


Модуль copy в Python, глубокое копирование объектов



mango-office.ru

Виртуальная АТС Расширенная

2 000 ₽

Узнать больше

РЕКЛАМА

[Стандартная библиотека Python3.](#) / Модуль copy в Python, глубокое копирование объектов

Операции неглубокого и глубокого копирования объектов

Операторы присваивания в Python не копируют объекты, они создают привязки между целью и объектом. Для коллекций, которые являются изменяемыми или содержат изменяемые элементы, иногда требуется копия, чтобы можно было изменить одну копию, не изменяя другую. [Модуль copy](#) обеспечивает общие операции неглубокого и глубокого копирования.

Краткое описание интерфейса.

copy.copy(x):

Функция `copy.copy(x)` возвращает мелкую копию `x`.

copy.deepcopy(x[, memo]):

Функция `copy.deepcopy()` возвращает глубокую копию `x`.

copy.error:

Исключение `copy.error` возбуждается для специфических ошибок модуля.

Описание модуля copy.

Разница между мелким и глубоким копированием актуальна только для составных объектов, содержащих другие объекты, например [списки](#) или [экземпляры классов](#):

- Неглубокая копия создает новый составной объект, а затем (насколько это возможно) вставляет в него ссылки на объекты, найденные в оригинале.
- Глубокая копия создает новый составной объект, а затем рекурсивно вставляет в него копии объектов, найденных в оригинале.

При операциях глубокого копирования часто возникают две проблемы, которых нет при операциях поверхностного копирования:

- Рекурсивные объекты (составные объекты, которые прямо или косвенно содержат ссылку на самих себя) могут вызвать рекурсивный цикл.
- Поскольку глубокая копия копирует все, она может копировать слишком много, например данные, которые предназначены для совместного использования между копиями.

Функция [copy.deepcopy\(\)](#) позволяет избежать этих проблем:

- ведет мемо-словарь объектов, уже скопированных во время текущего прохода копирования;
- позволяет пользовательским классам переопределять операцию копирования или набор копируемых компонентов.

[Модуль copy](#) не копирует такие типы, как модуль, метод, трассировка стека, кадр стека, файл, сокет, окно, массив или любые подобные типы. Он "копирует" функции и классы (неглубоко и глубоко), возвращая исходный объект без изменений, что совместимо с тем, как они обрабатываются [модулем pickle](#).

Неглубокие копии словарей можно сделать с помощью [метода dict.copy\(\)](#), а списков - назначив срез всего списка, например `copied_list = original_list[:]`.

Классы могут использовать те же интерфейсы для управления копированием, которые они используют для управления процессом pickling. Смотрите описание [модуля pickle](#) для получения информации об этих методах. Фактически, модуль копирования использует зарегистрированные функции pickle из [модуля copyreg](#).

Чтобы класс мог определить свою собственную реализацию копии, он может определять специальные методы `__copy__()` и `__deepcopy__()`.

- Первый вызывается для реализации операции поверхностного копирования, никаких дополнительных аргументов не передается.
- Последний вызывается для реализации операции глубокого копирования, передается один аргумент, мемо-словарь.

Если реализации `__deepcopy__()` необходимо сделать глубокую копию компонента, она должна вызвать функцию `copy.deepcopy()` с компонентом в качестве первого аргумента и мемо-словарем в качестве второго аргумента.

ХОЧУ ПОМОЧЬ
ПРОЕКТУ



[DOCS-Python.ru](https://docs-python.ru)[™], 2023 г.

(Внимание! При копировании материала ссылка на источник обязательна)

[@docs_python_ru](https://docs-python.ru)