

9. Копирование списков.

Модуль `copy`

Операция присваивания не копирует объект, он лишь создаёт ссылку на объект. Для изменяемых коллекций, или для коллекций, содержащих изменяемые элементы, часто необходима такая копия, чтобы её можно было изменить, не изменяя оригинал. Данный модуль предоставляет общие (поверхностная и глубокая) операции копирования.

`copy.copy(x)` - возвращает поверхностную копию `x`.

`copy.deepcopy(x)` - возвращает полную копию `x`.

Разница между поверхностным и глубоким копированием существенна только для составных объектов, содержащих изменяемые объекты (например, список списков, или словарь, в качестве значений которого - списки или словари):

- **Поверхностная копия** создает новый составной объект, и затем (по мере возможности) вставляет в него ссылки на объекты, находящиеся в оригинале.
- **Глубокая копия** создает новый составной объект, и затем рекурсивно вставляет в него копии объектов, находящихся в оригинале.

Для операции глубокого копирования часто возникают две проблемы, которых нет у операции поверхностного копирования:

- Рекурсивные объекты (составные объекты, которые явно или неявно содержат ссылки на себя) могут стать причиной рекурсивного цикла;
- Поскольку глубокая копия копирует всё, она может скопировать слишком много, например, административные структуры данных, которые должны быть разделяемы даже между копиями.

Функция `deepcopy` решает эти проблемы путем:

- Хранения "мето" словаря объектов, скопированных во время текущего прохода копирования;
- Позволения классам, определенным пользователем, переопределять операцию копирования или набор копируемых компонентов.

Поверхностная копия изменяемых объектов также может быть создана методом `.copy()` у `list`

Также поверхностная копия может быть осуществлена с помощью среза:

```
x = y[:]
```

Понимание разницы между поверхностной и глубокой копией


Поверхностная:

```
Users > georgiydemo > Desktop > check.py > ...
1  import copy
2
3  TEST1 = [[1,2,3], [1,2,3]]
4  TEST2 = copy.copy(TEST1)
5
6  print("TEST1", TEST1)
7  print("TEST2", TEST2)
8
9  TEST2[1].append(4)
10
11 print("TEST1", TEST1)
12 print("TEST2", TEST2)
13
```

TERMINAL 1: Python

```
[~]
[~] /usr/local/opt/python/bin/python3.7 /Users/georgiydemo/Desktop/check.py
TEST1 [[1, 2, 3], [1, 2, 3]]
TEST2 [[1, 2, 3], [1, 2, 3]]
TEST1 [[1, 2, 3], [1, 2, 3, 4]]
TEST2 [[1, 2, 3], [1, 2, 3, 4]]
[~]
```

Глубокая:

Users > georgiydemo > Desktop >  check.py > ...

```
1  import copy
2
3  TEST1 = [[1,2,3], [1,2,3]]
4  TEST2 = copy.deepcopy(TEST1)
5
6  print("TEST1", TEST1)
7  print("TEST2", TEST2)
8
9  TEST2[1].append(4)
10
11 print("TEST1", TEST1)
12 print("TEST2", TEST2)
13
```

TERMINAL

...

1: Python

▼

+

□

🗑

^

×

```
[~]
[~] /usr/local/opt/python/bin/python3.7 /Users/georgiydemo/Desktop/check.py
TEST1 [[1, 2, 3], [1, 2, 3]]
TEST2 [[1, 2, 3], [1, 2, 3]]
TEST1 [[1, 2, 3], [1, 2, 3]]
TEST2 [[1, 2, 3], [1, 2, 3, 4]]
```

```
[~]
[~]
```