (в основном это операции ввода/вывода) можно выбрать зависимости модуля python-benedict, чтобы не тащить все сразу. Обычно хватает базовой установки для работы с популярными данными, такими как `base64`, `csv`, `ini`, `json`, `pickle`...

Ниже представлена иерархия возможных целей установки, доступных при запуске `pip install python-benedict[...]`:

Иерархия зависимостей:

```
- python-benedict[all]
  - python-benedict[io]
    - python-benedict[toml]
    - python-benedict[xls]
    - python-benedict[xml]
    - python-benedict[yaml]
  - python-benedict[s3]
```

Способы создания словаря benedict.

[Модуль python-benedict](#) является подклассом [dict](#), поэтому его можно использовать как обычный словарь. Можете просто преобразовать обычный словарь Python в словарь benedict.

Примеры использования:

```
from benedict import benedict

# Вверх
d = benedict()
```

Дополнительные ресурсы:

[python-benedict](#) - официальная документация

[python-benedict](#) - репозиторий на GitHub

Ссылки:

[python-benedict](#) - официальная документация

[python-benedict](#) - репозиторий на GitHub

Ссылки:

[python-benedict](#) - официальная документация

[python-benedict](#) - репозиторий на GitHub

Ссылки:

[python-benedict](#) - официальная документация

[python-benedict](#) - репозиторий на GitHub

Ссылки:

[python-benedict](#) - официальная документация

[python-benedict](#) - репозиторий на GitHub

Ссылки:

[python-benedict](#) - официальная документация

[python-benedict](#) - репозиторий на GitHub

Ссылки:

[python-benedict](#) - официальная документация

[python-benedict](#) - репозиторий на GitHub

Ссылки:

[python-benedict](#) - официальная документация

[python-benedict](#) - репозиторий на GitHub

Ссылки:

[python-benedict](#) - официальная документация

[python-benedict](#) - репозиторий на GitHub

Ссылки:

[python-benedict](#) - официальная документация

[python-benedict](#) - репозиторий на GitHub

```
# или из существующего словаря Python
d = benedict(existing_dict)

# или создаем из источника данных
# (путь к файлу, URL-адрес или строка данных)
# в поддерживаемом формате:
# Base64, CSV, JSON, TOML, XML, YAML, query-string
d = benedict('https://localhost:8000/data.json', format='json')

# можно использовать в представлении Django
params = benedict(request.GET.items())
page = params.get_int('page', 1)
```

## Добавление/извлечение элементов словаря benedict.

К обычным методам [доступа к значению словаря](#) и [добавления/изменения значения ключа словаря Python](#) добавляются следующие...

## Функциональность KeyAttr при работе со словарем benedict.

Можно получить/установить элементы, используя ключи в качестве атрибутов (точечная нотация).

```
>>> from benedict import benedict
>>> d = benedict()
>>> d.profile.firstname = 'Fabio'
>>> d.profile.lastname = 'Caccamo'
>>> d
# {'profile':{ 'firstname':'Fabio', 'lastname':'Caccamo' } }
```

Предупреждение даже если эта функциональность очень полезна, у нее есть некоторые очевидные ограничения:

- KeyAttr работает только для строковых ключей;
- ключи не могут начинаться с подчеркивания \_
- имена ключей не должны конфликтовать с [именами поддерживаемых методов](#).

## Отключение функции keyattr:

Можно отключить функцию keyattr, передав keyattr\_enabled=False в конструкторе.

```
>>> from benedict import benedict
>>> d = benedict(existing_dict, keyattr_enabled = False)
```

Также можно отключить функцию keyattr, используя свойство:

```
>>> d.keyattr_enabled = False
```

## Функциональность KeyList при работе со словарем benedict.

Везде, где используется ключ, также можно использовать [список](#) или [кортеж](#) ключей.

```
>>> from benedict import benedict
>>> d = benedict()

# установить значения по списку ключей
>>> d['profile', 'firstname'] = 'Fabio'
>>> d['profile', 'lastname'] = 'Caccamo'
>>> d
# { 'profile':{ 'firstname':'Fabio', 'lastname':'Caccamo' } }
>>> d['profile']
# { 'firstname':'Fabio', 'lastname':'Caccamo' }

# проверка существования пути к ключу
>>> ['profile', 'lastname'] in d
# True

# удалить значение по списку ключей
>>> del d['profile', 'lastname']
>>> d['profile']
# {'firsttname':'Fabio' }
```

Вверх

# Функциональность KeyPath при работе со словарем benedict.

Разделителем пути по умолчанию является точка .

```
>>> from benedict import benedict
>>> d = benedict()

# установить значения по пути
>>> d['profile.firstname'] = 'Fabio'
>>> d['profile.lastname'] = 'Caccamo'
>>> d
# { 'profile':{ 'firstname':'Fabio', 'lastname':'Caccamo' } }
>>> d['profile']
# { 'firstname':'Fabio', 'lastname':'Caccamo' }

# проверка существования пути к ключу
>>> 'profile.lastname' in d
# True

# удаление значения по пути
>>> del d['profile.lastname']
```

Точка . в качестве разделителя пути к ключам конечно удобно, но это обстоятельство накладывает некоторые ограничения. Например, если обычный словарь Python имеет ключ, в названии которого содержится точка, например 'awesome.com', то при преобразовании его в benedict будет появляться ошибка.

```
data = {
    'awesome.com': 'https://awesome.com',
    'skills': {
        'programming': {
            'Python': '5 stars',
            'JavaScript': '4 stars',
        }
    }
}

>>> data = benedict(data)
# Traceback (most recent call last):
# ...
# ValueError: Key should not contain keypath separator '.', found: 'awesome.com'.
```

Чтобы обойти это ограничение, benedict предоставляет параметр keypath\_separator для указания другого разделителя.

## Изменение разделителя пути к ключу.

Можно настроить разделитель 'путей к ключу', передав аргумент keypath\_separator в конструкторе.

```
data = benedict(data, keypath_separator='/')
# теперь можно получить доступ к фактическому ключу
print(data['awesome.com'])
# https://awesome.com

# а так же обратиться, используя разделитель пути '/'
print(data['skills/programming/Python'])
# 5 stars
```

Также можно изменить keypath\_separator в любое время, используя свойство:

```
data.keypath_separator = '|'
```

Если какой-либо существующий ключ словаря содержит значение keypath\_separator, то будет возбуждено исключение. Например, словарь содержит ключ 'awesome.com'

```
# попробуем изменить разделитель пути снова на точку
data.keypath_separator = '.'
# Traceback (most recent call last):
# ...
# ValueError: Key should not contain keypath separator '.', found: 'awesome.com'.
```

Вверх

## Отключение функции keypath.

Можно отключить функцию `keypath`, передав в конструкторе `keypath_separator=None`.

```
d = benedict(existing_dict, keypath_separator=None)
```

Также можно отключить функцию `keypath`, используя свойство:

```
d.keypath_separator = None
```

## Поддержка индексации ключей.

'Пути к ключам' могут включать индексы (также отрицательные) с использованием `[n]` для очень быстрого выполнения любой операции:

Например, получим координаты последнего местоположения первого результата:

```
loc = d['results[0].locations[-1].coordinates']
lat = loc.get_decimal('latitude')
lng = loc.get_decimal('longitude')
```

## Пример обращения к элементам словаря benedict.

```
from benedict import benedict

data = {
    'name': 'Josh',
    'occupation': 'Data scientist',
    'skills': {
        'programming': {
            'Python': '5 stars',
            'JavaScript': '4 stars',
            'Java': '3 stars'
        },
        'frameworks': ['Django', 'React', 'Spring']
    }
}

# инициализируем словарь
data = benedict(data)

# проверка существования пути
if 'skills.programming.Java' in data:
    print(data['skills.programming.Java'])
    # '3 stars'

    # или эквивалентно
    print(data['skills', 'programming', 'Java'])
    # '3 stars'

# путь к несуществующему ключу
print('skills.softskills.managing' in data)
# False

# можно также использовать `.get(...)`
print(data.get('skills.softskills.managing'))
# None

# можно добавлять пути
data['skills.softskills.managing'] = '5 stars'
print(data['skills', 'softskills', 'managing'])
# '5 stars'

# вывод словаря в читаемом виде
print(data.dump())
# {
#     'name': 'Josh',
#     'occupation': 'Data scientist',
#     'skills': {
#         'frameworks': [
#             'Django',
#             'React',
```

```
#         'Spring'
#     ],
#     'programming': {
#         'Java': '3 stars',
#         'JavaScript': '4 stars',
#         'Python': '5 stars'
#     },
#     'softskills': {
#         'managing': '5 stars'
#     }
# }
```

Содержание раздела:

- [КРАТКИЙ ОБЗОР МАТЕРИАЛА.](#)
- [Модификация и извлечение данных из словаря python-benedict](#)
- [Загрузка/выгрузка данных в/из словаря python-benedict в различные форматы](#)
- [Приведение данных к определенному типу в словаре python-benedict](#)