

Объекты и переменные

Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память

Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память

Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память



Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память

set()

Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память

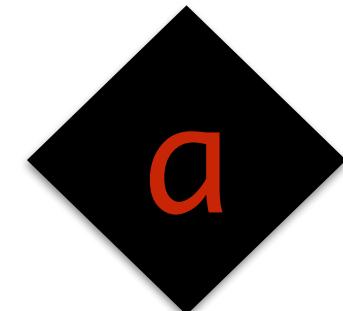
set()

Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память

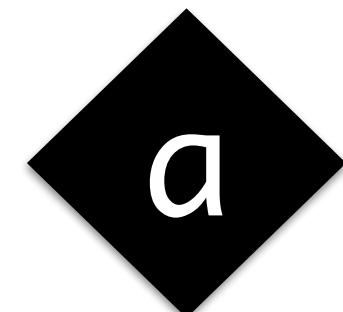


Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память



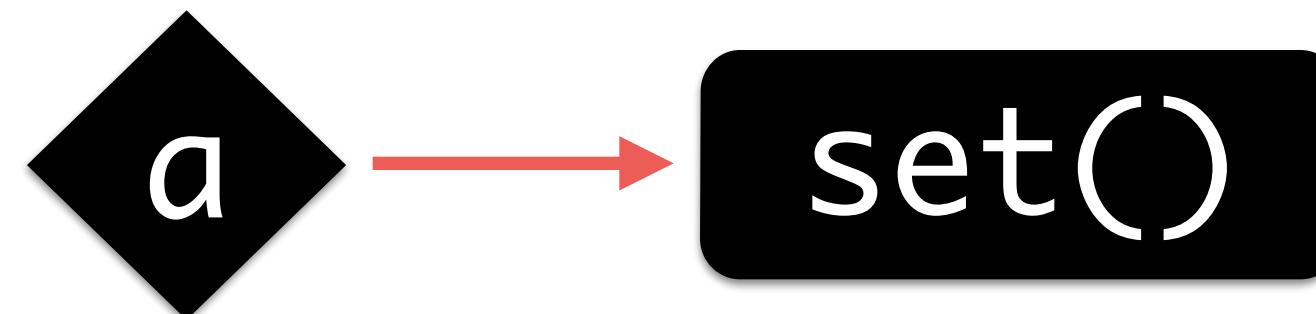
set()

Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память

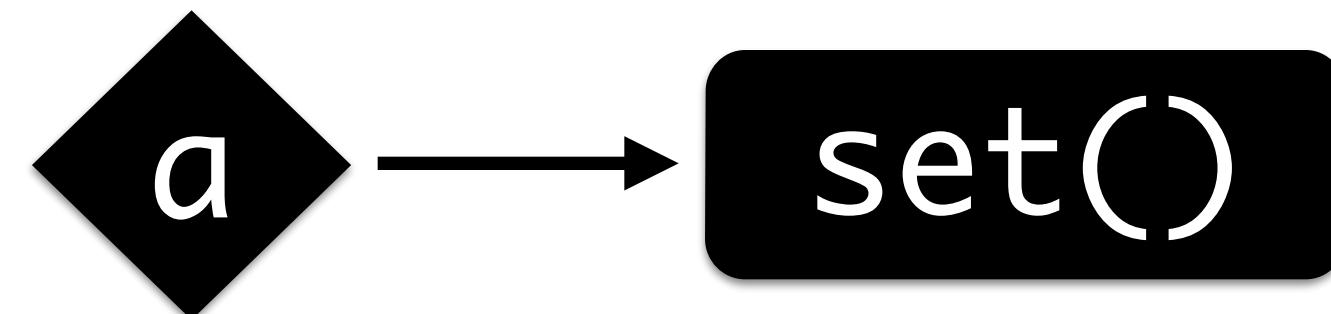


Объекты и переменные

Исходный код

```
a = set()  
b = a  
c = a
```

Память



Объекты и переменные

Исходный код

```
a = set()  
b = a  
c = a
```

Память



Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память



Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память



Объекты и переменные

Исходный код

```
a = set()  
b = a  
d = a
```

Память



Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

1

2

3

Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

1

2

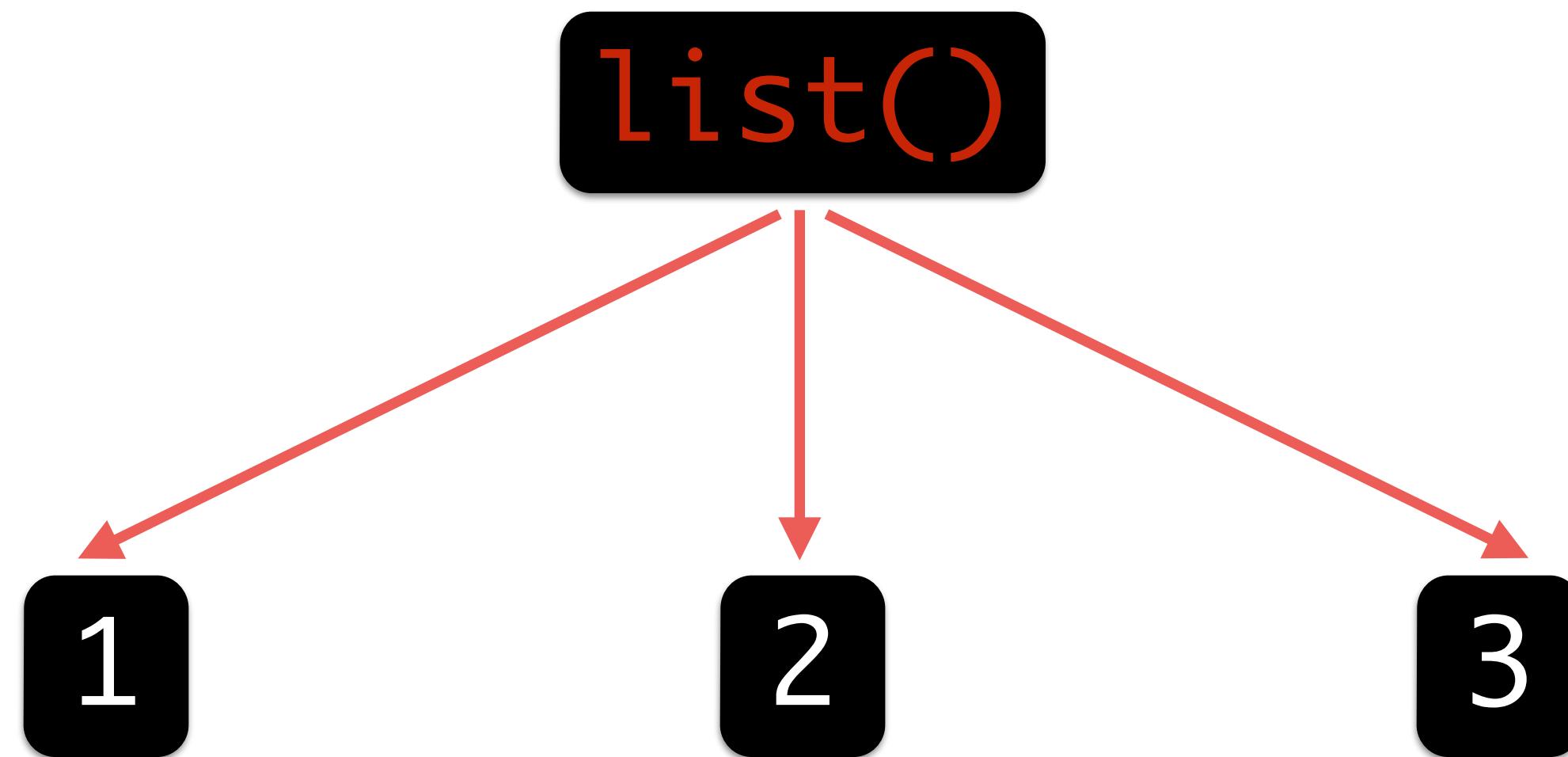
3

Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

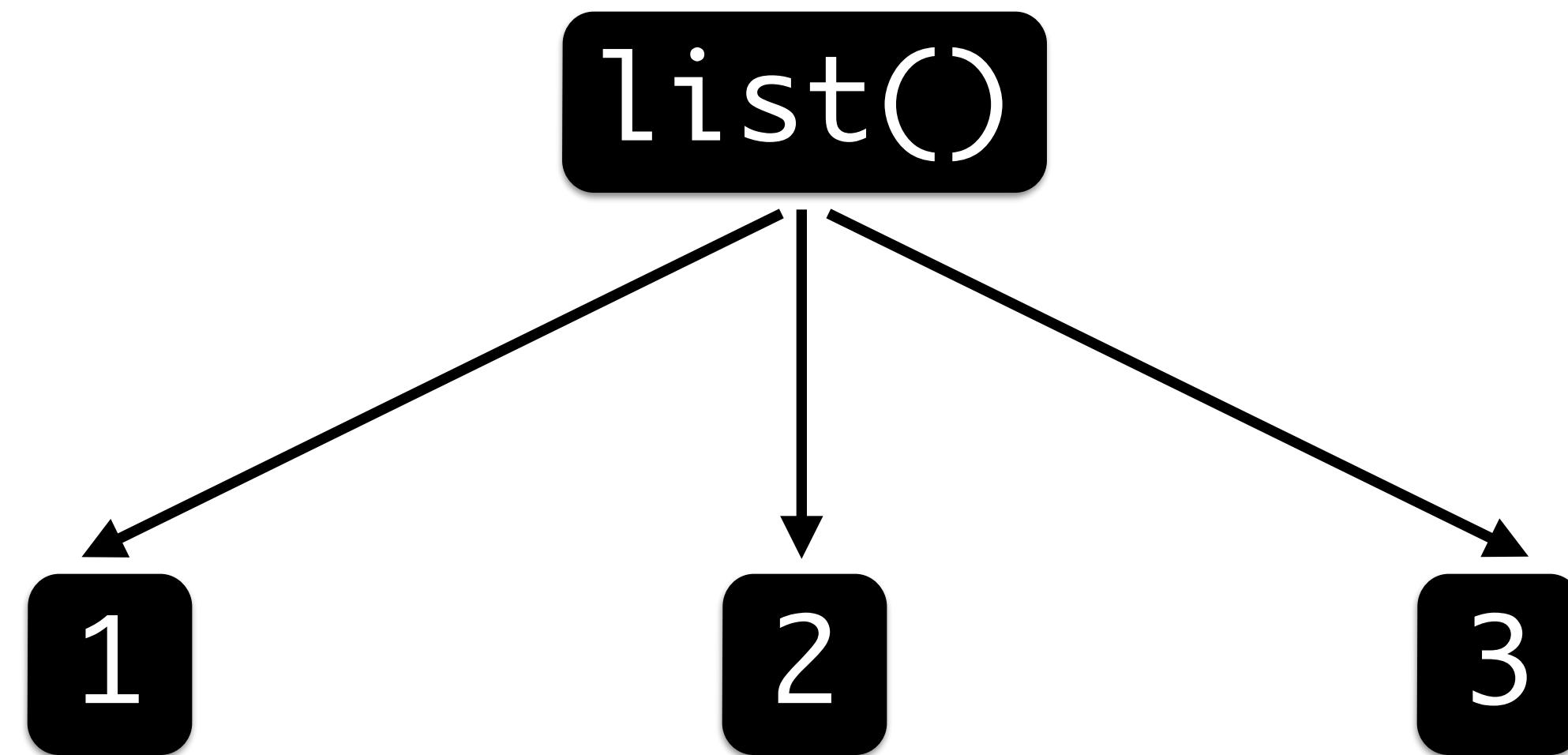


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

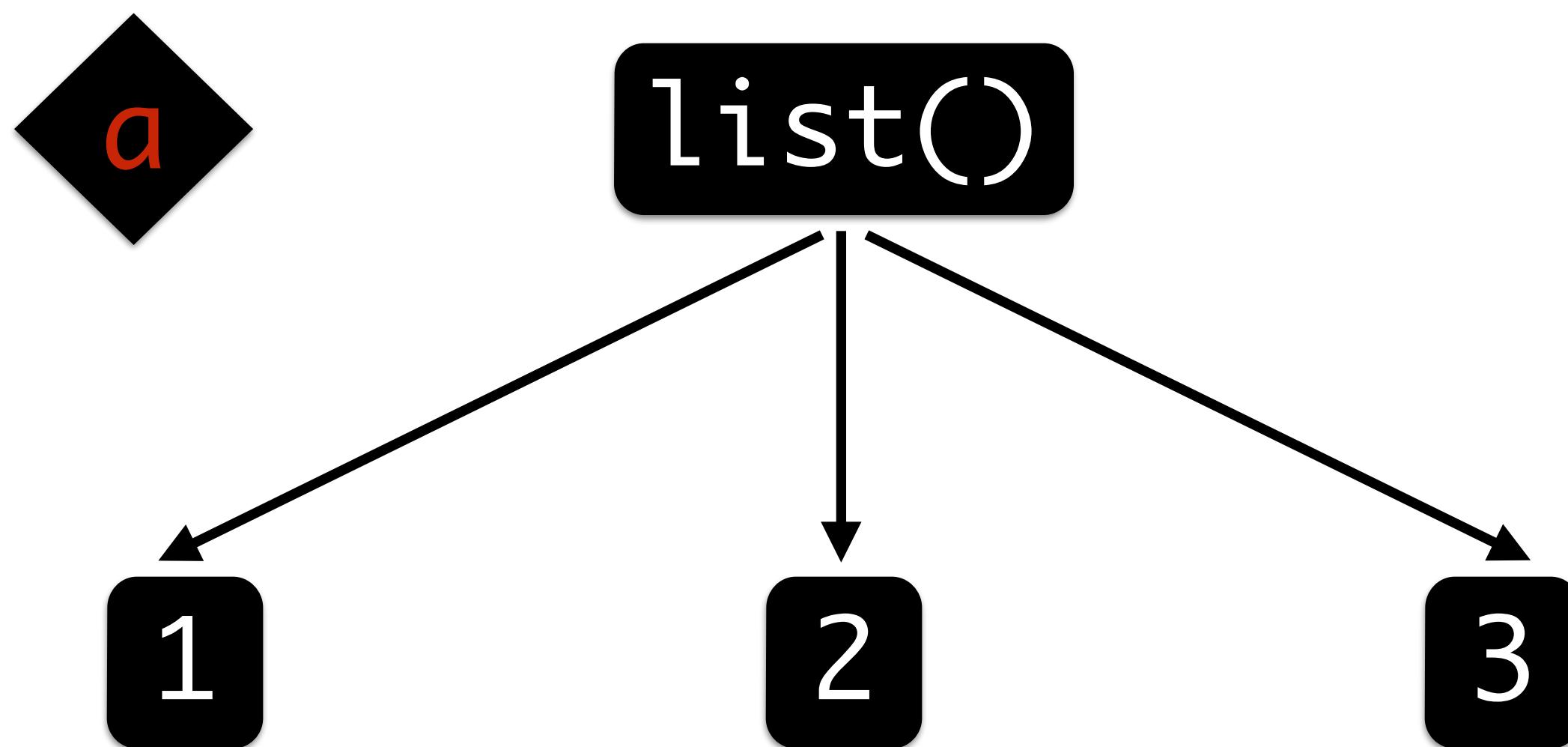


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

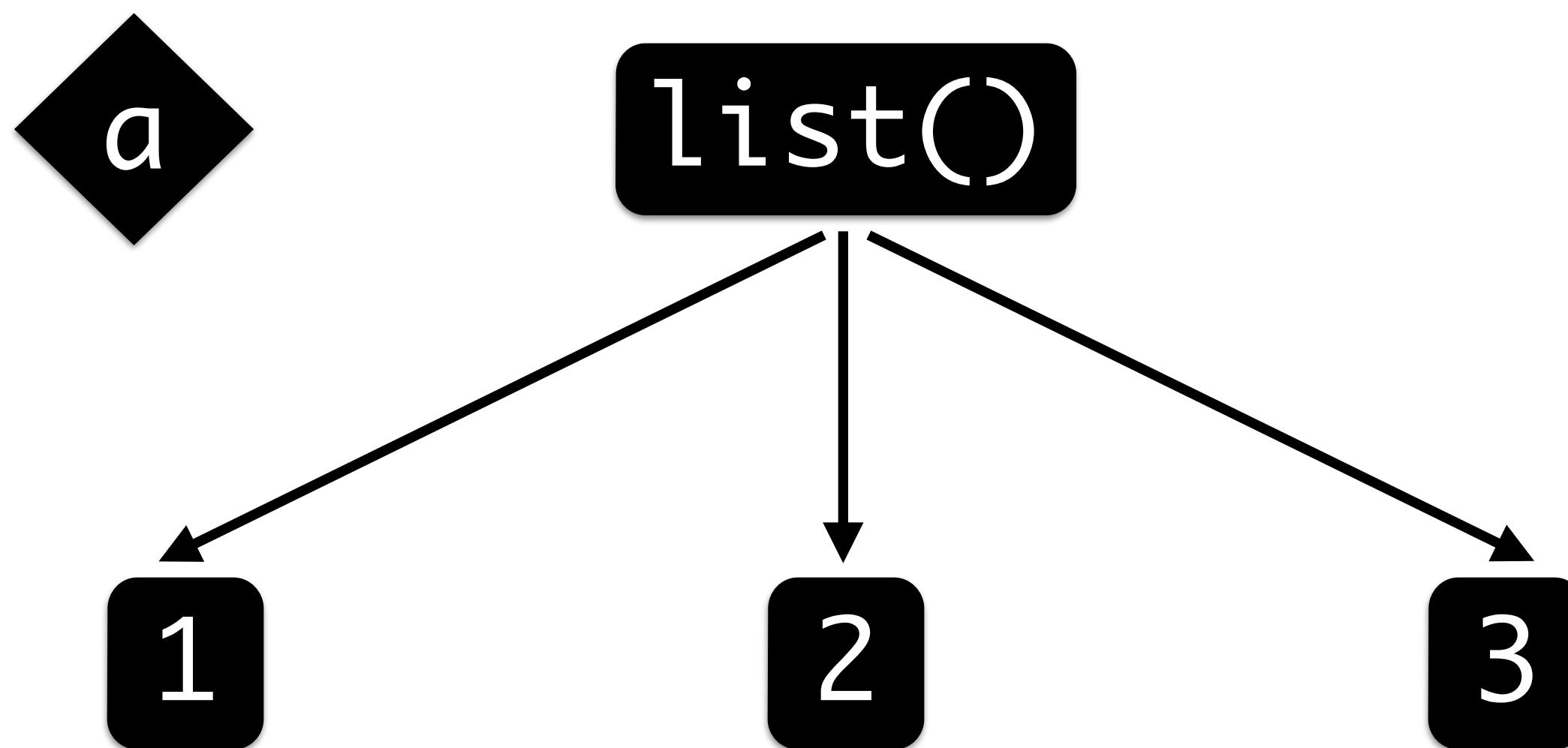


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

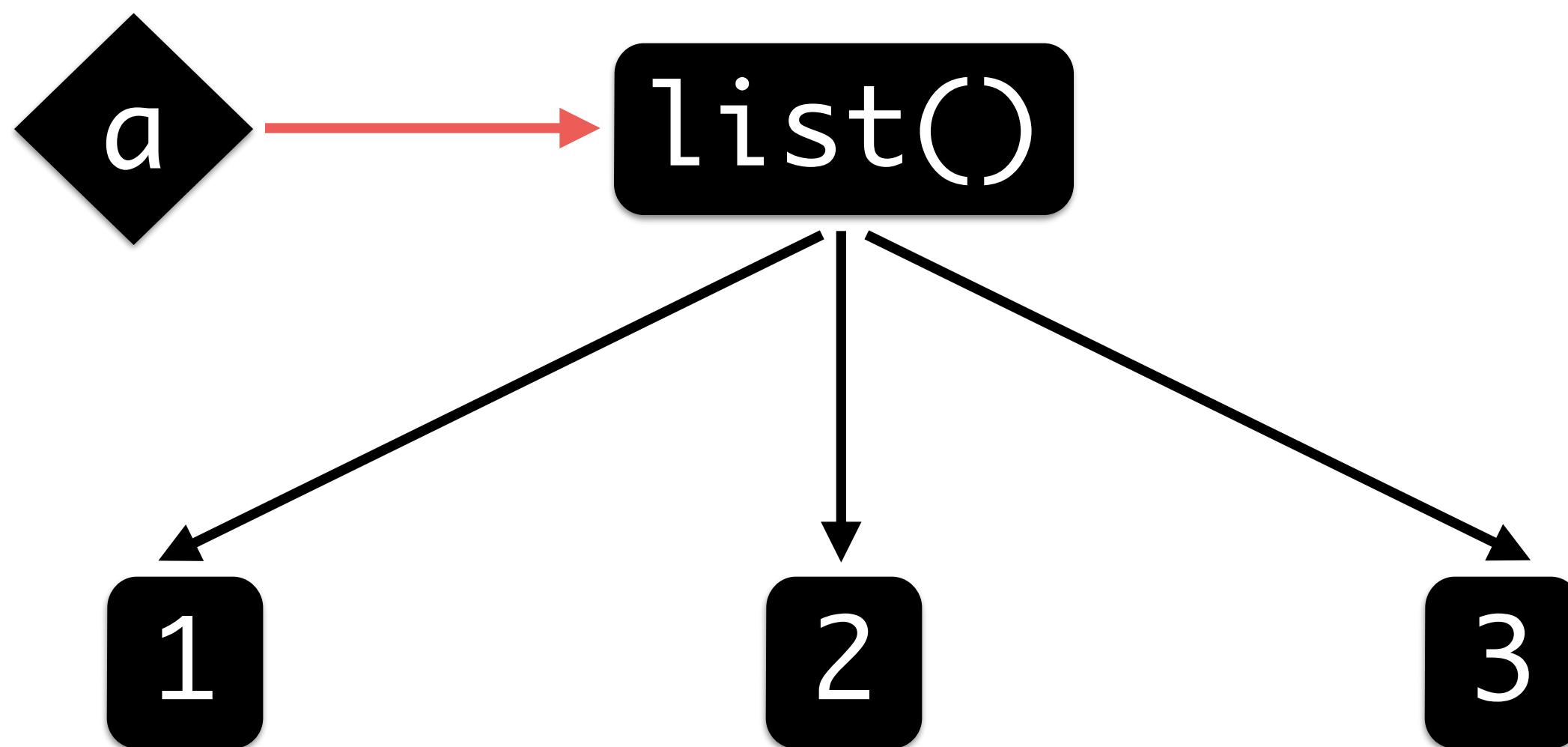


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

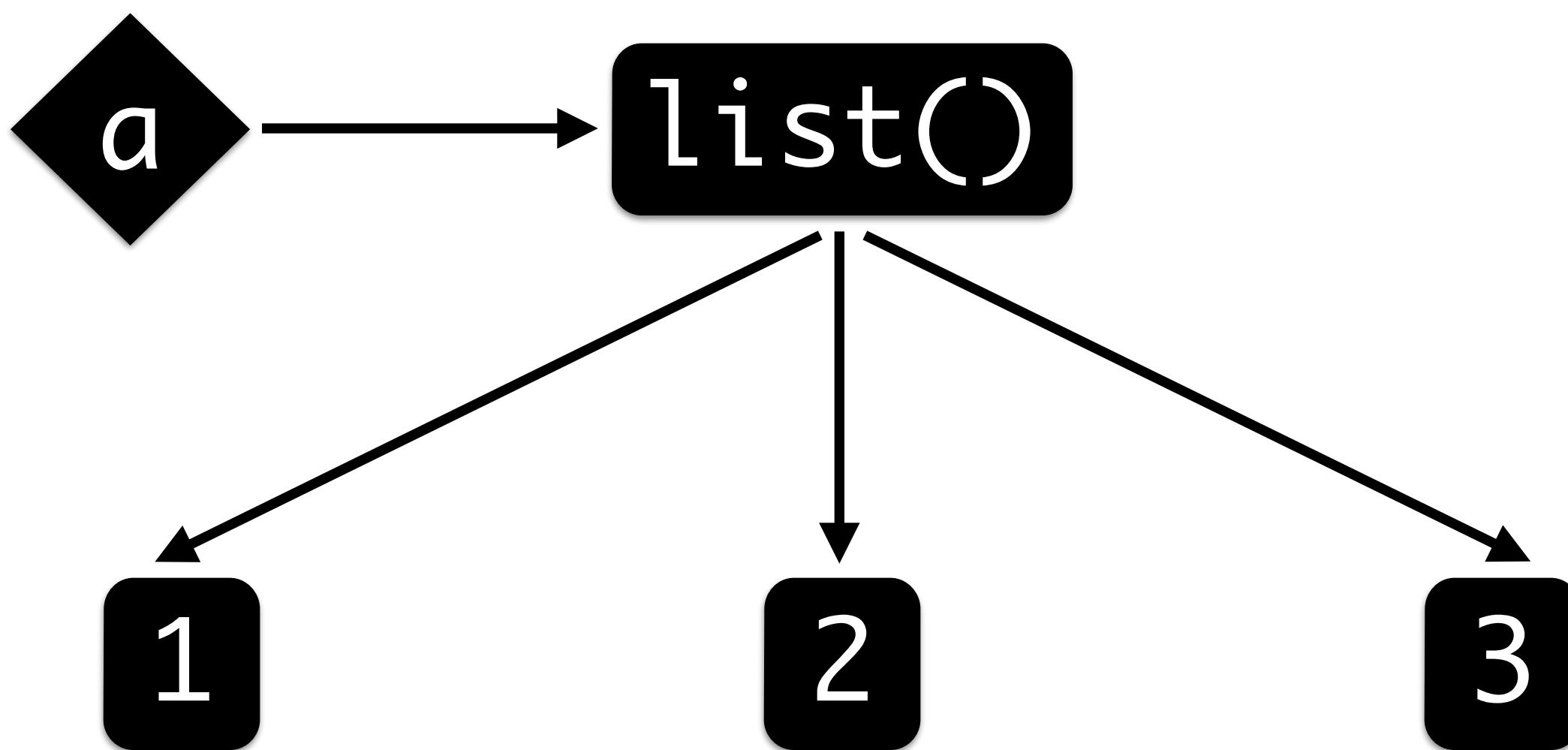


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

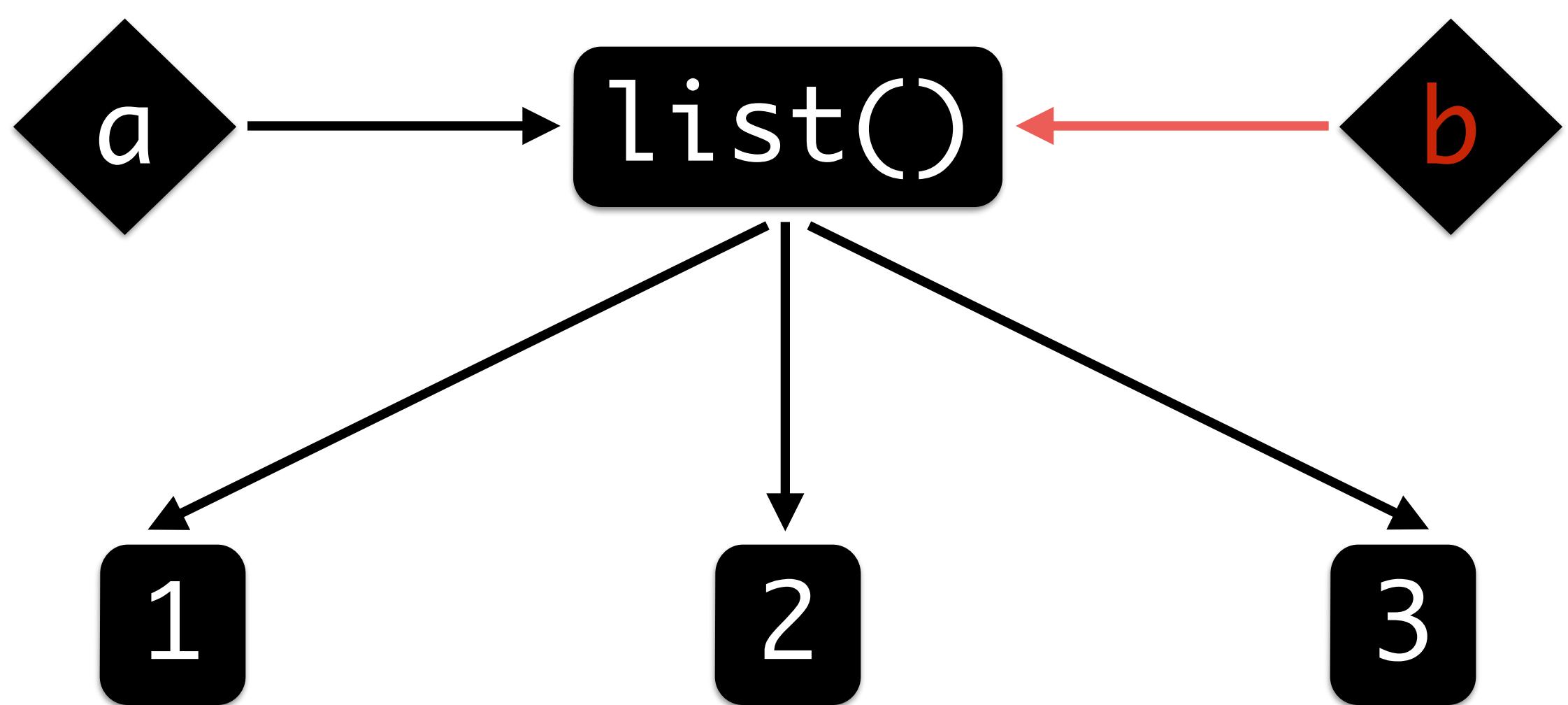


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
буғынғ(р)
```

Память

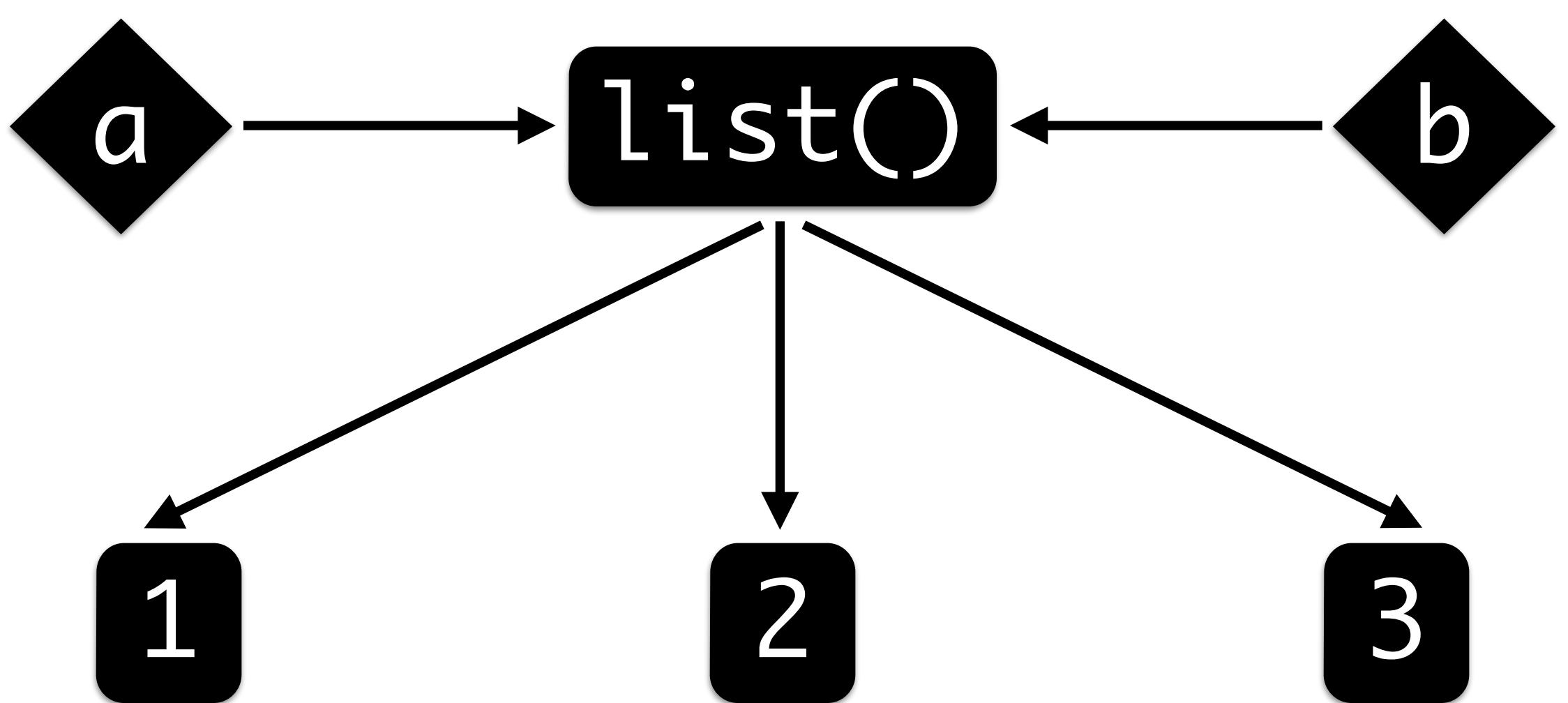


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

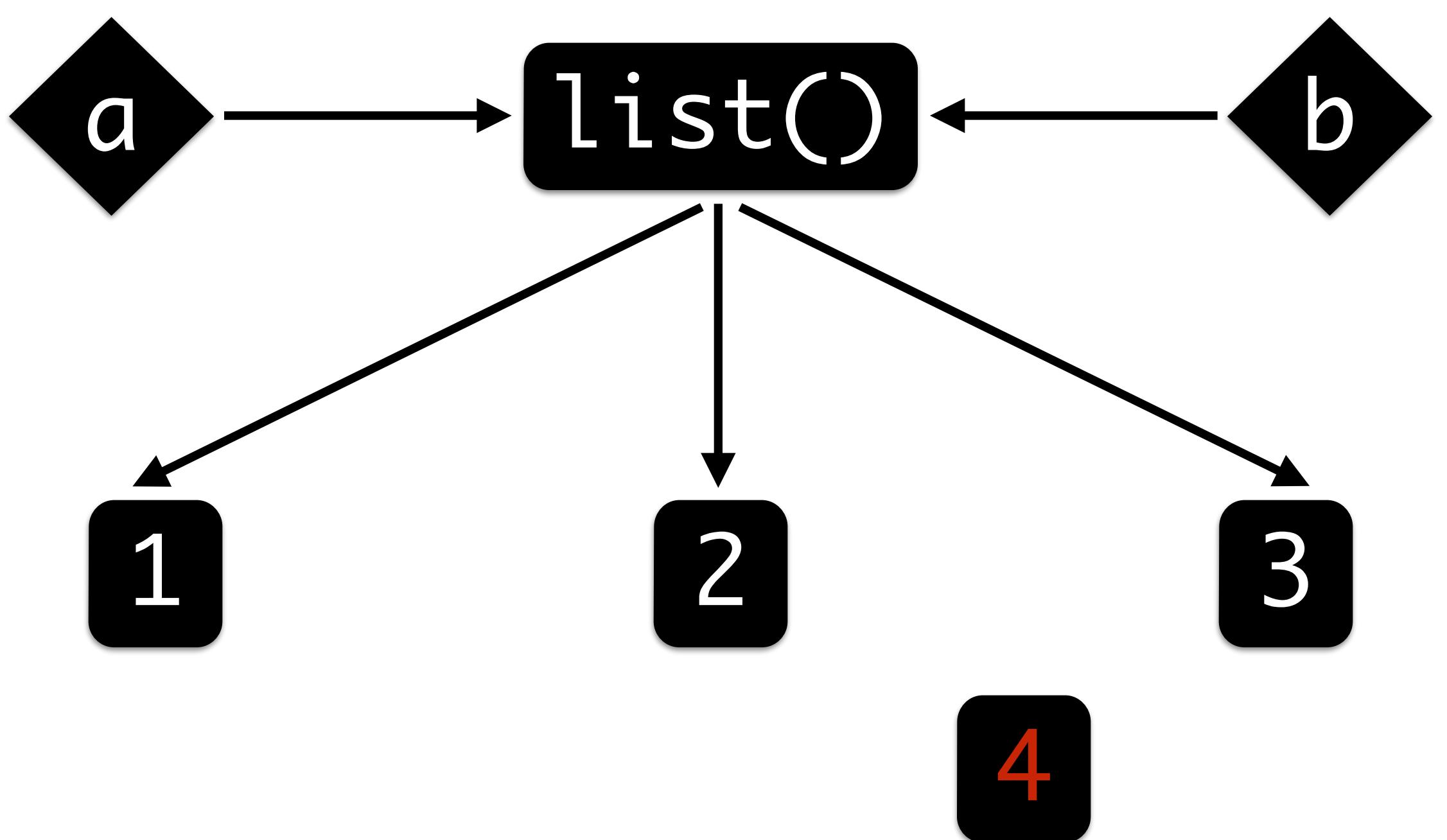


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

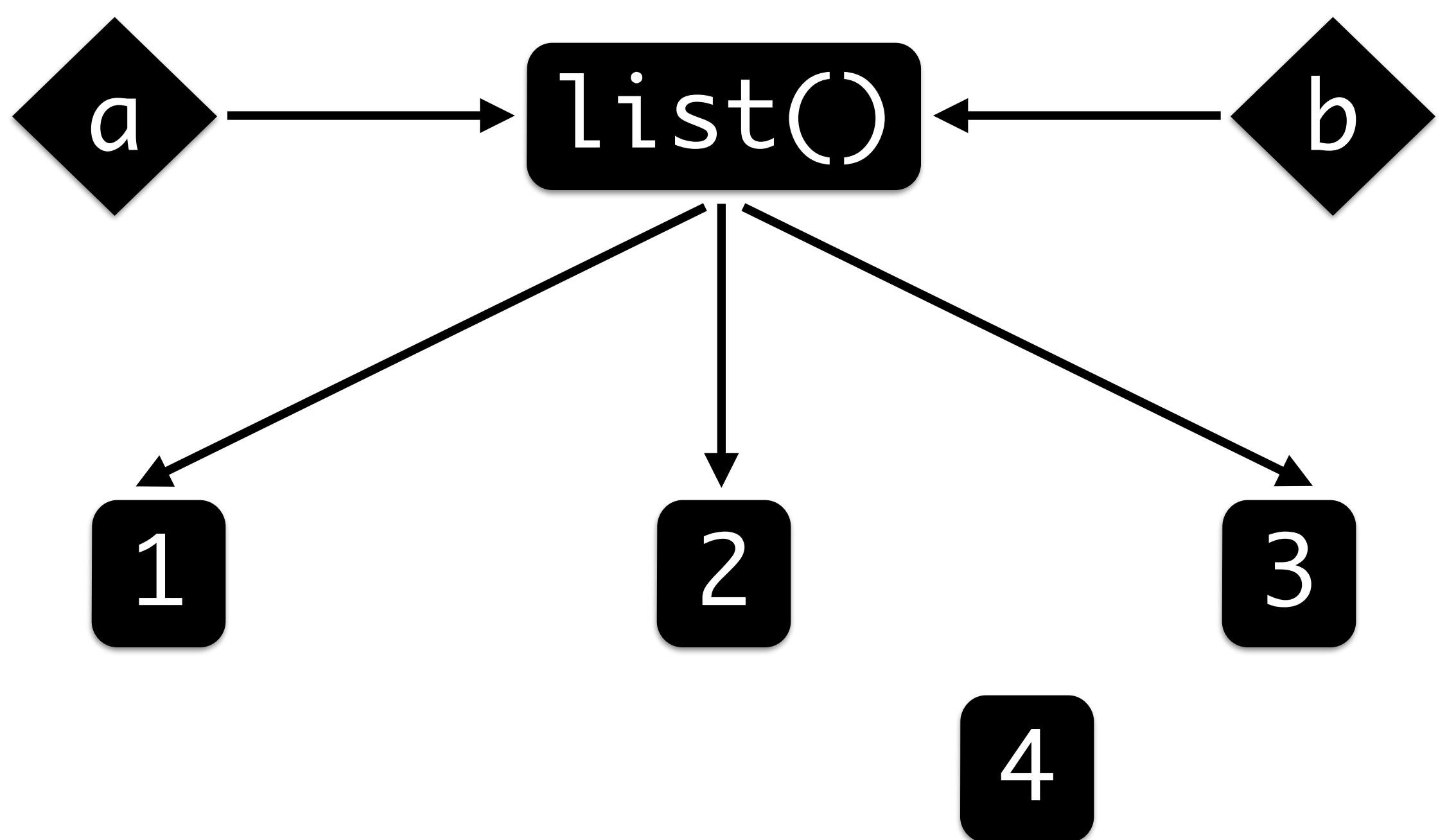


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

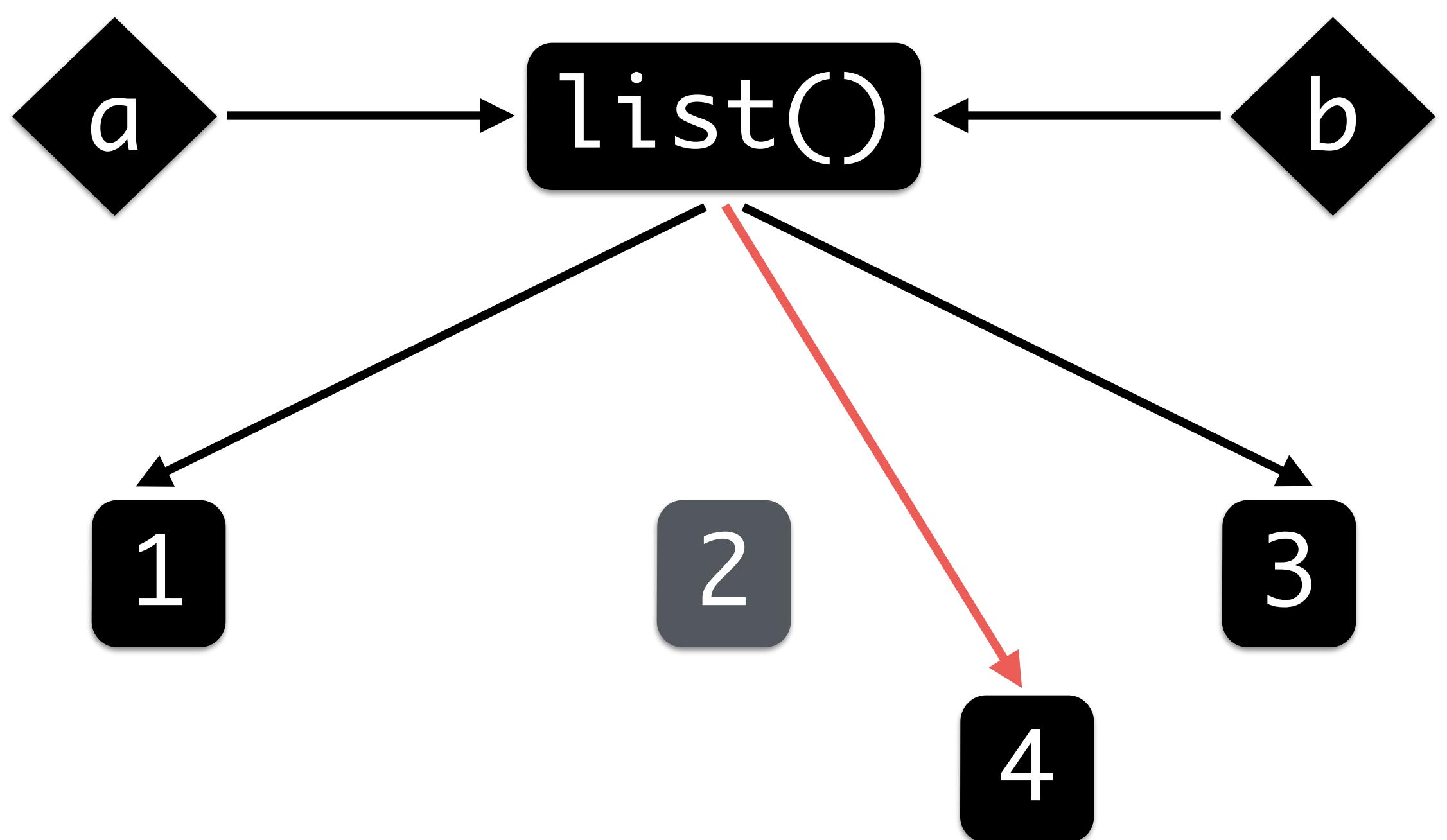


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
print(p)
```

Память

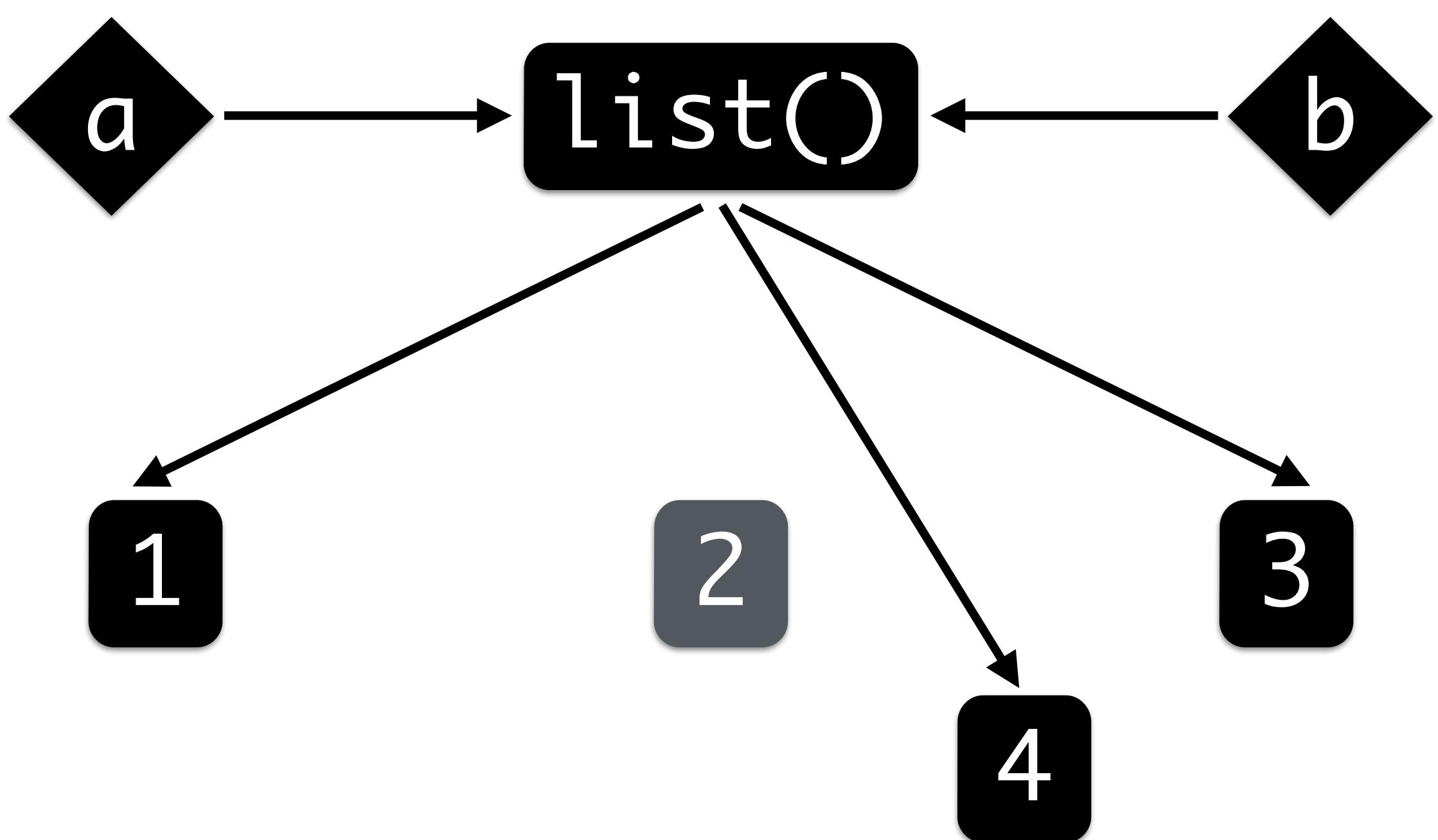


Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
буғынғы
```

Память



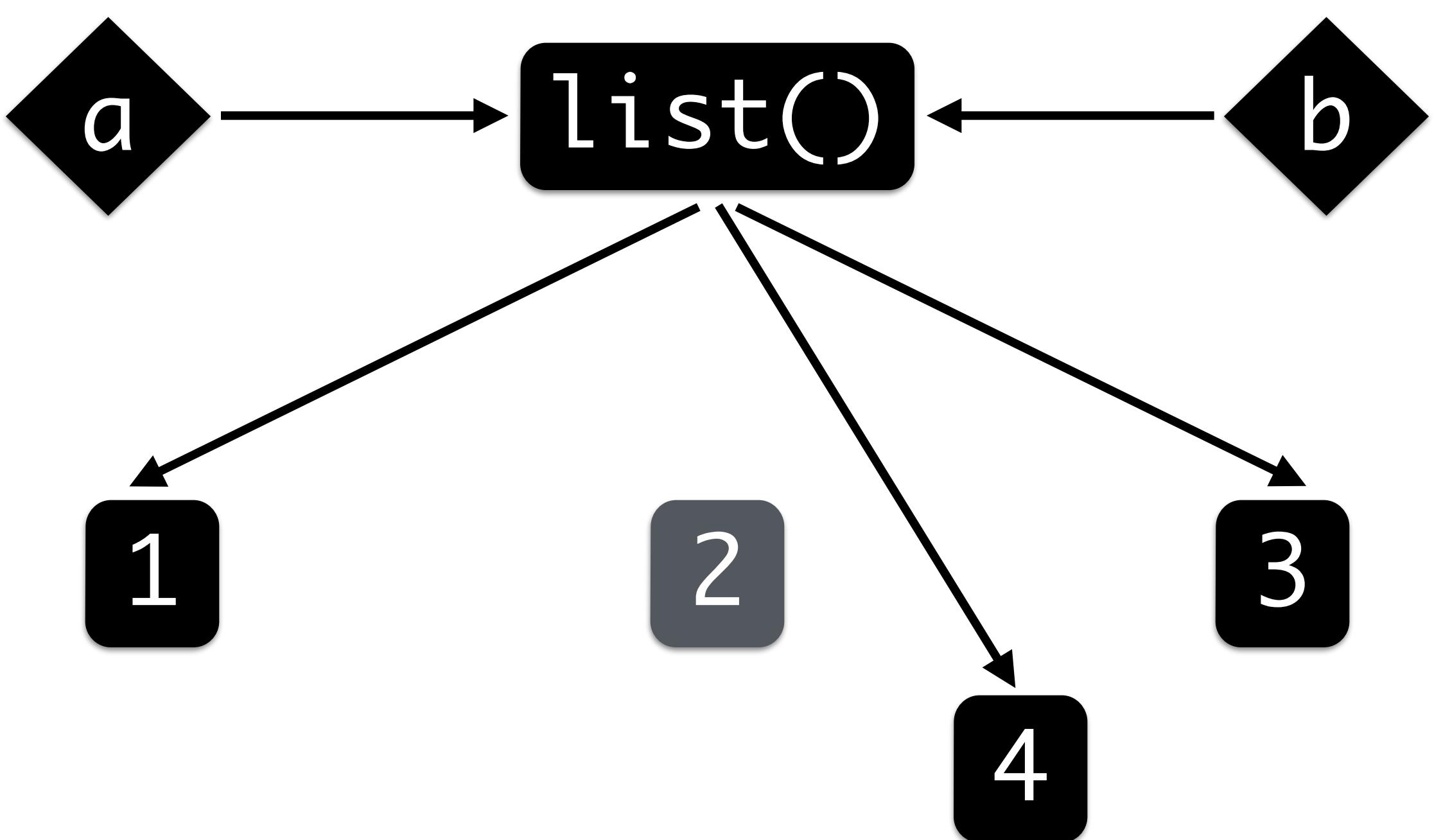
Объекты и переменные

Исходный код

```
a = [1, 2, 3]
b = a
b[1] = 4
print(a)
print(b)
```

вывод

Память



Переменные хранят
ссылки на объекты

locals()

Как получить копию?

Копия при помощи конструктора

Исходный код

```
a = [1, 2, 3]
b = list(a)
b[1] = 4
print(a)
print(b)
```

вывод

Память

Копия при помощи конструктора

Исходный код

```
a = [1, 2, 3]
b = list(a)
b[1] = 4
print(a)
print(b)
print(p)
```

Память

1 2 3

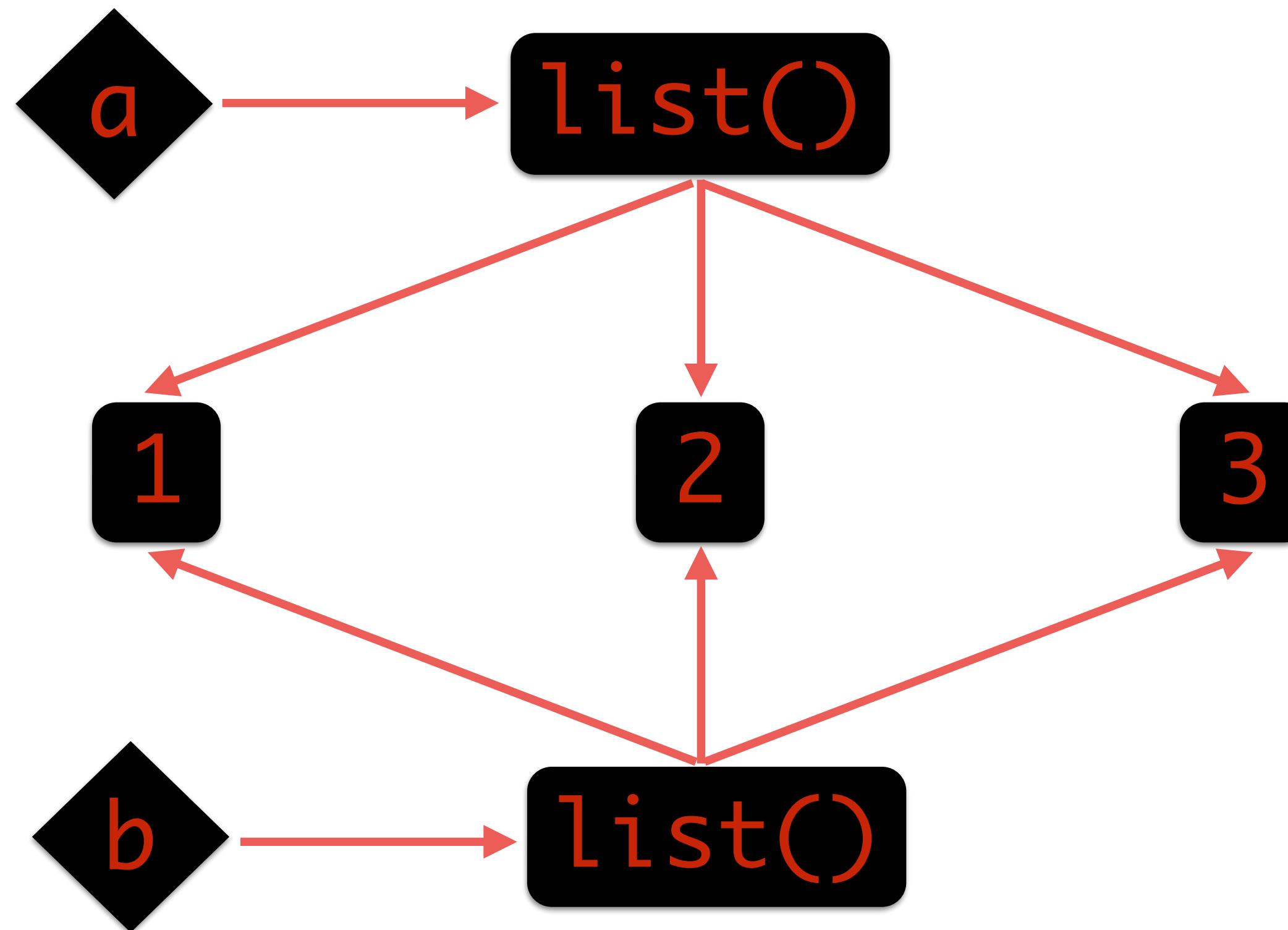
1 4 3

Копия при помощи конструктора

Исходный код

```
a = [1, 2, 3]
b = list(a)
b[1] = 4
print(a)
print(b)
bug(p)
```

Память

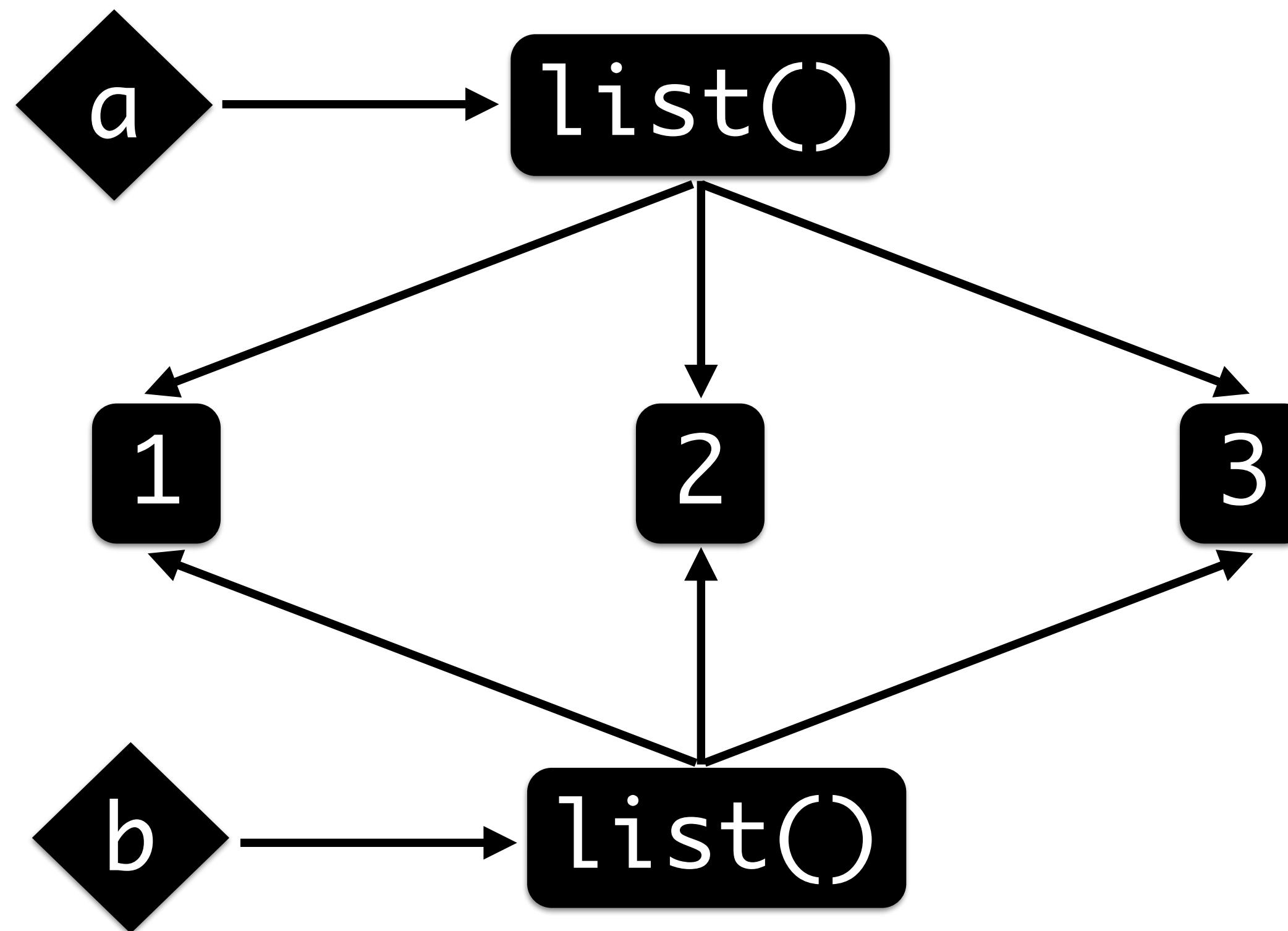


Копия при помощи конструктора

Исходный код

```
a = [1, 2, 3]
b = list(a)
b[1] = 4
print(a)
print(b)
# Вывод
[1, 2, 3]
[1, 4, 3]
```

Память



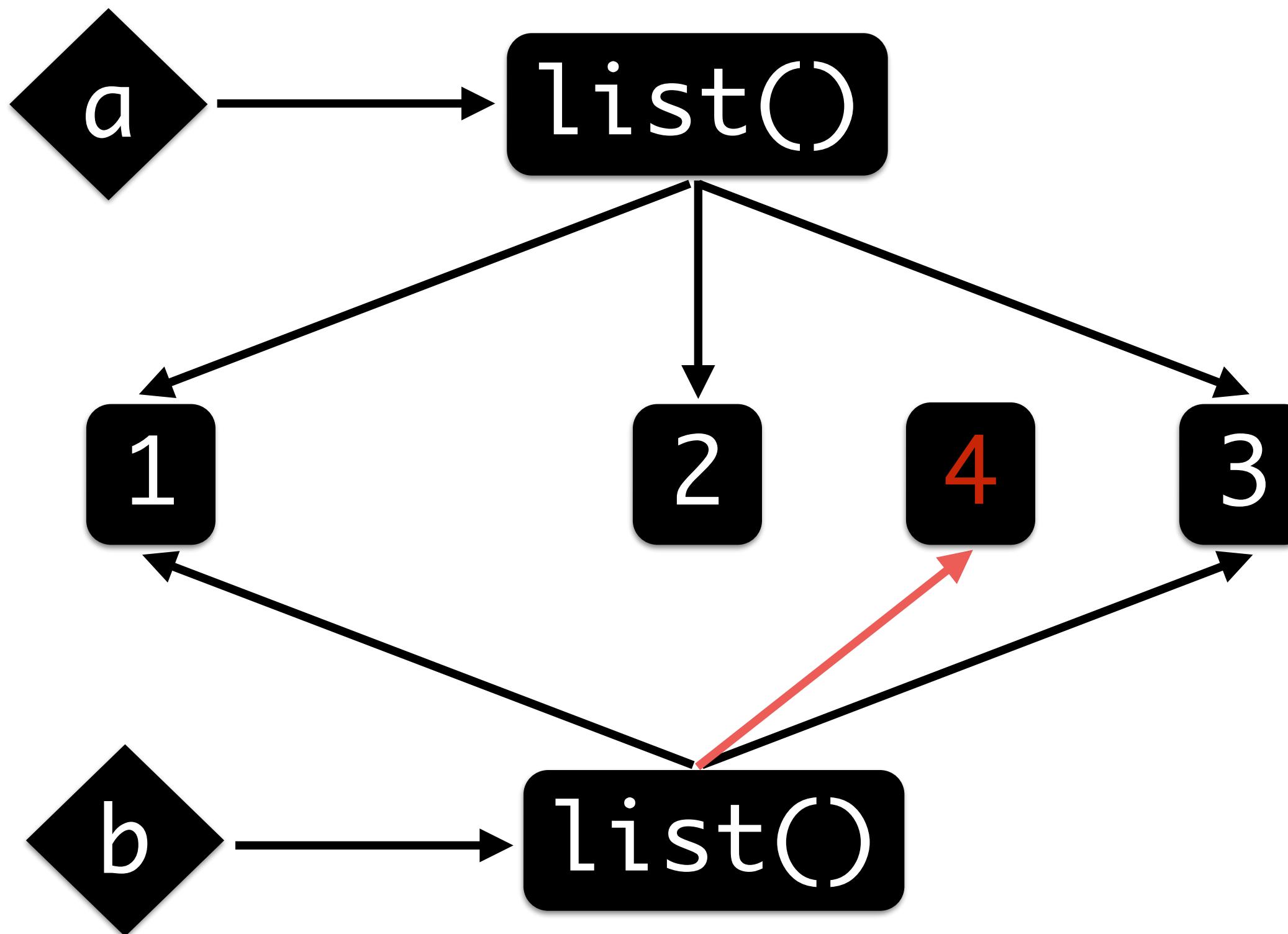
Копия при помощи конструктора

Исходный код

```
a = [1, 2, 3]
b = list(a)
b[1] = 4
print(a)
print(b)
```

вывод

Память

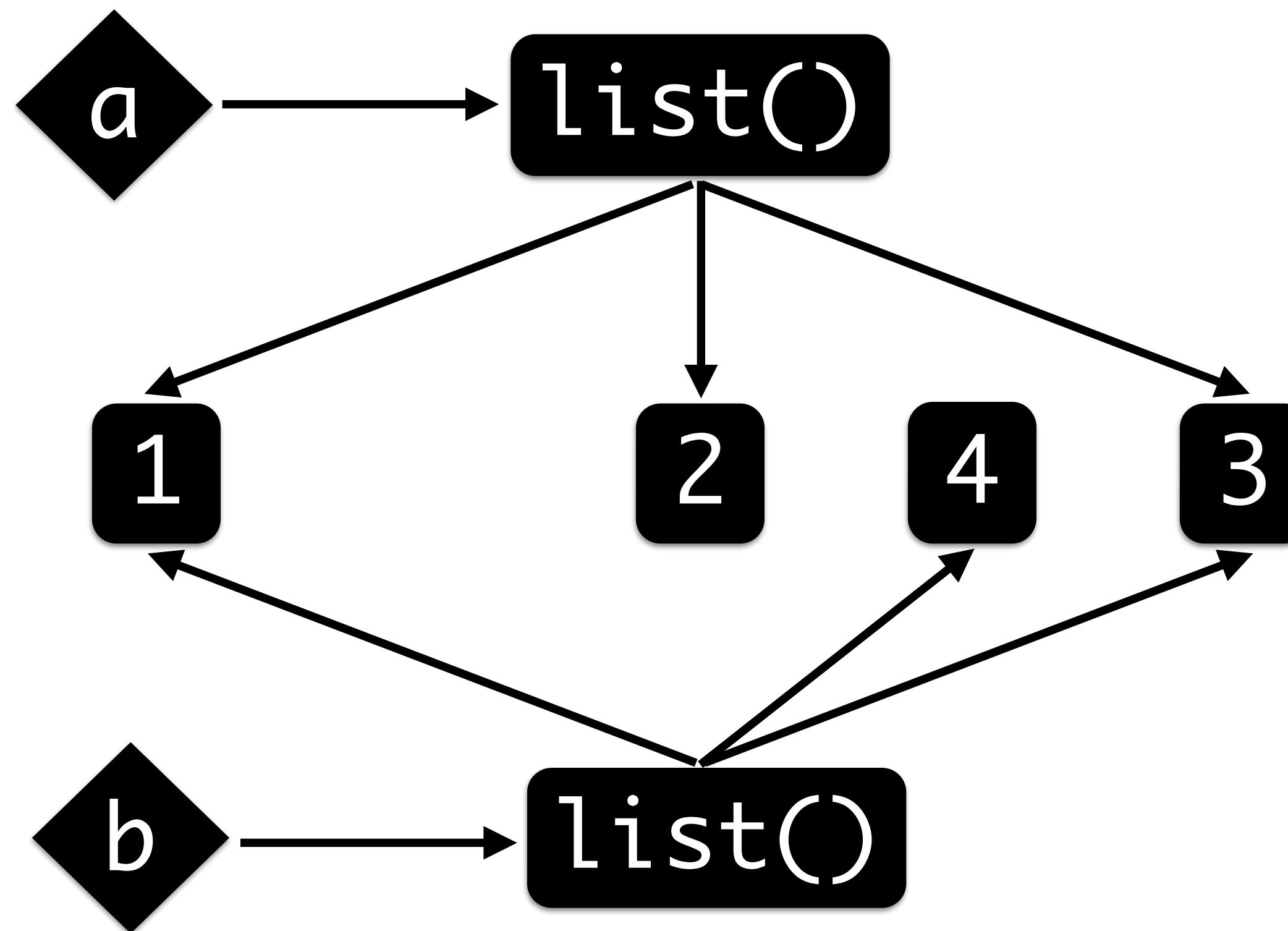


Копия при помощи конструктора

Исходный код

```
a = [1, 2, 3]
b = list(a)
b[1] = 4
print(a)
print(b)
b[1] = 5
```

Память



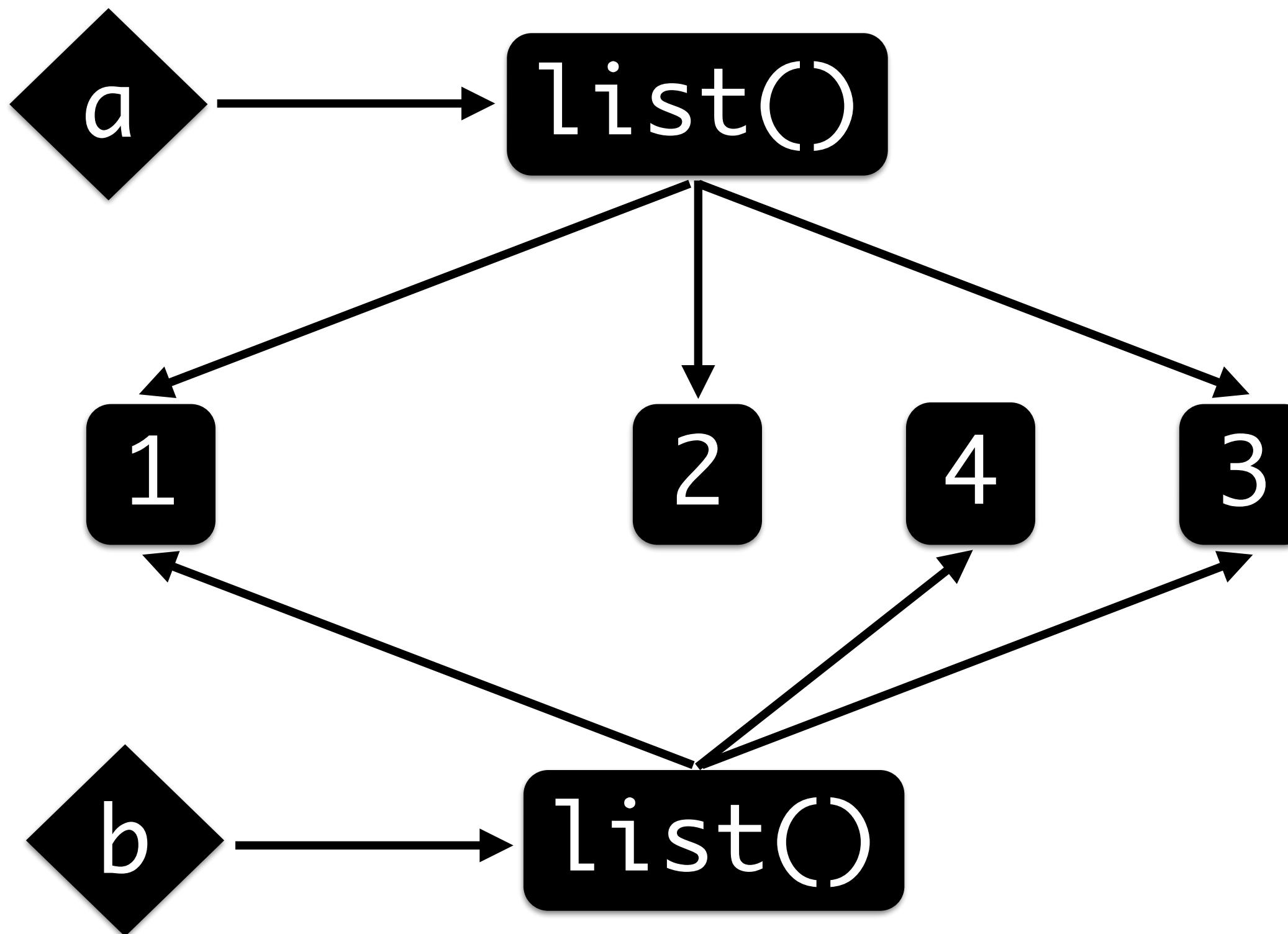
Копия при помощи конструктора

Исходный код

```
a = [1, 2, 3]
b = list(a)
b[1] = 4
print(a)
print(b)
```

вывод

Память



Копия при помощи конструктора

Исходный код

```
a = [1, [2, 3]]  
b = list(a)  
b[1][0] = 4  
print(a)  
print(b)  
print(p)
```

Память

Фрагмент

Копия при помощи конструктора

Исходный код

```
a = [1, [2, 3]]  
b = list(a)  
b[1][0] = 4  
print(a)  
print(b)  
print(p)
```

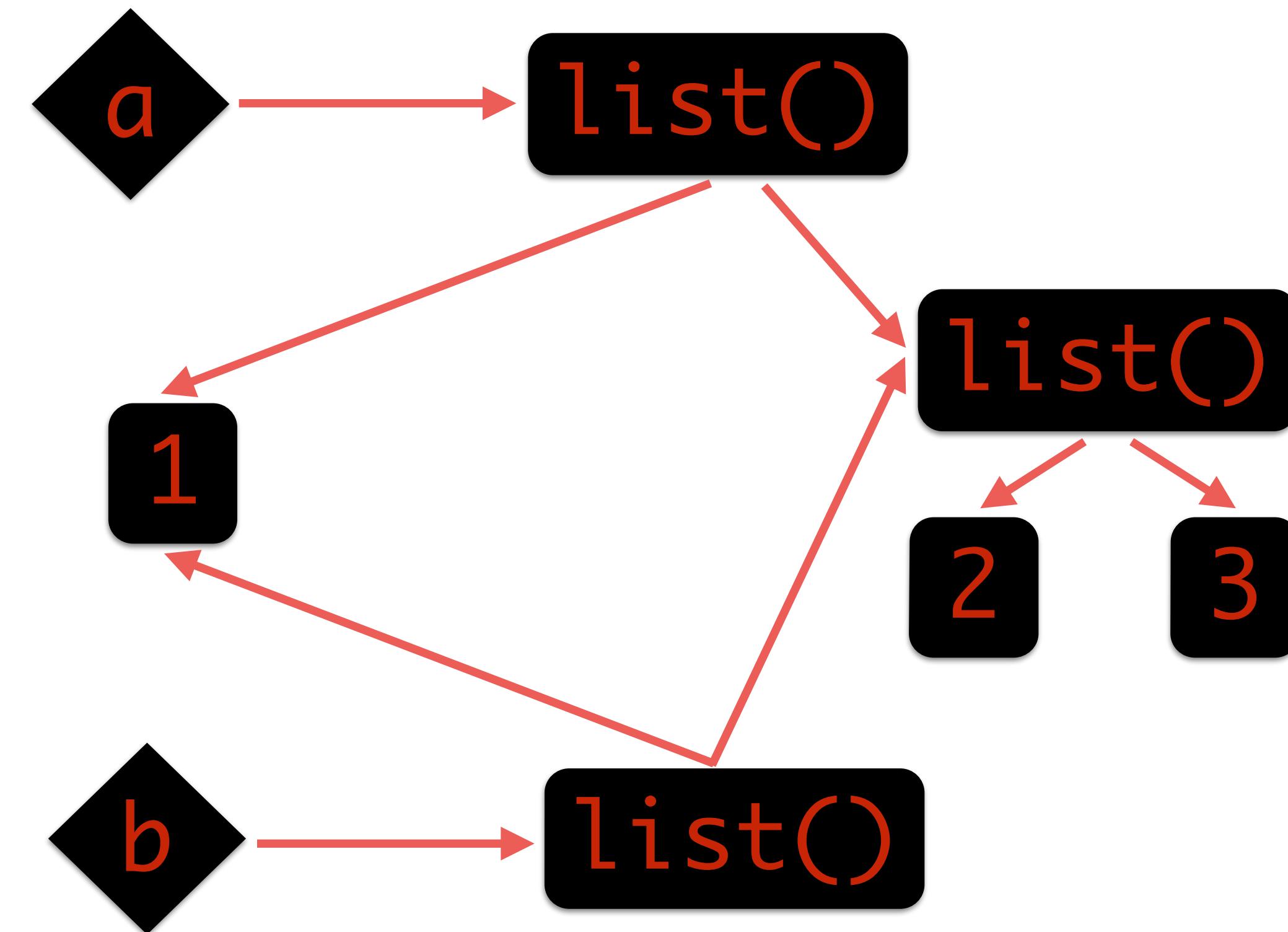
Память

Копия при помощи конструктора

Исходный код

```
a = [1, [2, 3]]  
b = list(a)  
b[1][0] = 4  
print(a)  
print(b)  
буғынғ(р)
```

Память

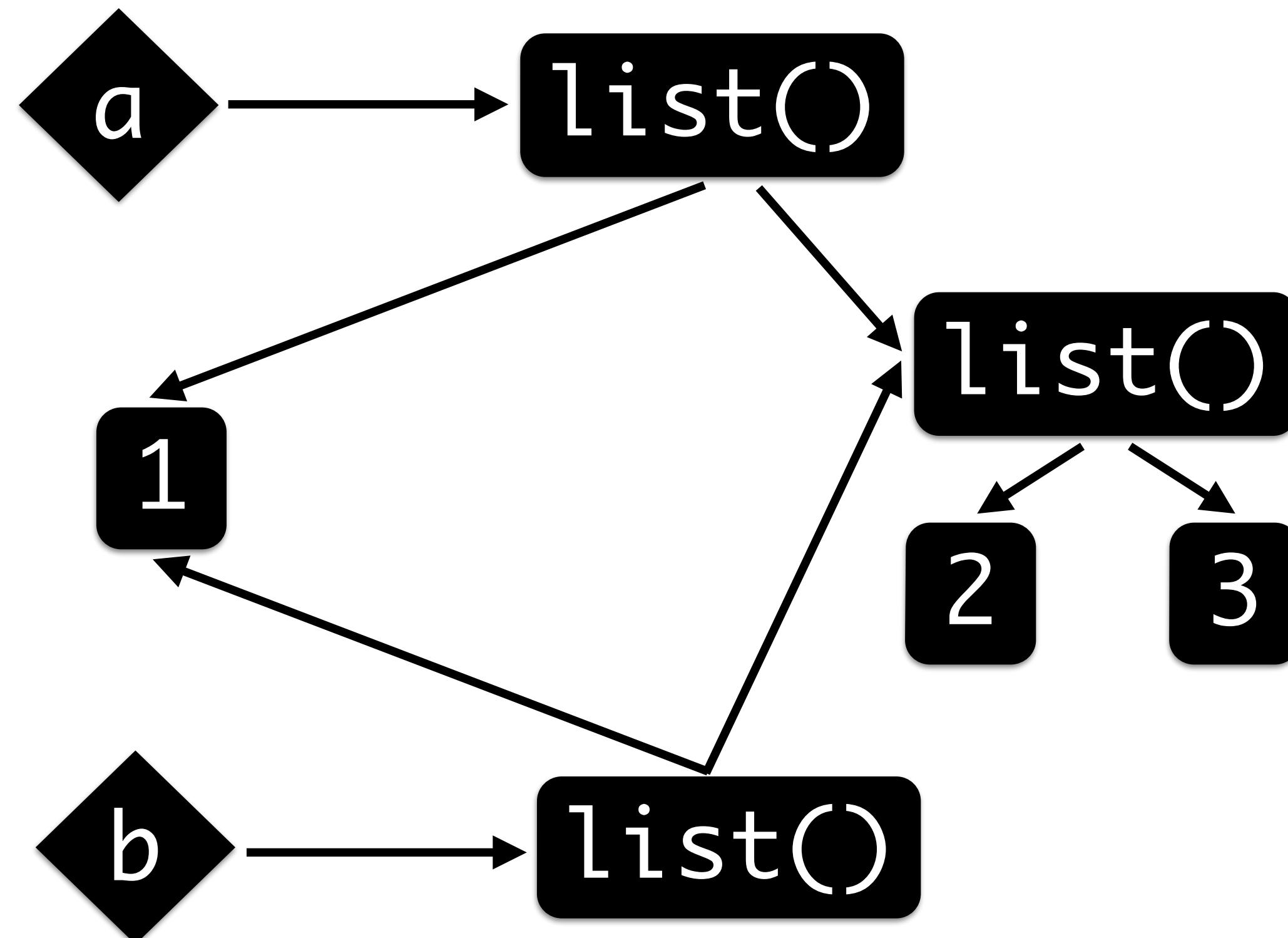


Копия при помощи конструктора

Исходный код

```
a = [1, [2, 3]]  
b = list(a)  
b[1][0] = 4  
print(a)  
print(b)  
буль(p)
```

Память

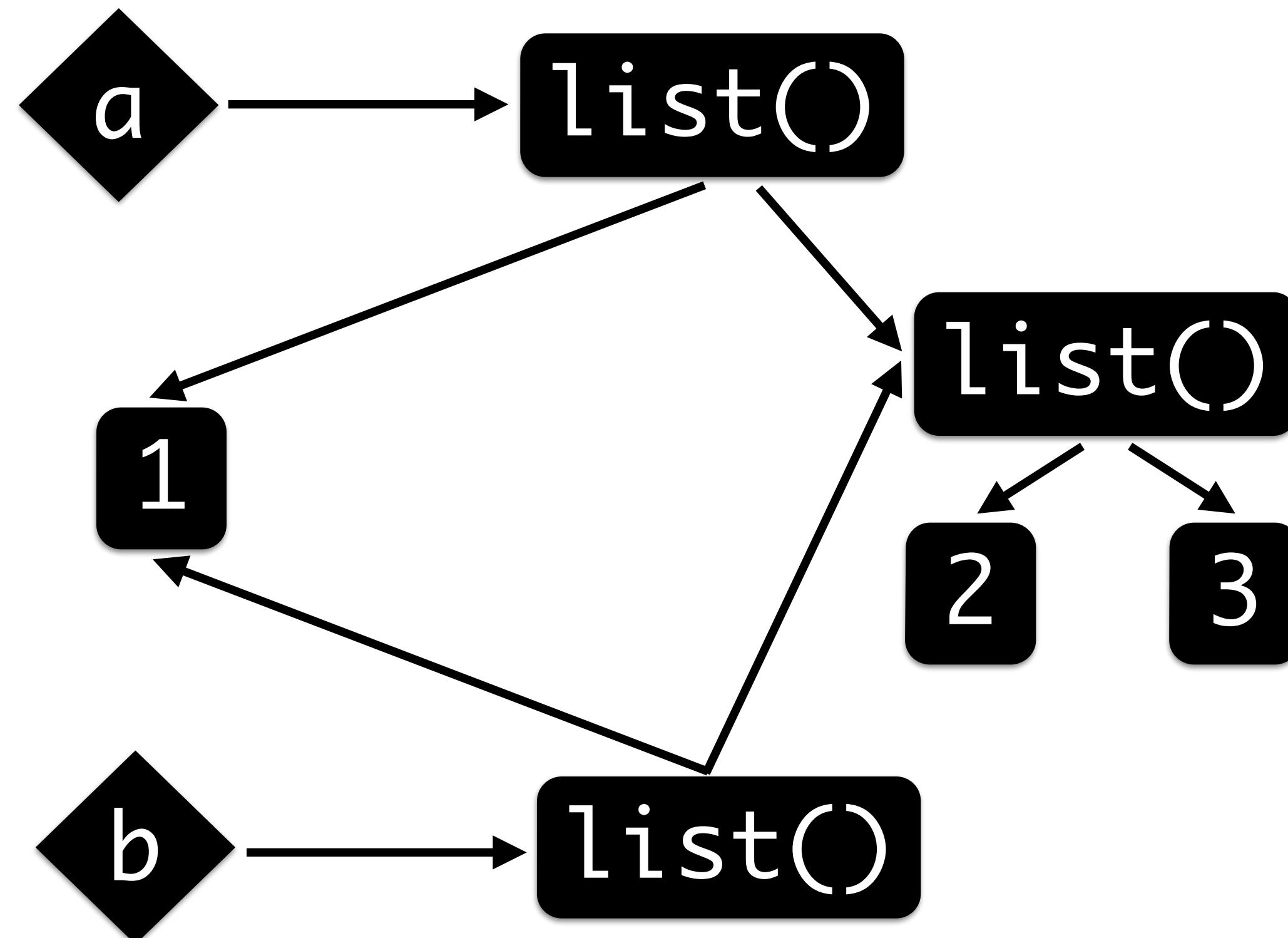


Копия при помощи конструктора

Исходный код

```
a = [1, [2, 3]]  
b = list(a)  
b[1][0] = 4  
print(a)  
print(b)  
буль(p)
```

Память

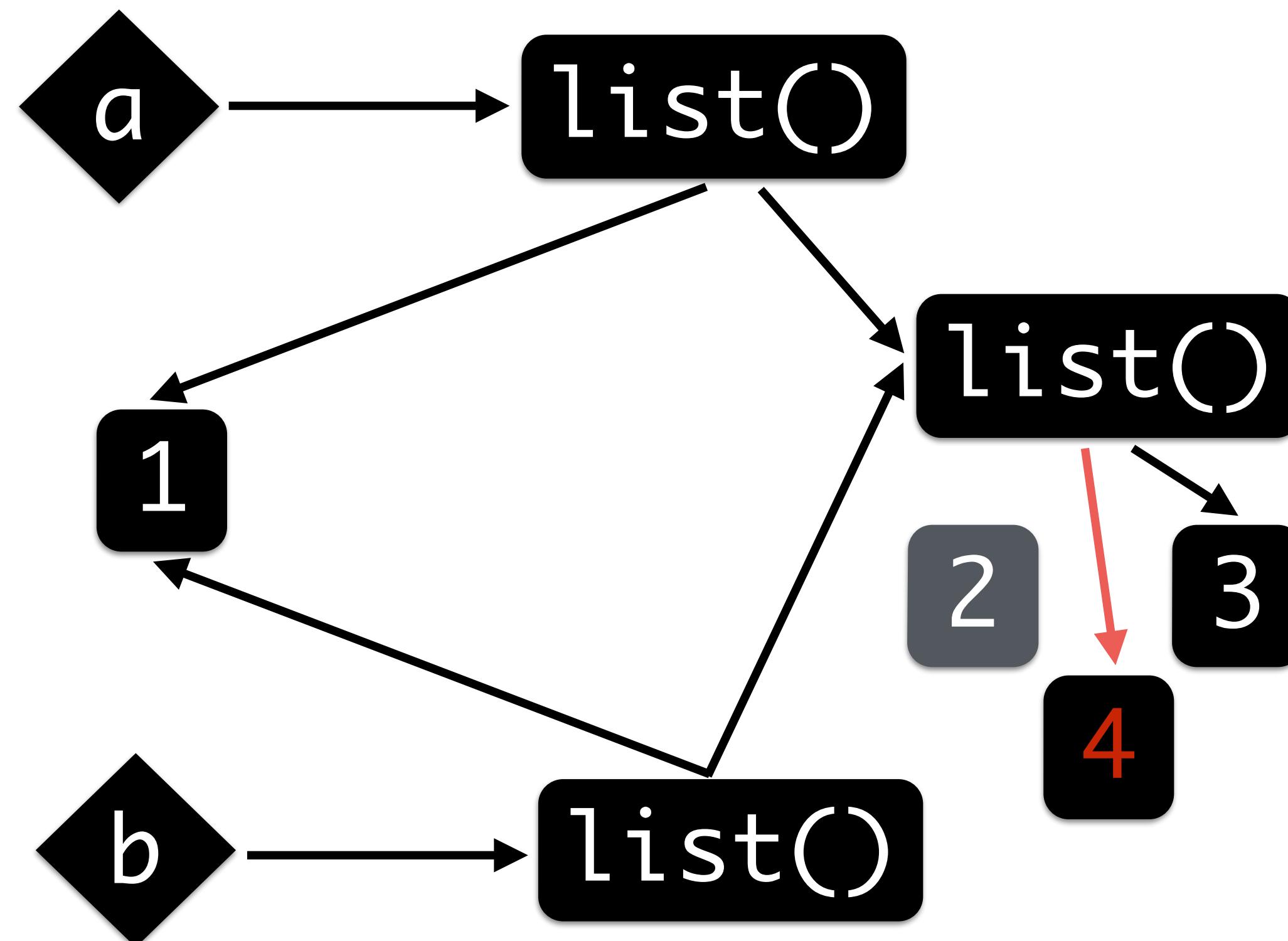


Копия при помощи конструктора

Исходный код

```
a = [1, [2, 3]]  
b = list(a)  
b[1][0] = 4  
print(a)  
print(b)  
буль(p)
```

Память

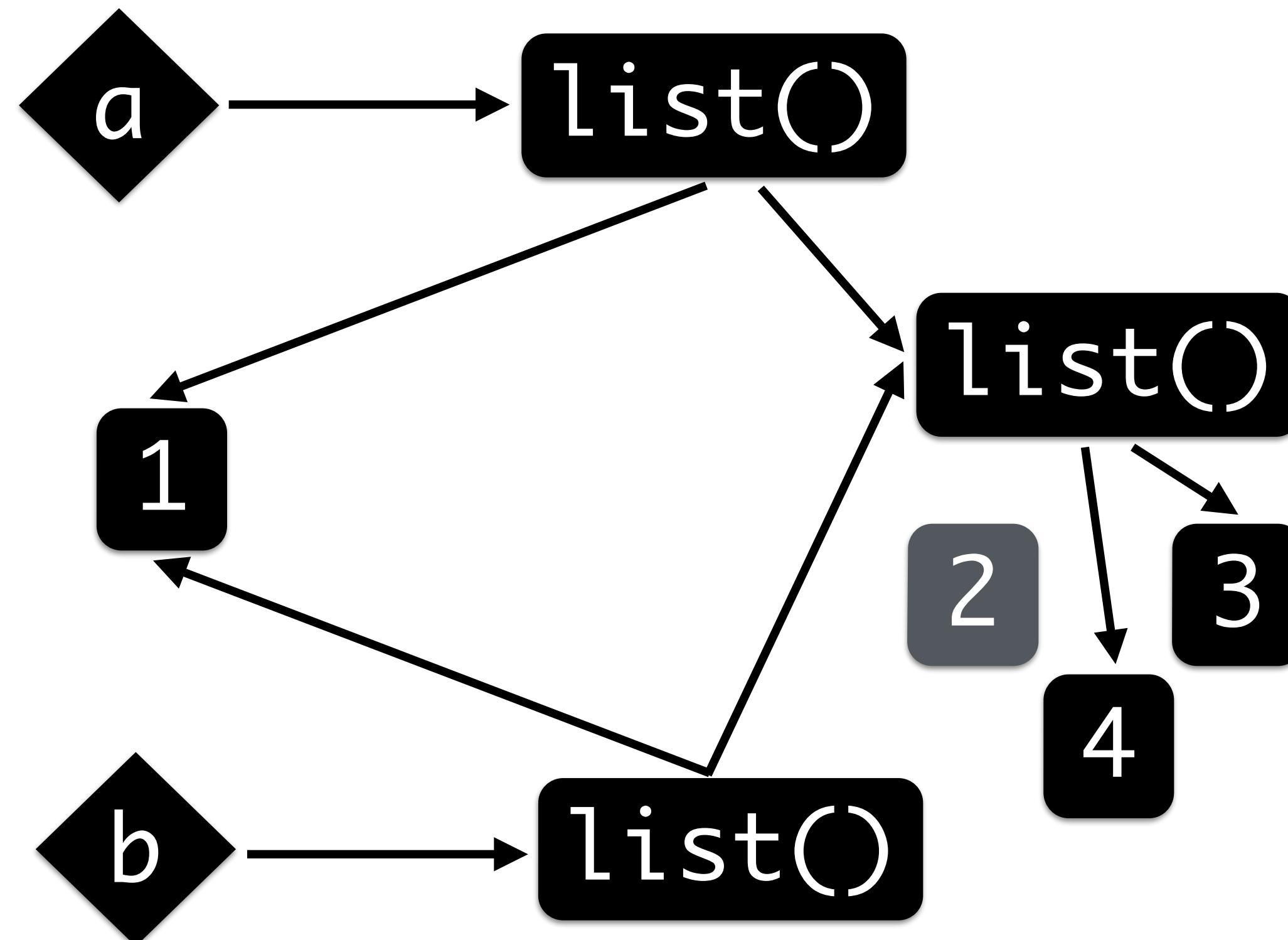


Копия при помощи конструктора

Исходный код

```
a = [1, [2, 3]]  
b = list(a)  
b[1][0] = 4  
print(a)  
print(b)  
вывод(p)
```

Память



Такое копирование называется
поверхностным (shallow copy)

Копия при помощи copy.copy()

Исходный код

```
import copy
a = [1, [2, 3]]
b = copy.copy(a)
b[1][0] = 4
print(a)
print(b)
```

вывод

Память

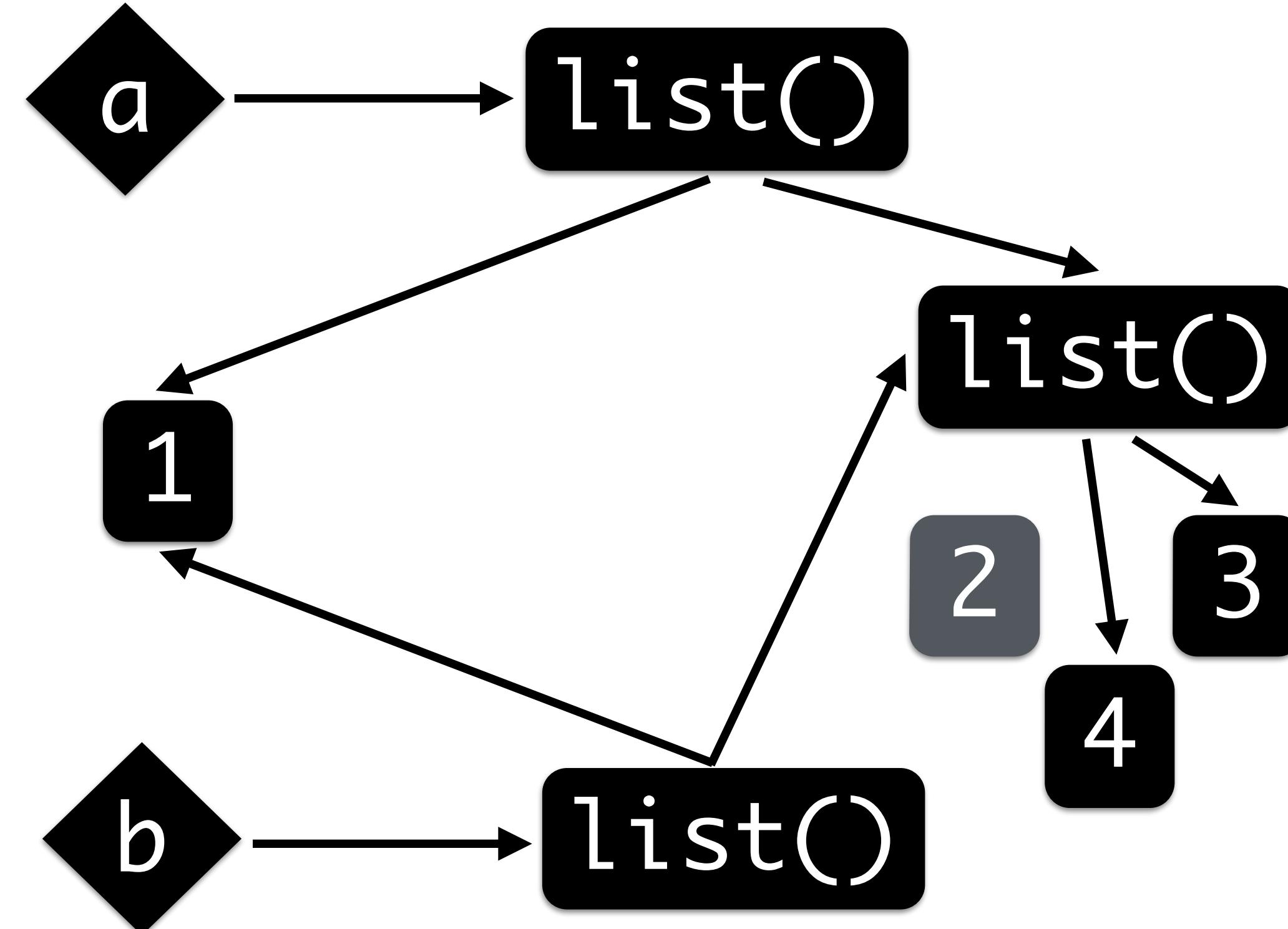
Копия при помощи copy.copy()

Исходный код

```
import copy
a = [1, [2, 3]]
b = copy.copy(a)
b[1][0] = 4
print(a)
print(b)
```

вывод

Память



Копия при помощи `copy.deepcopy()`

Исходный код

```
import copy
a = [1, [2, 3]]
b = copy.deepcopy(a)
b[1][0] = 4
print(a)
print(b)
```

Память

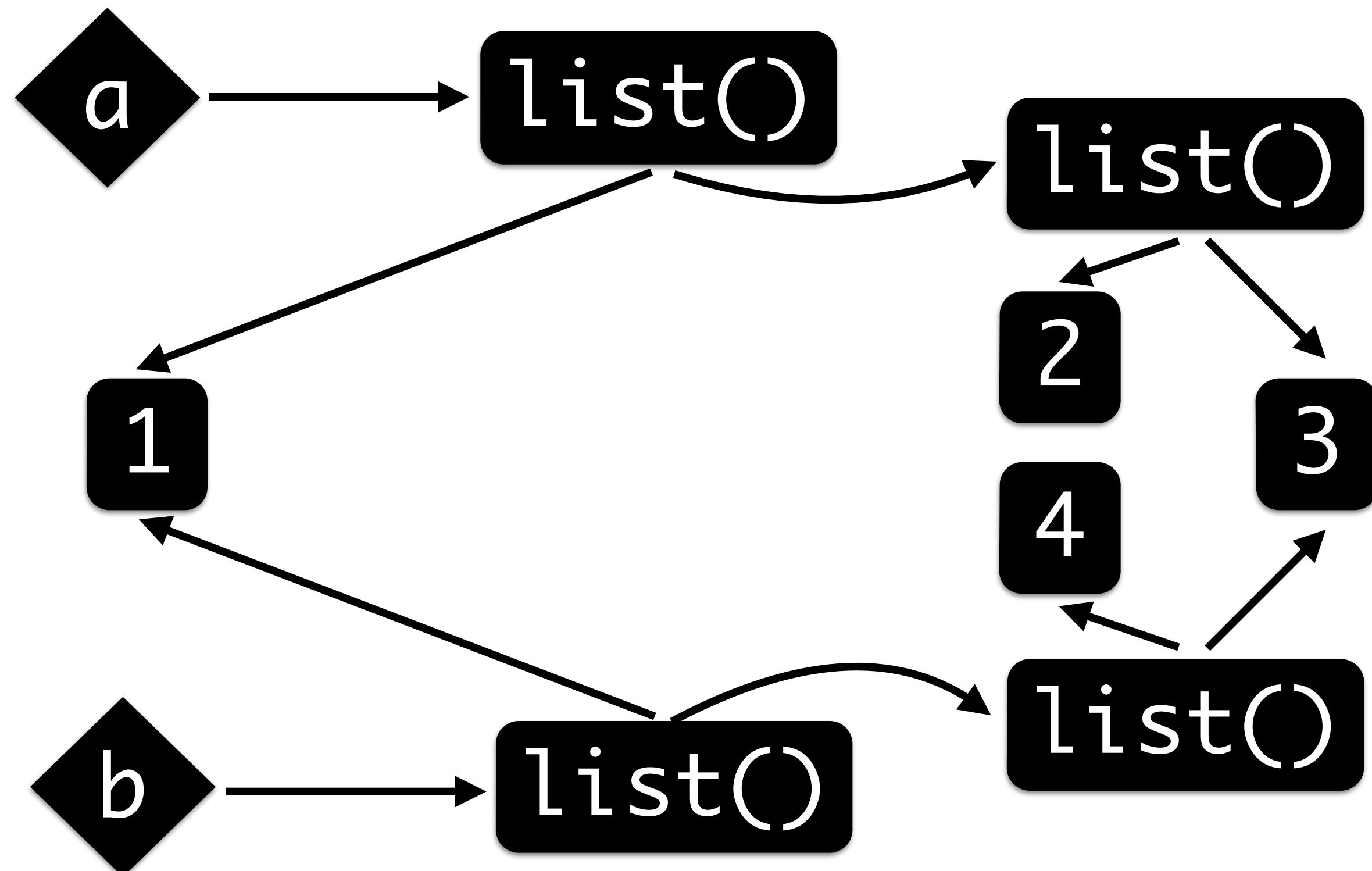
вывод(p)

Копия при помощи copy.deepcopy()

Исходный код

```
import copy
a = [1, [2, 3]]
b = copy.deepcopy(a)
b[1][0] = 4
print(a)
print(b)
```

Память



Но иногда даже `сору.deepcopy()`
недостаточно глубок

Ветвления и циклы

Условные выражения

```
if_stmt ::= "if" expression ":" suite
          ("elif" expression ":" suite )*
          ["else" ":" suite]

suite      ::= stmt_list NEWLINE
              | NEWLINE INDENT statement+ DEDENT

statement   ::= stmt_list NEWLINE | compound_stmt

stmt_list   ::= simple_stmt (";" simple_stmt)* [";"]
```

Цикл while

```
while_stmt ::= "while" expression ":" suite  
              ["else" ":" suite]
```

"break"

"continue"

Цикл **for**

```
for_stmt ::= "for" target_list "in" expression_list ":" suite
           ["else" ":" suite]
```

Устройство цикла **for**

```
product = 1
for i in [1, 2, 4, 8]:
    product *= i
print(product)
```

```
product = 1
i = iter([1, 2, 4, 8])
while True:
    try:
        product *= next(i)
    except StopIteration:
        break
print(product)
```

ФУНКЦИИ

str.format