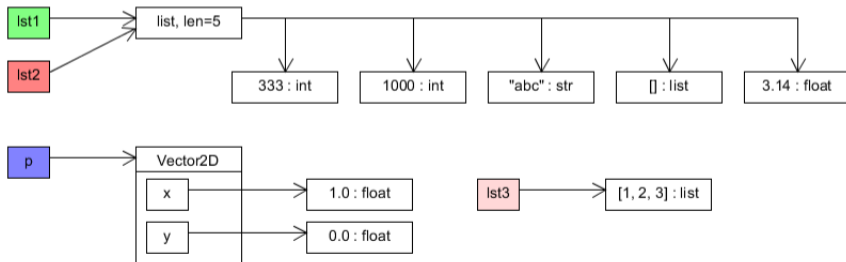


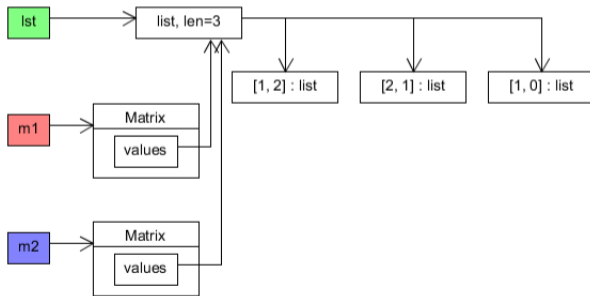
Diagram obiektów (uproszczony):

```
lst1 = [333, 1000, "abc", [], 3.14]  
lst2 = lst1  
lst3 = [1, 2, 3]  
p = Vector2D(x=1.0, y=0.0)
```



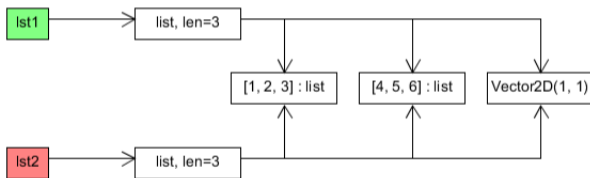
Głębokie kopie

```
lst = [[1, 2], [2, 1], [1, 0]]  
m1 = Matrix(lst)  
m2 = Matrix(lst)  
lst[0] = ... # dotyczy wielu obiektów
```



Kopia płytka: kopiuje obiekt, ale nie zawartość (np. obiekty nazywane przez atrybuty).

```
lst1 = [[1, 2, 3], [4, 5, 6], Vector2D(1, 1)]  
lst2 = lst1.copy() # lst1[:], list(lst1), copy.copy(lst1)
```

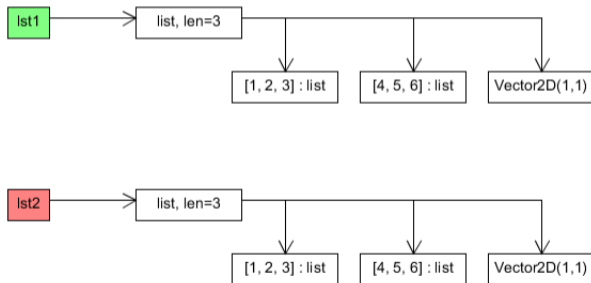


Funkcja `copy` z modułu `copy` tworzy płytką kopię obiektu.

Głębokie kopie

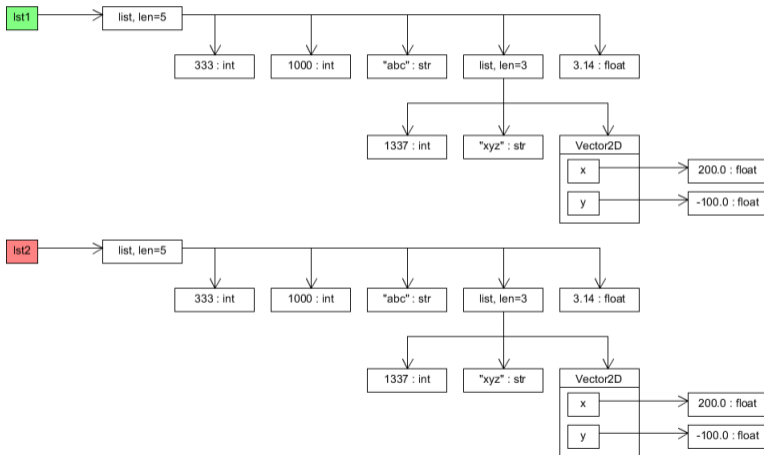
Kopia głęboka: rekurencyjnie kopiuje obiekt i wykonuje głębokie kopie zawartości.

```
lst1 = [[1, 2, 3], [4, 5, 6], Vector2D(1, 1)]  
lst2 = copy.deepcopy(lst1) # import copy
```



Głębokie kopie

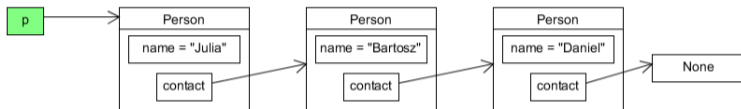
Głębokie kopiowanie rekurencyjnie tworzy pełną kopię całej struktury powiązanej z obiektem.



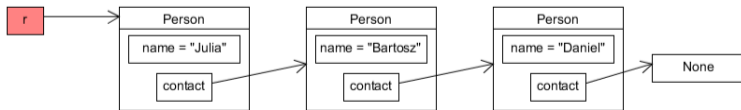
Głębokie kopie

Przykład: kontakt ICE.

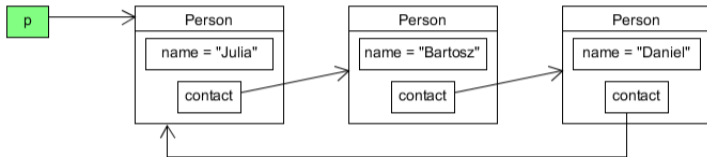
Klasa Person zawierająca imię osoby (name) oraz atrybut contact oznaczający jej kontakt alarmowy (również instancję Person) lub None.



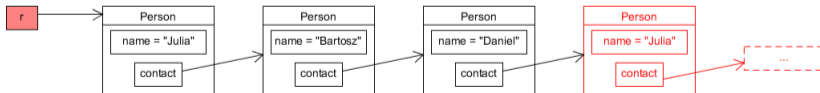
```
r = copy.deepcopy(p)
```



Problem przy rekurencyjnym kopiowaniu: samoodniesienia.



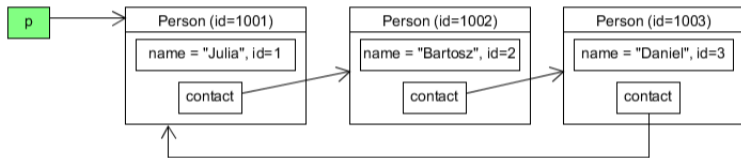
```
r = copy.deepcopy(p)
```

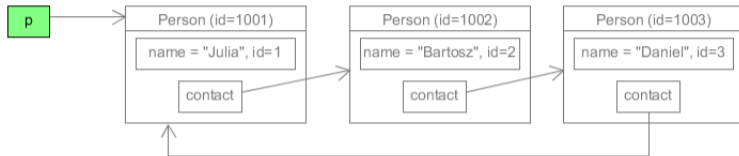


Głębokie kopie

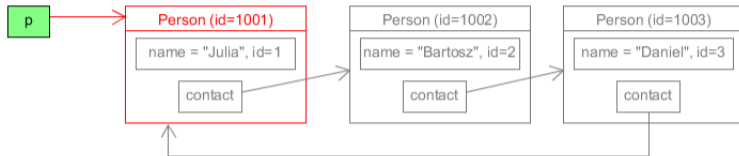
Rozwiązanie: algorytm wykonujący głęboką kopię zapamiętuje identyfikatory skopiowanych już obiektów, stwarzając je z ich kopiami.

Przykładowe id obiektów z przykładu:



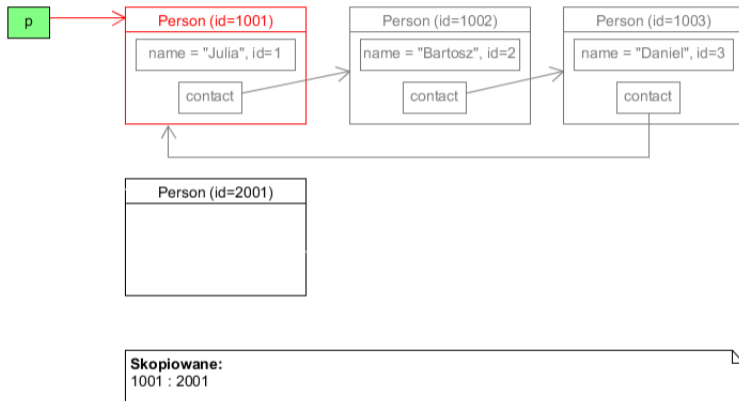


Skopiowane:

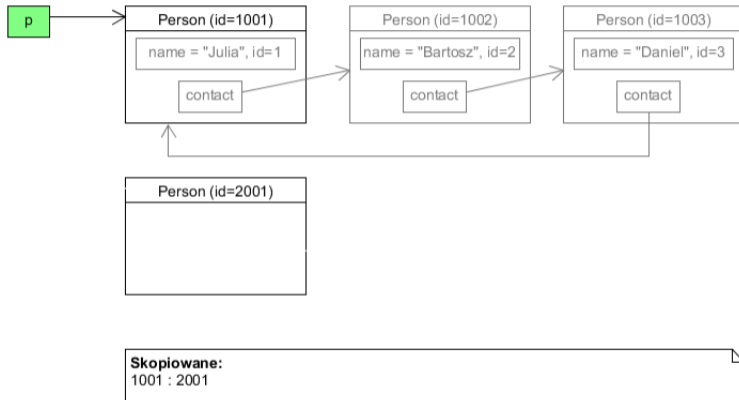


Skopiowane:

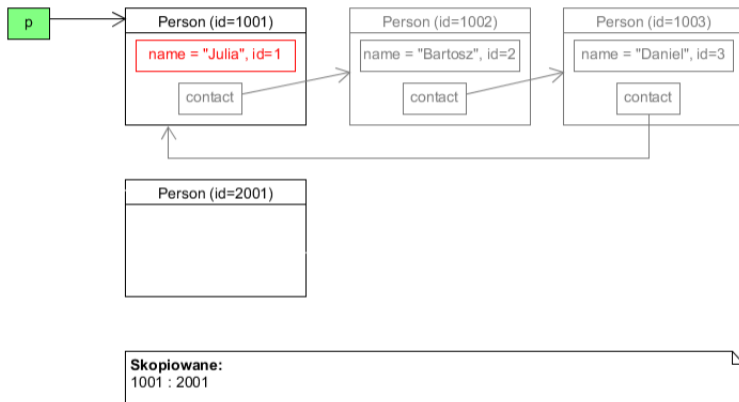
Głębokie kopie



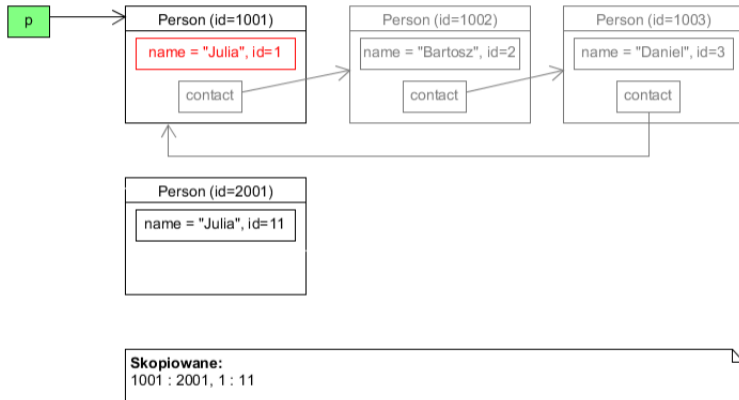
Głębokie kopie

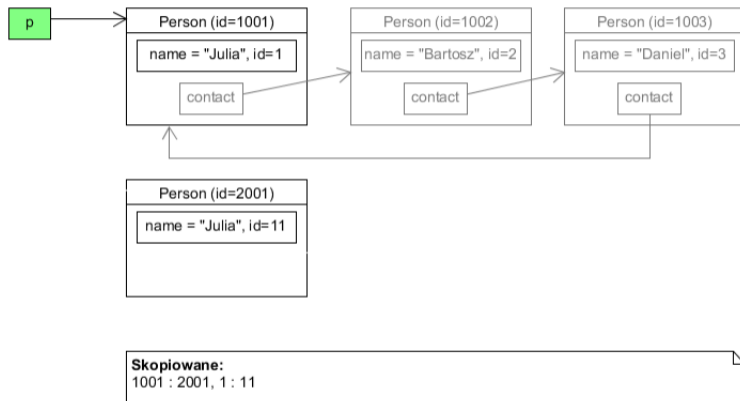


Głębokie kopie

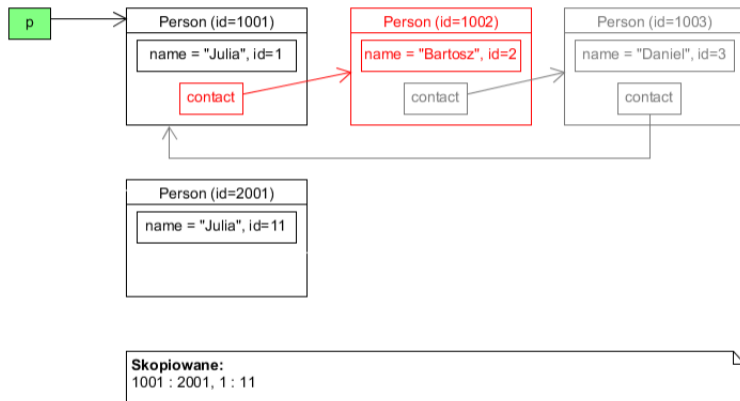


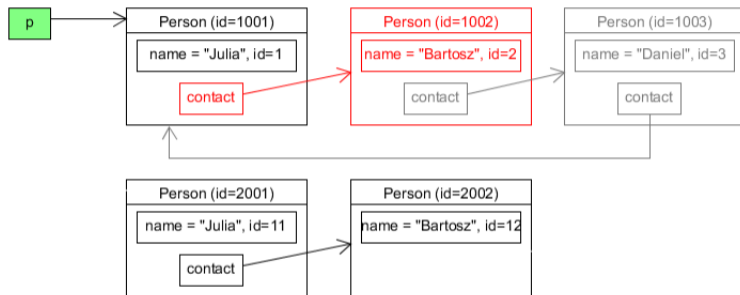
Głębokie kopie





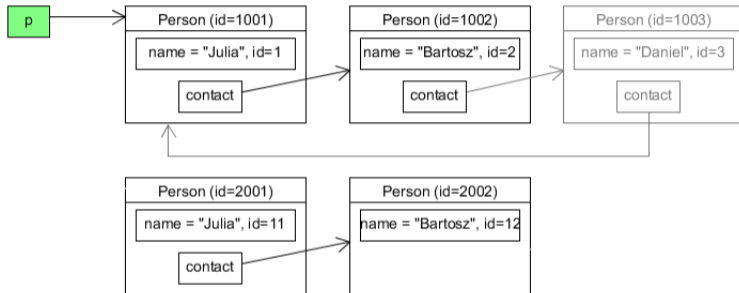
Głębokie kopie





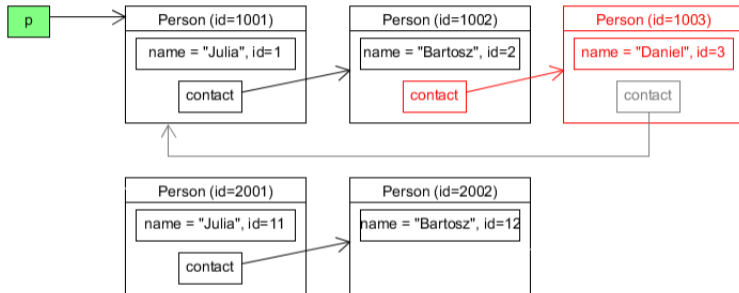
Skopiowane:

1001 : 2001, 1 : 11, 1002 : 2002, 2 : 12



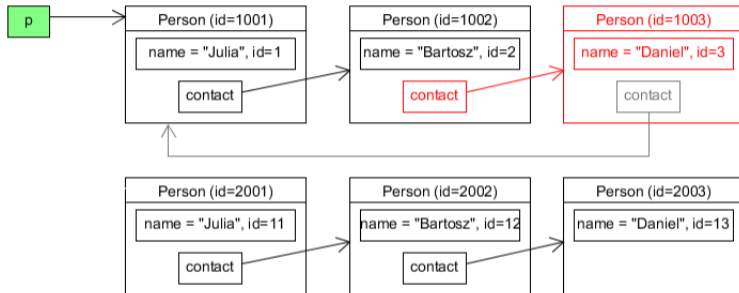
Skopiowane:

1001 : 2001, 1 : 11, 1002 : 2002, 2 : 12



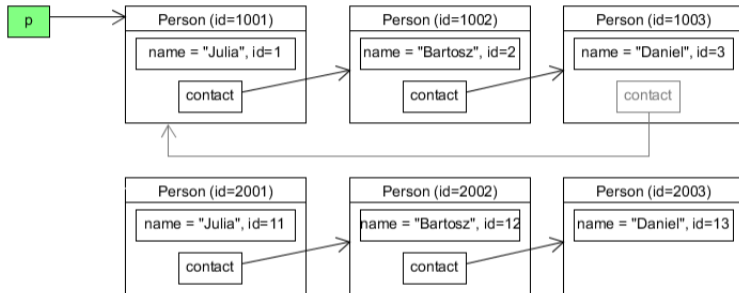
Skopiowane:

1001 : 2001, 1 : 11, 1002 : 2002, 2 : 12



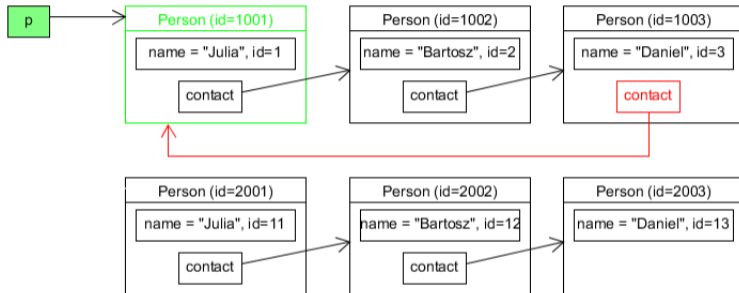
Skopiowane:

1001 : 2001, 1 : 11, 1002 : 2002, 2 : 12, 1003 : 2003, 3 : 13



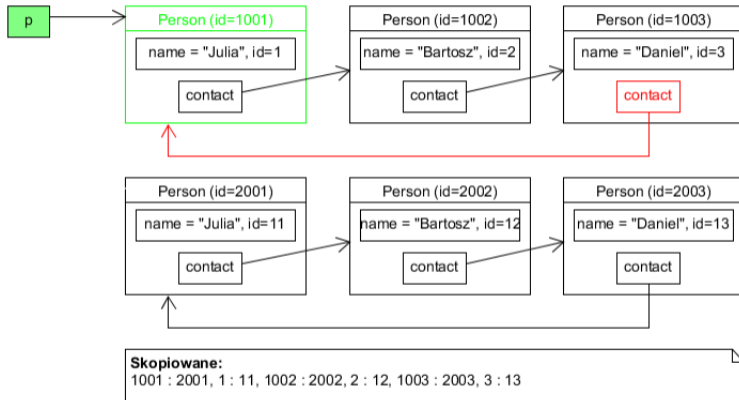
Skopiowane:

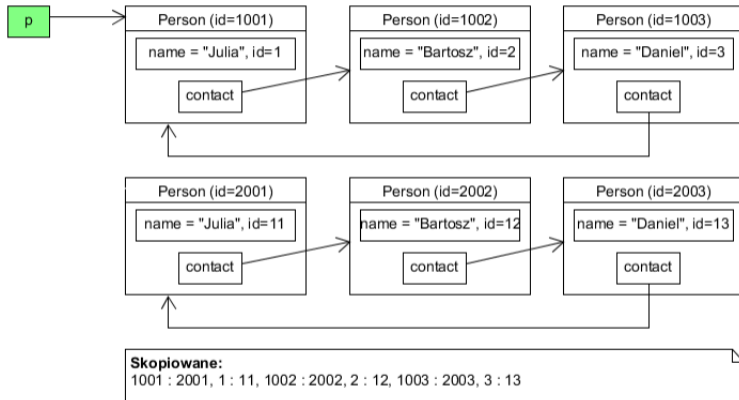
1001 : 2001, 1 : 11, 1002 : 2002, 2 : 12, 1003 : 2003, 3 : 13

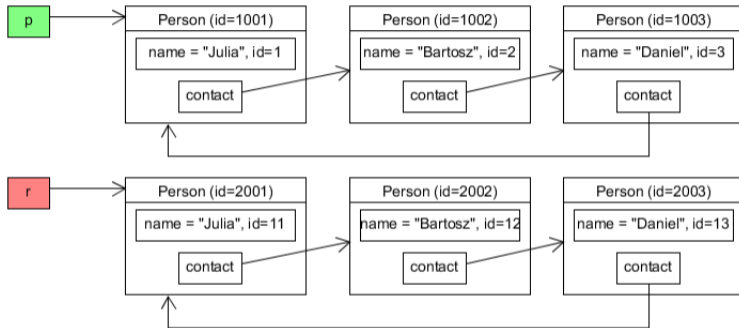


Skopiowane:

1001 : 2001, 1 : 11, 1002 : 2002, 2 : 12, 1003 : 2003, 3 : 13



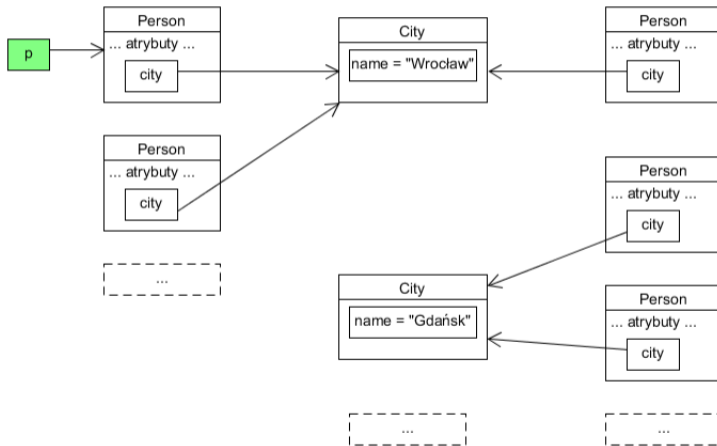




Głębokie kopie

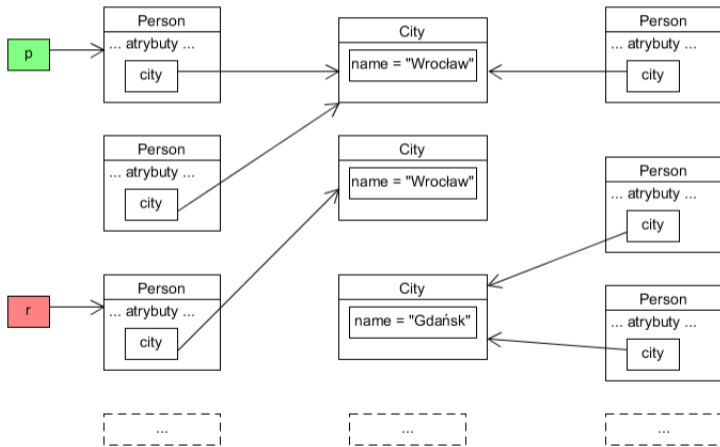
A co, jeśli nie chcemy pełnej kopii struktury?

Przykład:



A co, jeśli nie chcemy pełnej kopii struktury?

```
r = copy.deepcopy(p)
```



Głębokie kopie

Specjalna metoda `__deepcopy__` pozwala na zdefiniowanie sposobu, w jak `copy.deepcopy` ma wykonać głęboką kopię obiektu.

```
def __deepcopy__(self, memo):  
    # ...
```

`memo` - słownik pamiętający skopiowane już obiekty (w parach (id, obiekt-kopia)).
Metoda zwraca głęboką kopię obiektu.

Przykładowe rozwiązanie* problemu: zdefiniowanie `__deepcopy__` dla klasy `City` tak, żeby zwracała oryginał i nie tworzyła kopii.

```
class City:  
    # ...  
    def __deepcopy__(self, memo):  
        return self
```

(podobnie: metoda `__copy__` dla zwykłej kopii)

* - niekoniecznie optymalne dla tego problemu