Lists & Tuples

Alfan Farizki Wicaksono
Universitas Indonesia

Motivation

Buatlah program python yang terus menerus menerima input bilangan integer dan stop ketika input berupa integer negatif.

Lalu, hitung rata-rata dari semua bilangan tersebut!

Motivation

```
sum   = 0
count = 0
val   = int(input("masukkan sebuah bilangan: "))
while val >= 0:
    sum   = sum + val
    count += 1
    val   = int(input("masukkan sebuah bilangan: "))

if count > 0:
    print("rata-rata: {}".format(sum/count))
else:
    print("no data")
```

Motivation

Modifikasi kode sebelumnya sehingga program tidak hanya menampilkan rata-rata; tetapi juga informasi **berapa banyak bilangan yang berada di atas rata-rata**!

Bisakah Anda melakukannya?

Data Structure

- Part of the "science" in computer science is the design and use of data structures and algorithms
- Data structures are particular ways of storing data to make some operation easier or more efficient. That is, they are tuned for certain tasks.
- Python comes with a set of data structures: strings, lists, tuples, dictionaries, and sets.

Lists

Ciri khas:

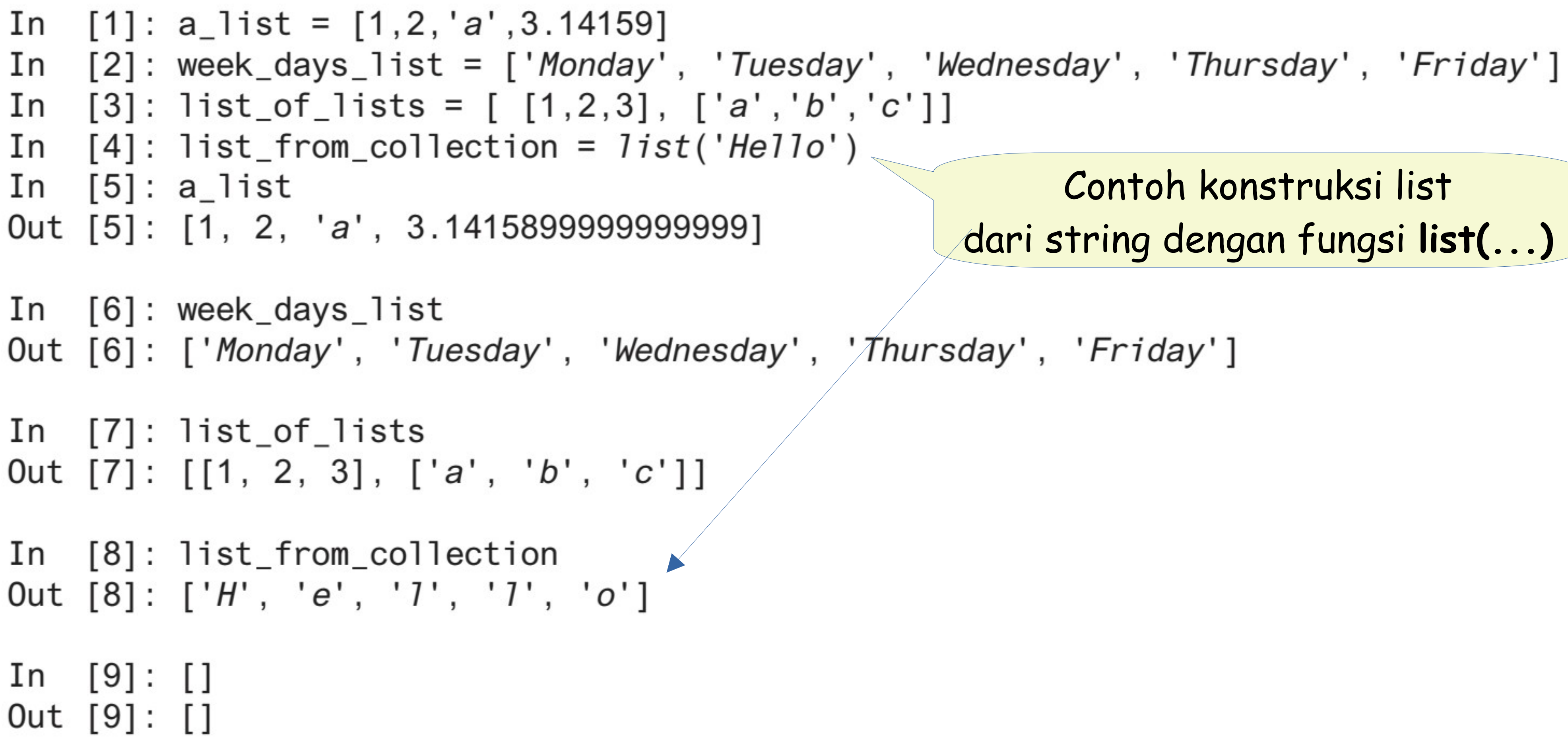
- Ada indeks
- Elemen-elemen boleh berbeda tipe
- Mutable!

```
my_list = [1, 'a', 3.14159, True]
```

1	'a'	3.14159	True	
0	1	2	3	Index Forward
−4	−3	−2	−1	Index Backward

```
my_list[1]  --> 'a'
my_list[:3] --> [1, 'a', 3.14159]
```

List construction: (1) [...] (2) list(...)



```
In  [1]: a_list = [1,2,'a',3.14159]
In  [2]: week_days_list = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
In  [3]: list_of_lists = [ [1,2,3], ['a','b','c']]
In  [4]: list_from_collection = list('Hello')
In  [5]: a_list
Out [5]: [1, 2, 'a', 3.1415899999999999]

In  [6]: week_days_list
Out [6]: ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']

In  [7]: list_of_lists
Out [7]: [[1, 2, 3], ['a', 'b', 'c']]

In  [8]: list_from_collection
Out [8]: ['H', 'e', 'l', 'l', 'o']

In  [9]: []
Out [9]: []
```

Contoh konstruksi list
dari string dengan fungsi **list(...)**

List mirip dengan String :)

- Concatenate (+)
- Repeat (*)
- Indexing ([])
- Slicing ([:])
- Membership (in)
- len()

```
[1, 5, 9] + [45, 7] --> [1, 5, 9, 45, 7]
```

```
[1, 5, 9] * 2 --> [1, 5, 9, 1, 5, 9]
```

```
[1, 5, 9][1] --> 5
```

```
>>> lst = [1, 5, 9]
>>> lst[-1]
9
```

```
>>> lst = [1, 5, 9]
>>> 9 in lst
True
```

```
>>> lst = [1, 5, 9]
>>> lst[:2]
[1,5]
```

```
>>> len([1, 5, 9])
3
```

List mirip dengan String :)

Operasi perbandingan

```
[1, 2, 9] < [1, 2, 4]  -->  False
```

```
[1, 2, 4, 0] > [1, 2, 4]  -->  True
```

compare index to index, first difference determines the result

List boleh berisi list

```
>>> my_list = [1, [2, 3], 4]
>>> len(my_list)
3
>>> my_list[1]
[2, 3]
>>> my_list[1][0]
2
```

List Functions (selain `len(...)`)

- min(lst) : smallest element. Must all be the same type!
- max(lst) : largest element. Must all be the same type!
- sum(lst) : sum of the elements; numeric only.

```
>>> lst = [34, 2, 9]
>>> min(lst)
2
```

```
>>> sum(lst)
45
>>> max(lst)
34
```

Iteration

Template

```
lst = [3, 5, 7, 9]
for elem in lst:
    process(elem)
```

Contoh

```
lst = [3, 5, 7, 9]
sum = 0
for elem in lst:
    sum += elem
print(sum)
```

```
lst = [3, 5, 7, 9]
for i in range(len(lst)):
    process(lst[i])
```

```
lst = [3, 5, 7, 9]
for i in range(len(lst)):
    print(lst[i])
```

Latihan

Buat sebuah fungsi yang menerima sebuah list of integers, lalu menghitung berapa banyak bilangan negatif yang ada di dalam list.

```
def count_neg(lst):
  ....
```

List of lists

Buatlah fungsi **count_neg_v2(lst_of_lsts)**, yang menerima input list integer dua dimensi, dan mengembalikan list 1 dimensi berisi banyaknya elemen negatif dari setiap anggotanya. **Hint: gunakan method append()**

```
>>> count_neg_v2 ([[2,-1,0], [4,3,4], [15,6,2,7,9]])
[1,0,0]
>>> count_neg_v2 ([[0, -1], [-2,-4],[5,5],[4,-4]])
[1,2,0,1]
```

Latihan

Buat sebuah fungsi yang menerima sebuah list of integers, lalu mengembalikan **true** jika list tersebut terurut; dan **false** jika tidak.

```
def is_sorted(lst):
  ....
```

```
>> is_sorted([1,2,3])
True
>> is_sorted([1,3,2])
False
```

*List kosong dan list yang terdiri dari satu elemen adalah terurut

Latihan

Diberikan fungsi geometric() yang mengecek apakah sebuah list berisi bilangan bulan merupakan deret geometri atau bukan. Sebuah deret $a_0, a_1, a_2, \ldots a_{n-2}, a_{n-1}$ merupakan deret geometri jika $a_1/a_0, a_2/a_1, a_3/a_2 \ldots a_{n-1}/a_{n-2}$ semuanya sama.

Contoh:

```
>>> geometric ([1, 3, 5, 7, 9, 11])
False
>>> geometric ([1, 3, 9, 27])
True
```

```
def geometric(lst):

    if len(lst) <= 1:
        return True

    rasio = ....
    for i in range(1, ....):
        ....
    ....
```

Latihan

Buat sebuah fungsi yang menerima sebuah list of integers, lalu mengembalikan list baru hasil **shift left** sekali.

```
def shift_left(lst):
  ....
```

```
>> shift_left([1,2,3])
[2,3,1]
>> shift_left([1,3,2])
[3,2,1]
```

Diberikan sebuah file yang berisi dokumen tekstual. Buatlah program yang meminta input nama file; lalu menampilkan ke layar daftar **semua kata unik** yang ada di dokumen tersebut.

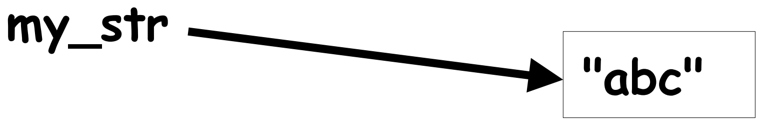
input.txt

```
Belajar programming membutuhkan keteguhan
Programming bukan keterampilan yang didapatkan secara instan
Setiap orang perlu fokus agar bisa programming
Fokus belajar programming bukan berarti hidup akan terbelenggu
```

String (immutable) vs List (mutable)

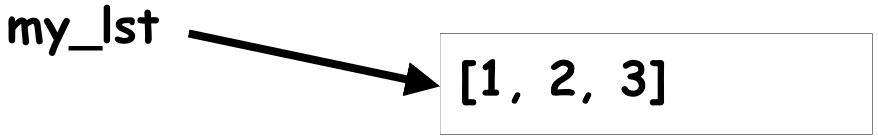
String object bersifat immutable. Setelah dibuat, **isi dari string object tidak bisa diubah**.

```
my_str = 'abc'
my_str[0] = 'z'  # cannot do!

# instead, make new str
new_str = my_str.replace('a','z')
c
```



List objects bersifat **mutable**!

```
my_lst = [1, 2, 3]
my_lst[0] = 129  # ok

print(my_lst)  # [129, 2, 3]
```



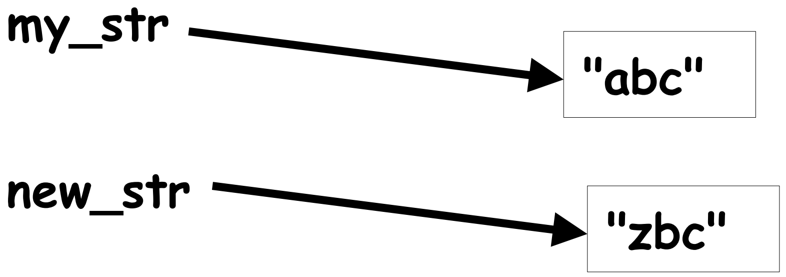
String (immutable) vs List (mutable)

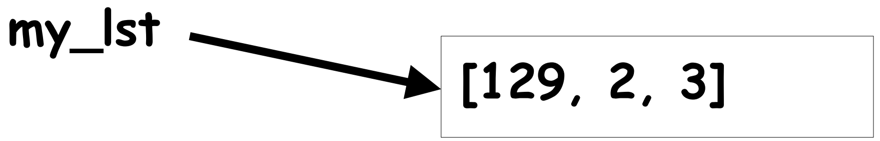
String object bersifat **immutable**. Setelah dibuat, **isi dari string object tidak bisa diubah.**

```
my_str = 'abc'
my_str[0] = 'z'  # cannot do!

# instead, make new str
new_str = my_str.replace('a','z')
```

List objects bersifat **mutable**!

```
my_lst = [1, 2, 3]
my_lst[0] = 129  # ok

print(my_lst)  # [129, 2, 3]
```





List Methods (beda lho dengan list functions)

Contoh method yang sering digunakan: **append()**

```
my_list = ['a', 1, True]
my_list.append('z')
```

Argument

The list object

The name of the method

Note: method ini tidak mengembalikan nilai (alias return None)

my_lst → ['a', 1, True, 'z']

Usage	Explanation
`lst.append(item)`	adds `item` to the end of `lst`
`lst.count(item)`	returns the number of times `item` occurs in `lst`
`lst.index(item)`	Returns index of (first occurrence of) `item` in `lst`
`lst.pop()`	Removes and returns the last item in `lst`
`lst.remove(item)`	Removes (the first occurrence of) `item` from `lst`
`lst.reverse ()`	Reverses the order of items in `lst`
`lst.sort ()`	Sorts the items of `lst` in increasing order

Methods `append()`, `remove()`, `reverse()`, and `sort()` do not return any value; they, along with method `pop()`, modify list `lst`

```
>>> lst = [1, 2, 3]
>>> lst.append(7)
>>> lst.append(3)
>>> lst
[1, 2, 3, 7, 3]
>>> lst.count(3)
2
>>> lst.remove(2)
>>> lst
[1, 3, 7, 3]
>>> lst.reverse()
>>> lst
[3, 7, 3, 1]
>>> lst.index(3)
0
>>> lst.sort()
>>> lst
[1, 3, 3, 7]
>>> lst.remove(3)
>>> lst
[1, 3, 7]
>>> lst.pop()
7
>>> lst
[1, 3]
```

Usage	Explanation
`lst.extend(C)`	Requires a collection C as an argument. The list is extended by adding *each individual* element of the argument collection C to the end of the list.
`lst.insert(i,x)`	Inserts an element at a given position. The first argument is the index *before* which to insert in the lst. Thus my list.insert(1, '*a*') inserts the '*a*' into position 1 of the list, sliding all the rest of the list elements down one

```
In [1]: lst = [1, 12, 5, 8]
In [2]: lst.insert(2,100)
In [3]: lst
Out[4]: [1, 12, 100, 5, 8]

In [5]: lst.extend(list('Hello')
In [6]: lst
Out[7]: [1, 12, 100, 5, 8, 'H', 'e', 'l', 'l', 'o']
```

Notes

Kebanyakan dari list methods tidak mengembalikan nilai (`return None`)

```
my_list = [4, 7, 1, 2]
my_list.append(9)
my_list = my_list.sort()
print(my_list) # apa yang terjadi?
```

Split & Join

```
>>> sentence = 'halo selamat belajar'
>>> words = sentence.split()
>>> words
['halo', 'selamat', 'belajar']
>>> words.reverse()
>>> words
['belajar', 'selamat', 'halo']
>>> ' '.join(words)
'belajar selamat halo'
>>> ','.join(words)
'belajar,selamat,halo'
```

sorted() function **vs** sort() method

```
>>> my_list = [27, 53, 8, 11]
>>> sorted_list = sorted(my_list)
>>> sorted_list
[8, 11, 27, 53]
>>> sorted_list = sorted(my_list, reverse = True)
[53, 27, 11, 8]
```

my_list tidak berubah!

```
>>> my_list = [27, 53, 8, 11]
>>> my_list.sort()
>>> my_list
[8, 11, 27, 53]
```

my_list berubah!

More on sorting

```
from operator import itemgetter

word_freq = [['pergi', 41], ['sebuah', 17], ['orang', 39]]

print(sorted(word_freq)) # default sort on index 0
# [['orang', 39], ['pergi', 41], ['sebuah', 17]]

print(sorted(word_freq, key = itemgetter(0))) #sort on index 0
# [['orang', 39], ['pergi', 41], ['sebuah', 17]]

print(sorted(word_freq, key = itemgetter(1))) #sort on index 1
# [['sebuah', 17], ['orang', 39], ['pergi', 41]]
```

Sort on index 0, and then 1

```
from operator import itemgetter

# daftar nama & umur
data_penduduk = [['rudi', 41], ['andi', 39], ['andi', 17]]
print(sorted(data_penduduk, key = itemgetter(0, 1)))
# [['andi', 17], ['andi', 39], ['rudi', 41]]
```

Latihan

Buat program yang menerima dua buah kata; lalu periksa apakah dua buah kata tersebut merupakan **anagram**.

```
Kata pertama: rudi
Kata kedua    : duri


anagram
```

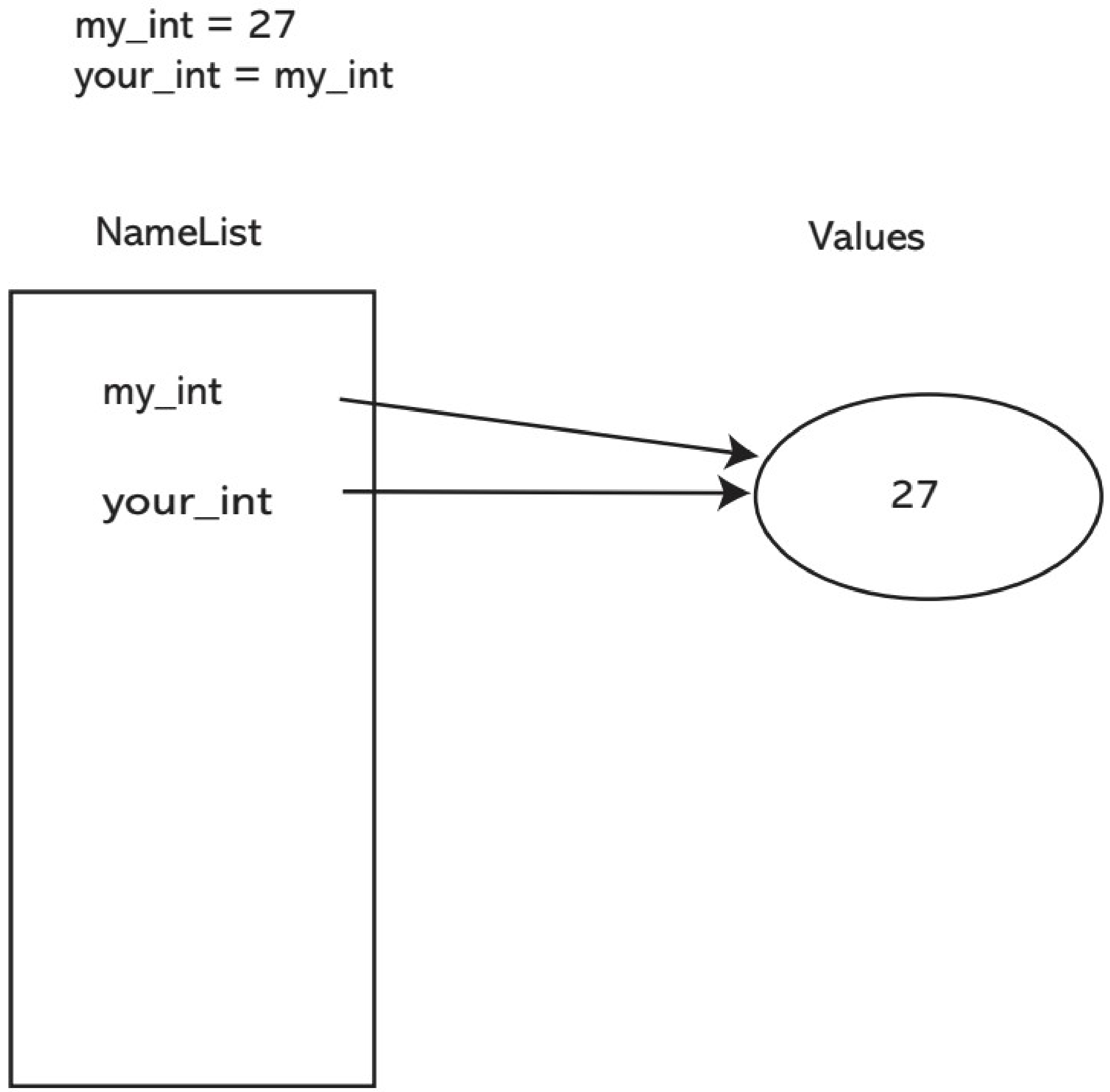
```
Kata pertama: andi
Kata kedua    : dika


bukan anagram
```

List of lists

Buat sebuah fungsi yang menerima sebuah list of lists of integers; lalu kembalikan list of lists of booleans yang bersesuaian sehingga jika bilangan >=0 diganti dengan True dan bilangan negatif diganti dengan False.

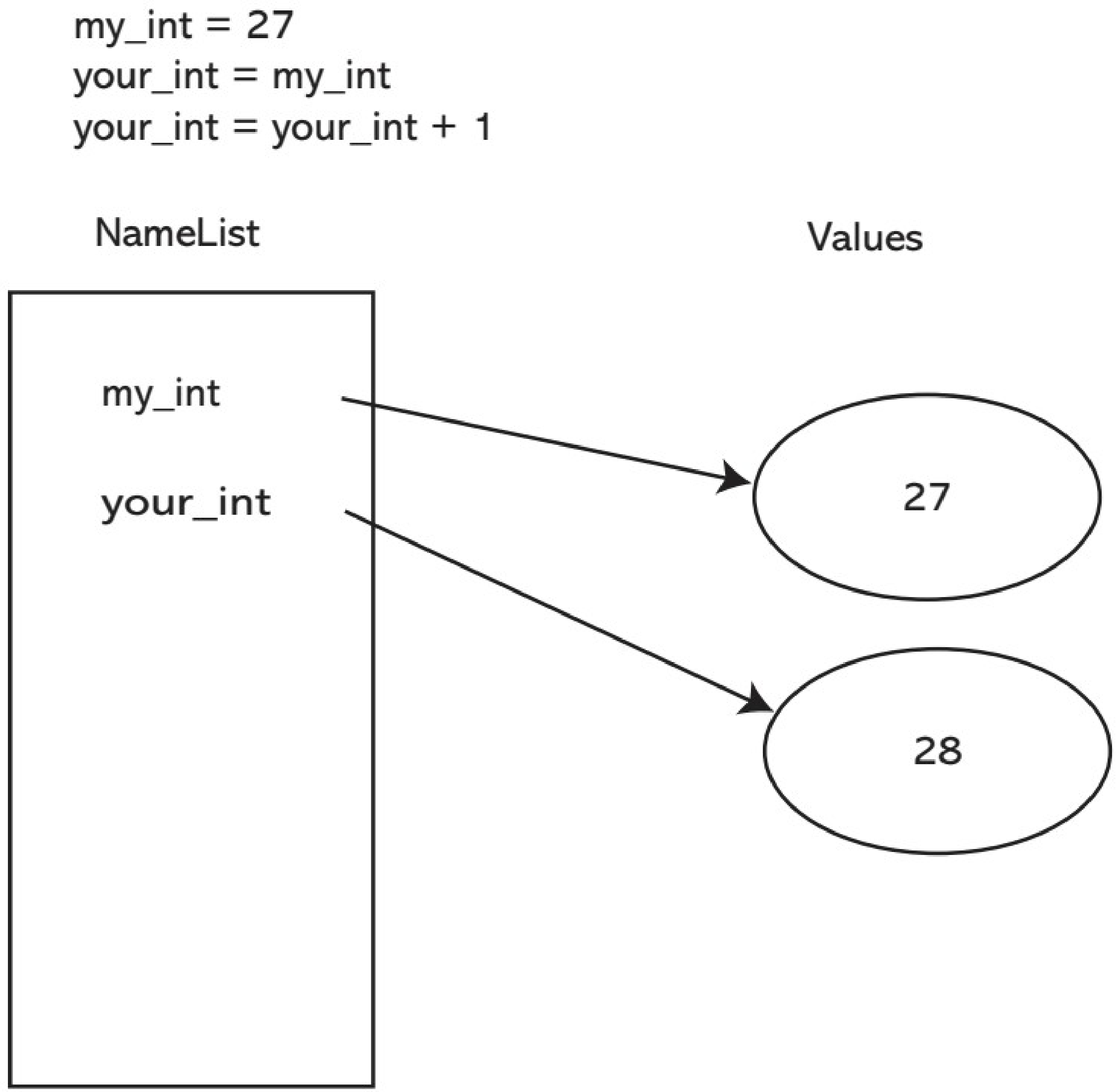
```
def check(lsts):
  ....
```

```
>> check([[1, 0, -2], [-3, 4], [-9, -1, -6, 1]])
[[True, True, False], [False, True], [False, False, False, True]]
```



```
my_int = 27
your_int = my_int
```

FIGURE 7.2 Namespace snapshot #1.



```
my_int = 27
your_int = my_int
your_int = your_int + 1
```

FIGURE 7.3 Modification of a reference to an immutable object.

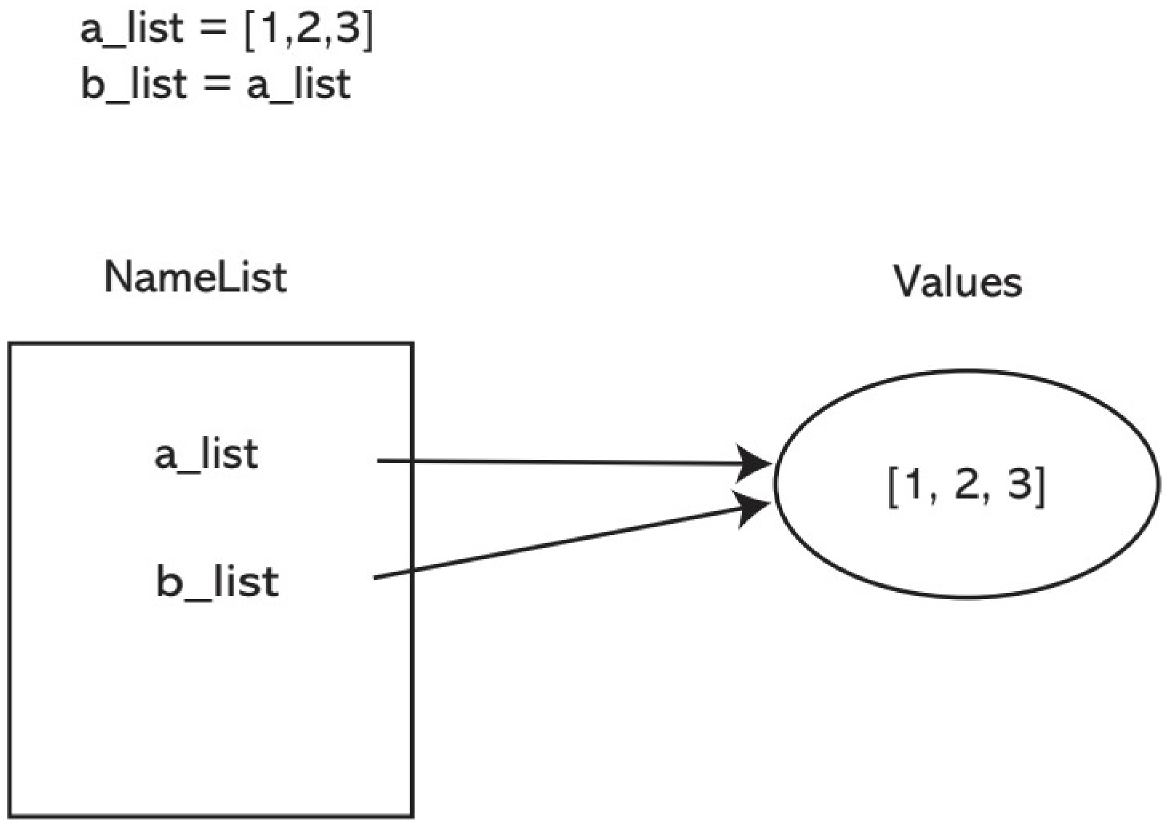
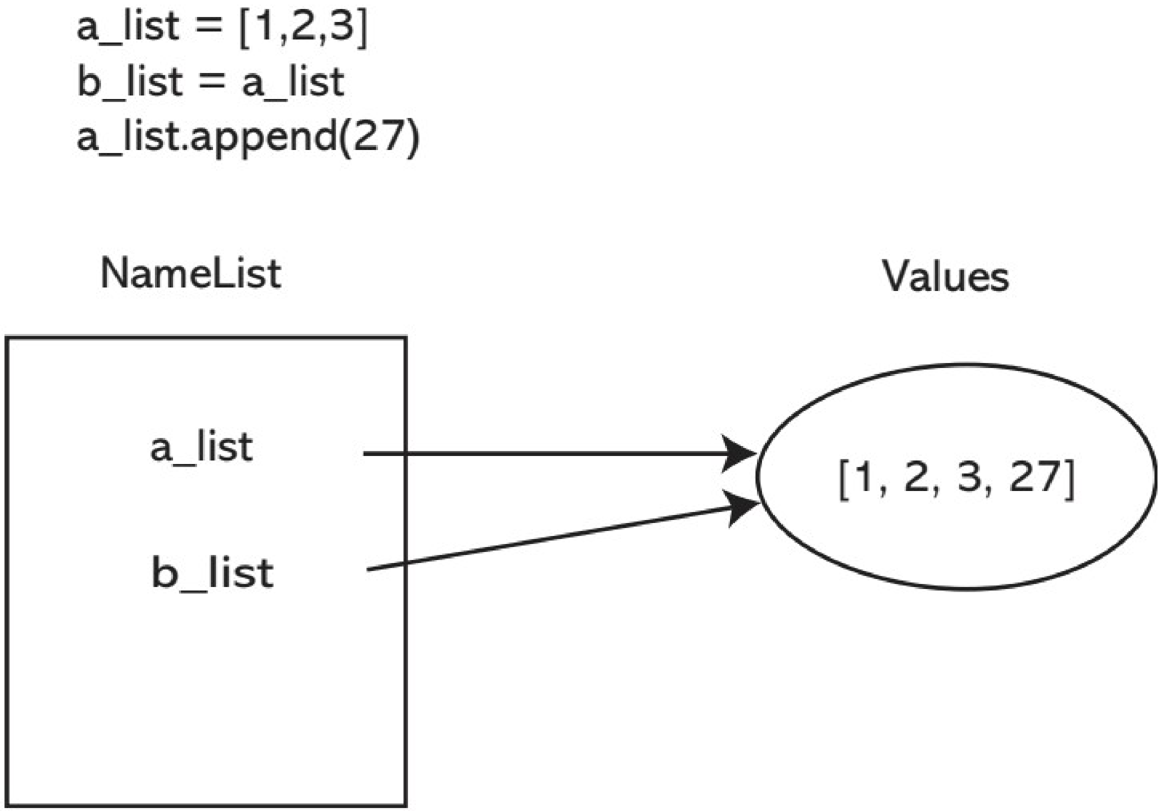


FIGURE 7.4 Namespace snapshot after assigning mutable objects.

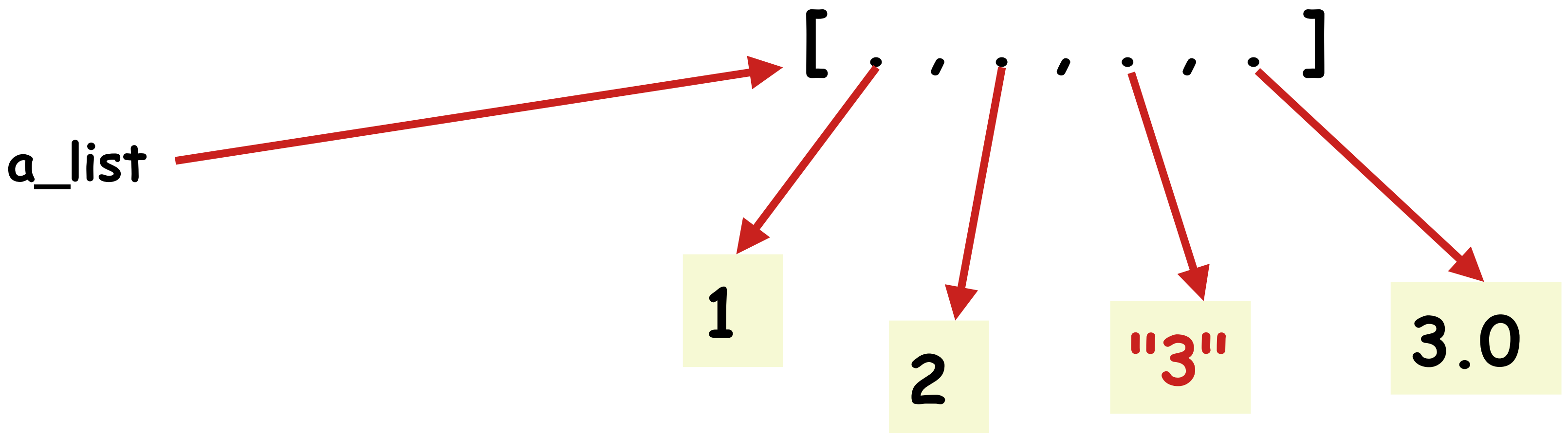


```
a_list = [1,2,3]
b_list = a_list
a_list.append(27)
```

FIGURE 7.5 Modification of shared, mutable objects.

Actually ...

```
a_list = [1, 2, "3", 3.0]
```



Masing-masing elemen di list adalah reference atau alamat!

```
a_list = [1,2,3]
a_list.append(a_list)
print(a_list)   →   [1, 2, 3, [...]]
```

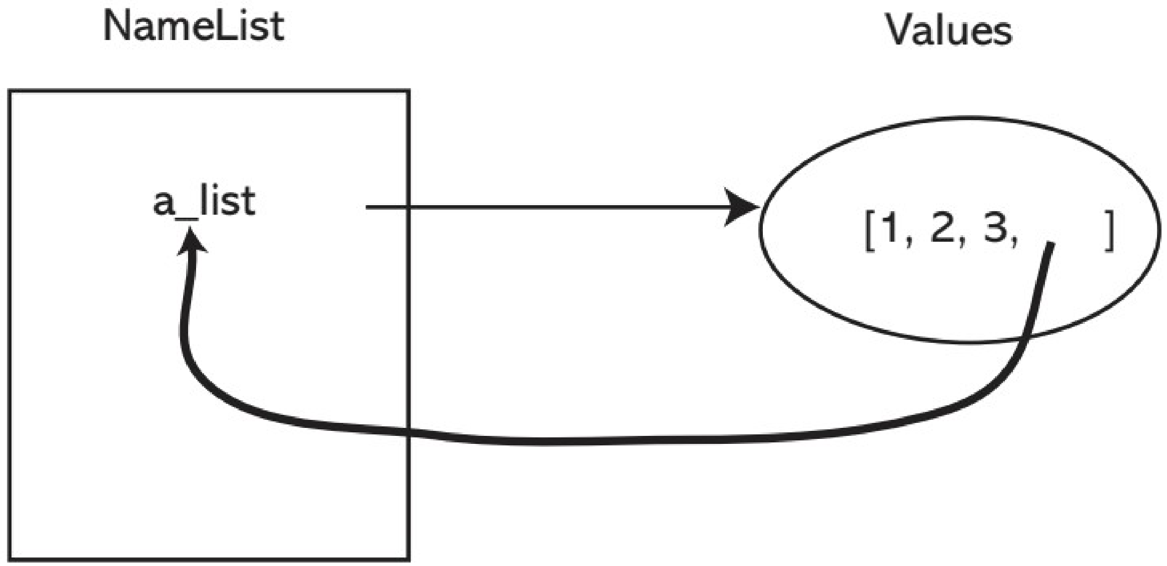
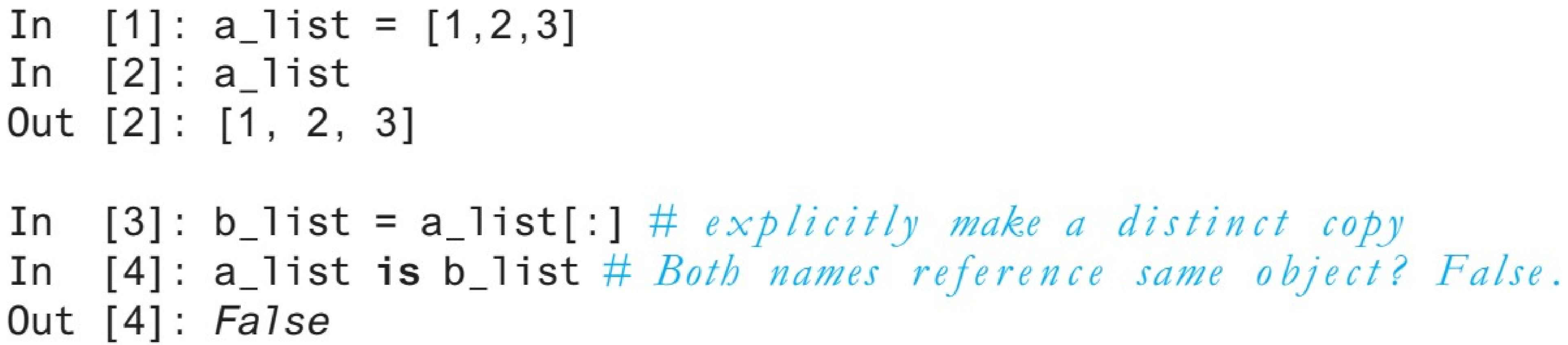


FIGURE 7.7 Self-referencing.

Copying

Bagaimana jika kita copy?



```
In  [1]: a_list = [1,2,3]
In  [2]: a_list
Out [2]: [1, 2, 3]

In  [3]: b_list = a_list[:] # explicitly make a distinct copy
In  [4]: a_list is b_list # Both names reference same object? False.
Out [4]: False
```

```
a_list = [1,2,3]
b_list = a_list[:]     # explicitly make a distinct copy
a_list.append(27)
```

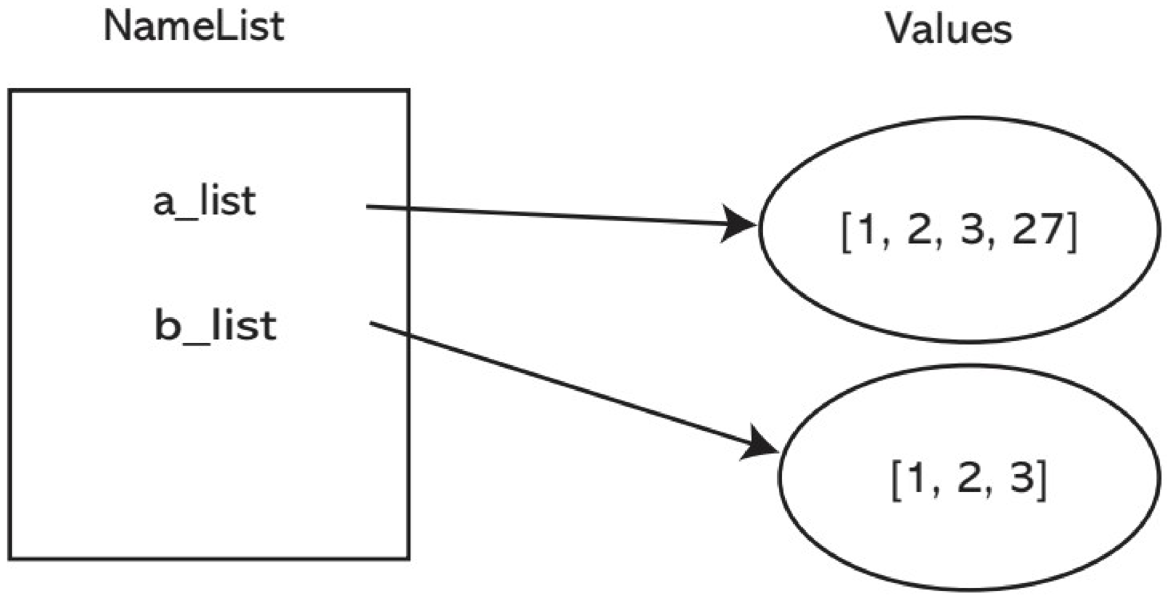


FIGURE 7.6 Making a distinct copy of a mutable object.

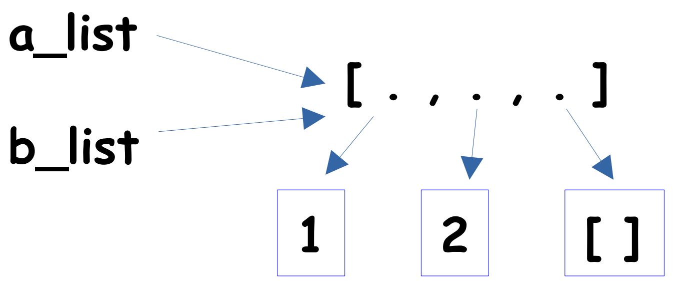
Sebenarnya ini adalah Shallow Copy!

Shallow Copy vs Deep Copy

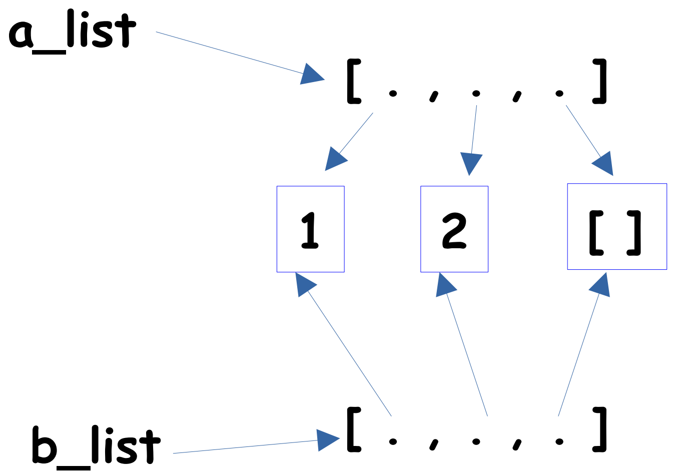
- Shallow copy: yang di-copy adalah alamat/references, bukan object-nya.

- Deep copy: yang di-copy adalah isi-nya (object-nya).

```python
a_list = [1, 2, []]
b_list = a_list
```



```python
a_list = [1, 2, []]
b_list = a_list[:]
```



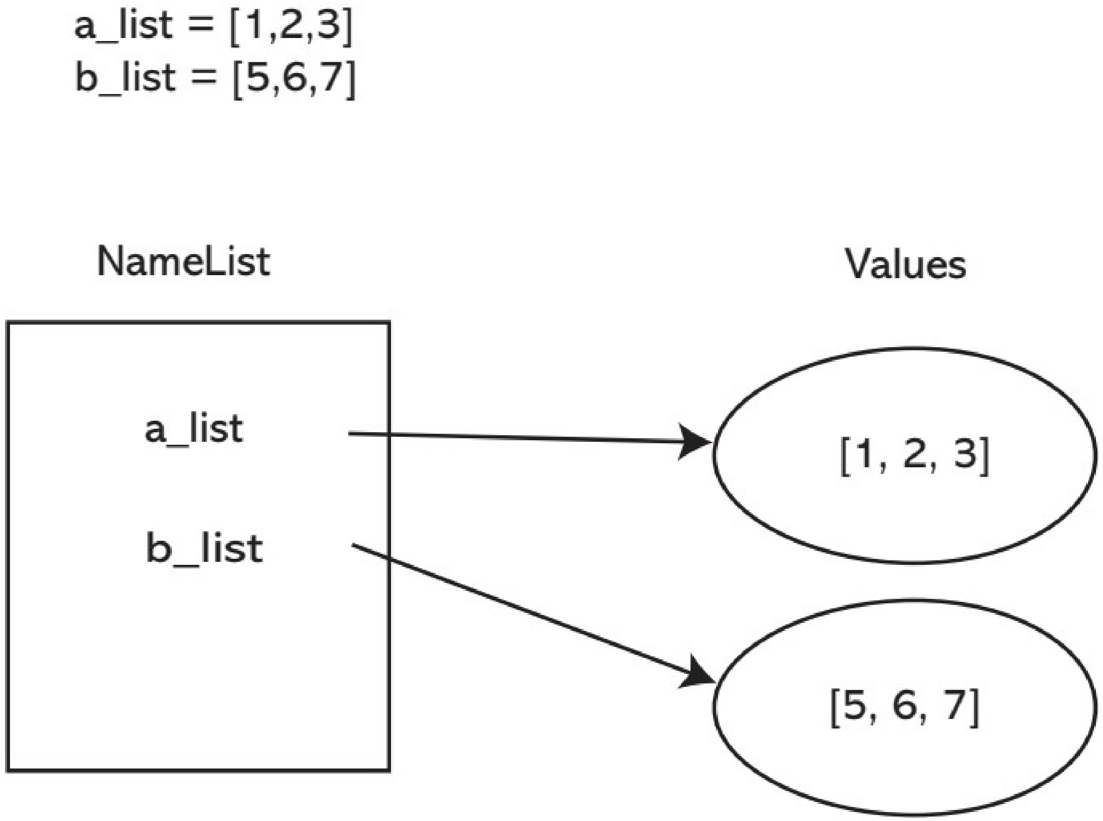


FIGURE 7.8 Simple lists before append.

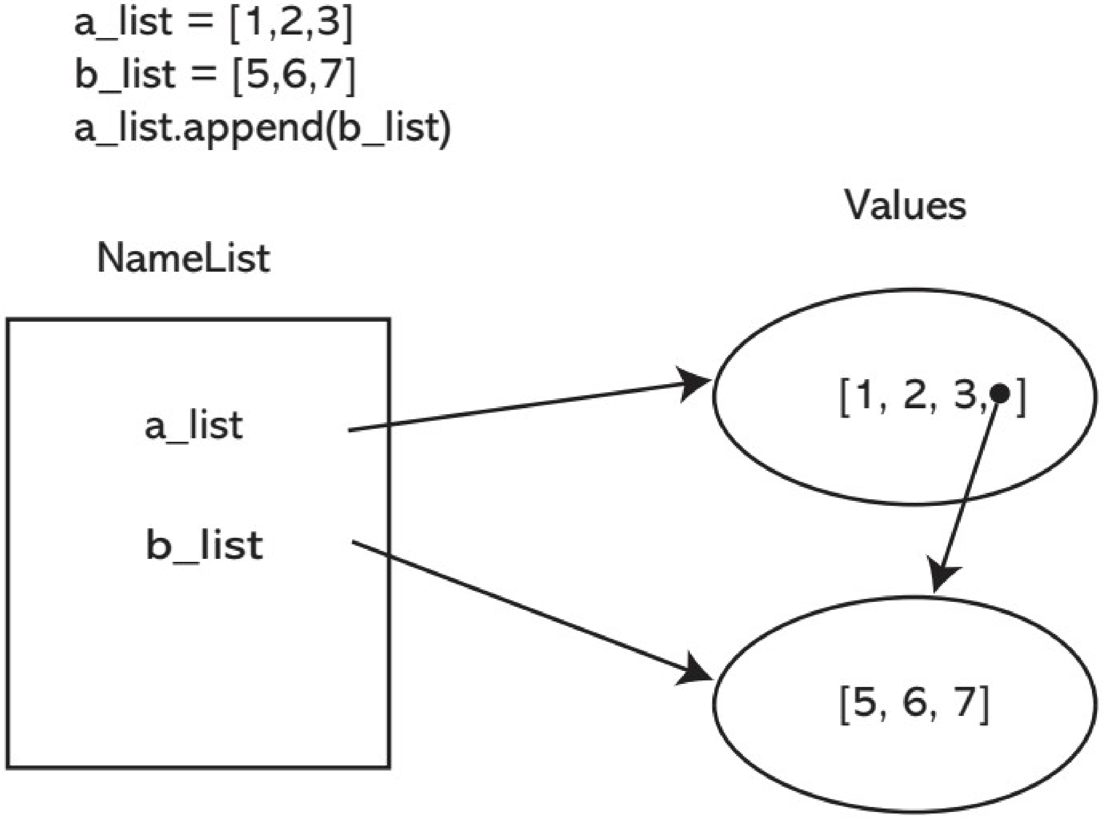


FIGURE 7.9 Lists after append.

```
a_list = [1,2,3]
b_list = [5,6,7]
a_list.append(b_list)
c_list = b_list
c_list[2] = 88
```

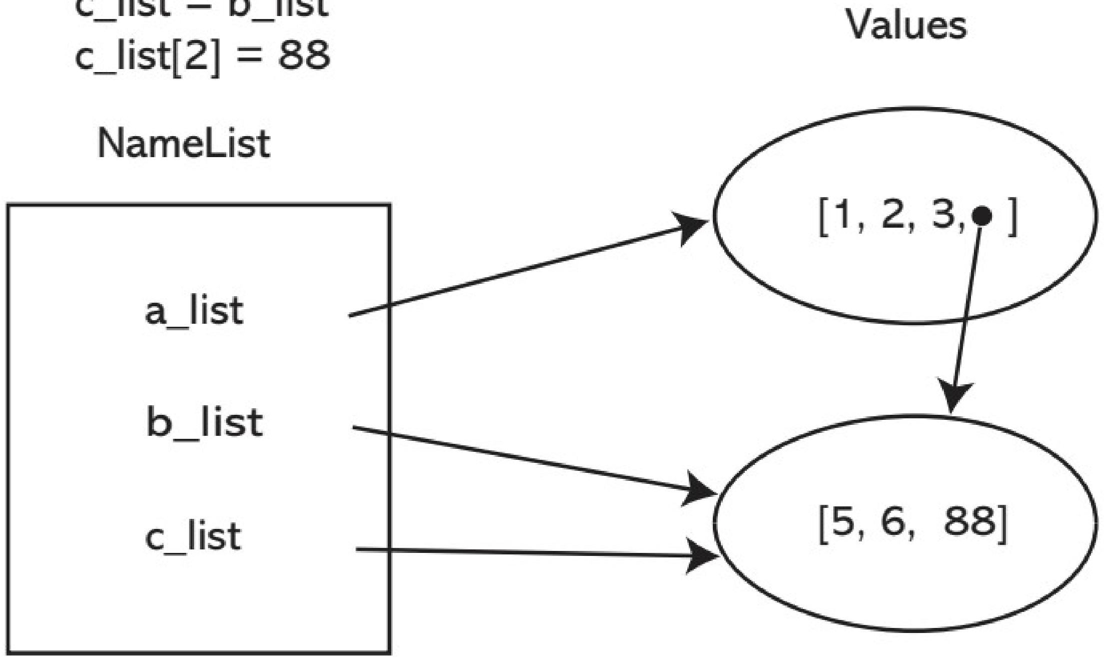
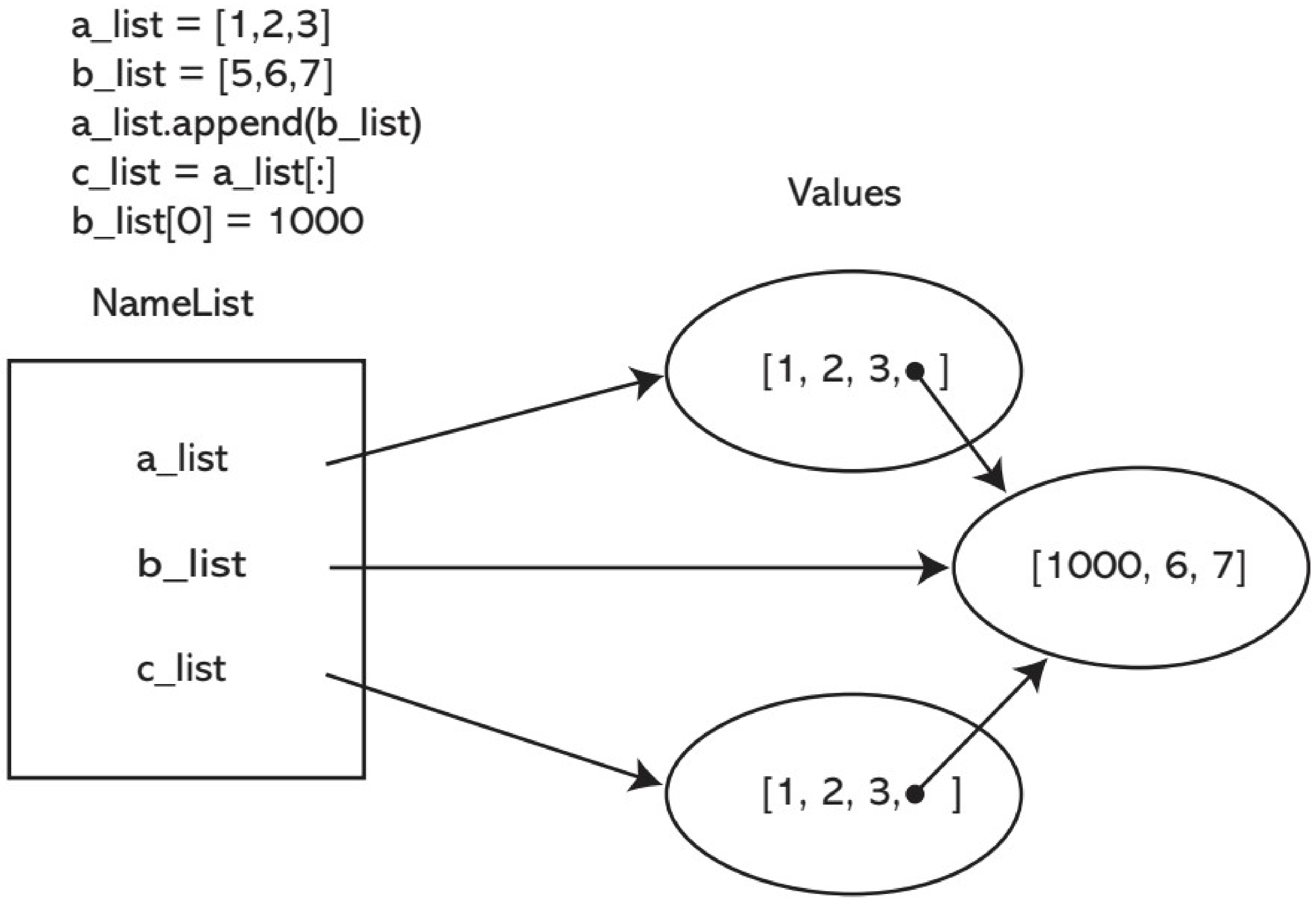


FIGURE 7.10 Final state of copying example.



```
a_list = [1,2,3]
b_list = [5,6,7]
a_list.append(b_list)
c_list = a_list[:]
b_list[0] = 1000
```

FIGURE 7.11 Effects of copy slice (a shallow copy).

```
a_list = [1,2,3]
b_list = [5,6,7]
a_list.append(b_list)
c_list = copy.deepcopy(a_list)
b_list[0] = 1000
```

Untuk menggunakannya, perlu `import copy`

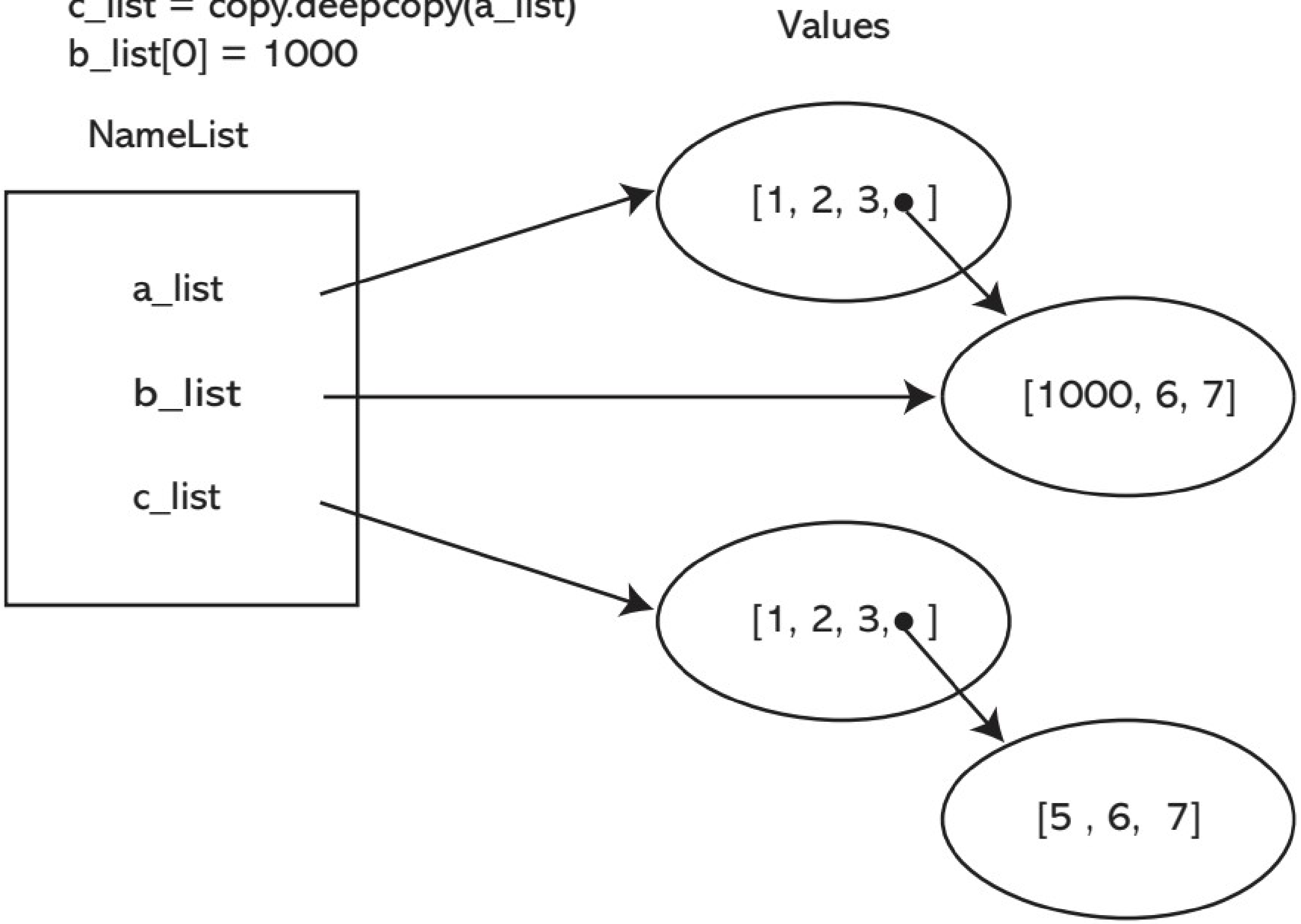
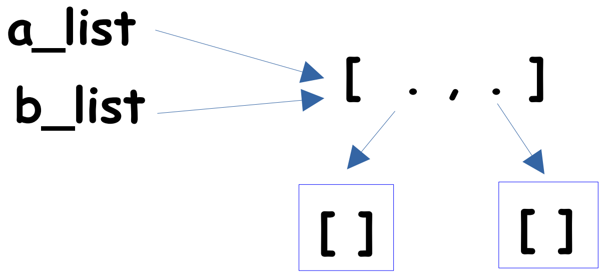
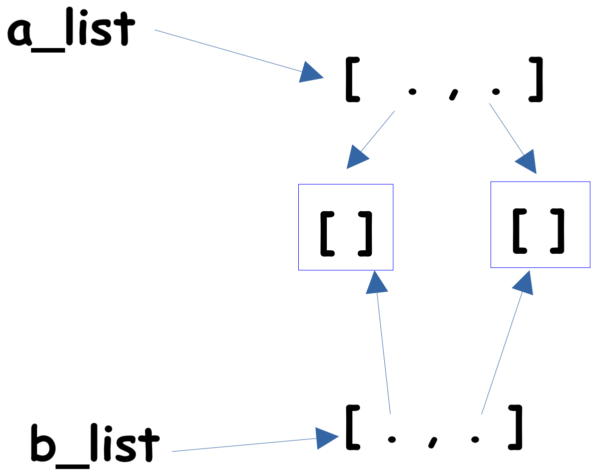


FIGURE 7.12 Using the copy module for a deep copy.

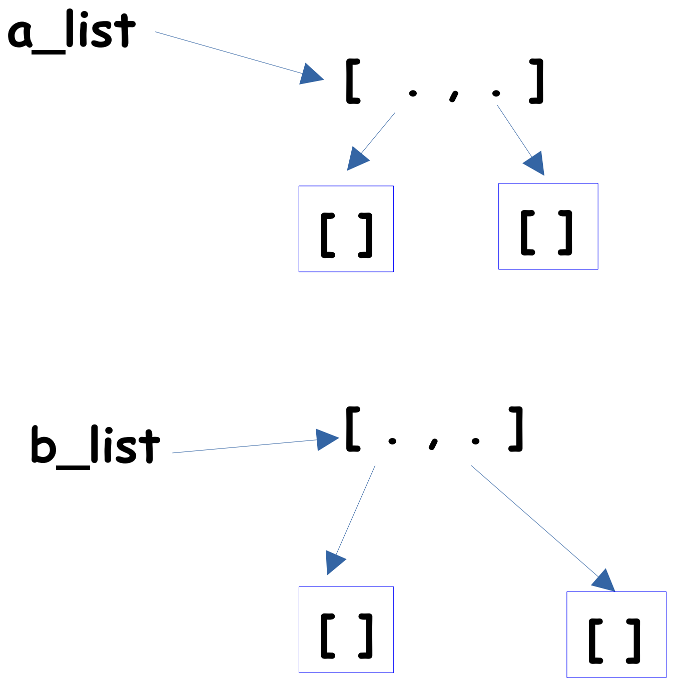
```python
a_list = [[], []]
b_list = a_list
```



```python
a_list = [[], []]
b_list = a_list[:]
```



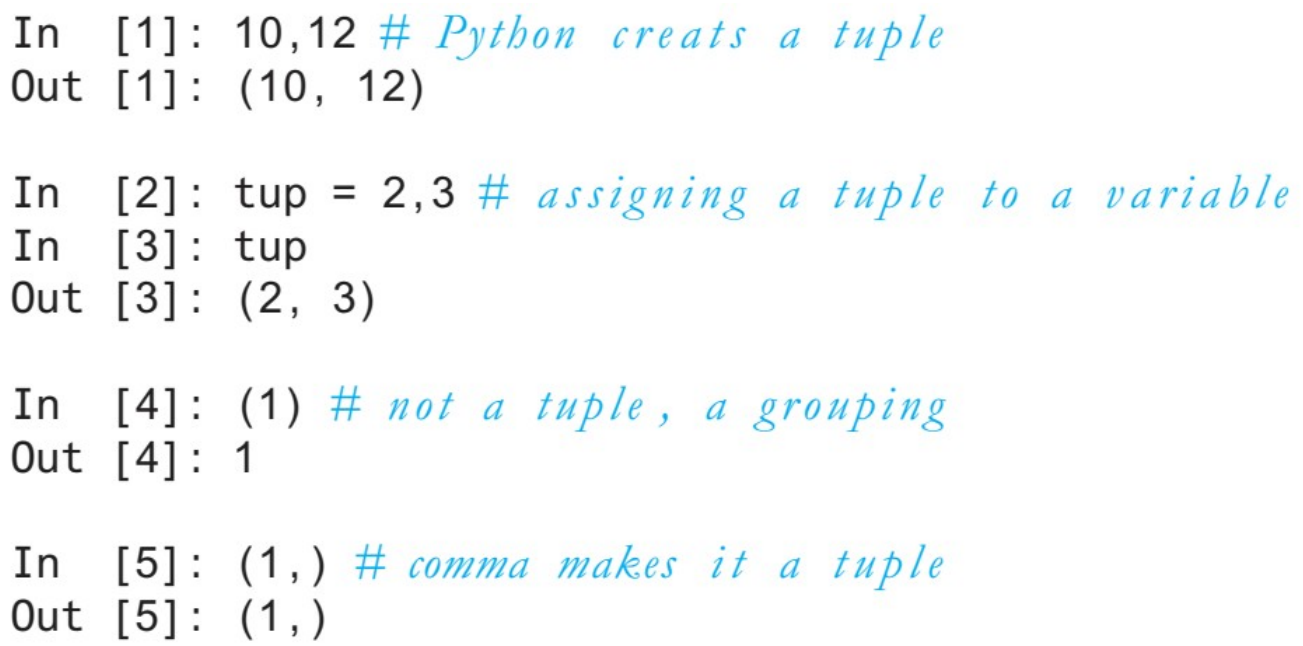
```python
import copy

a_list = [[], []]
b_list = copy.deepcopy(a_list)
```



Tuples

Tuples

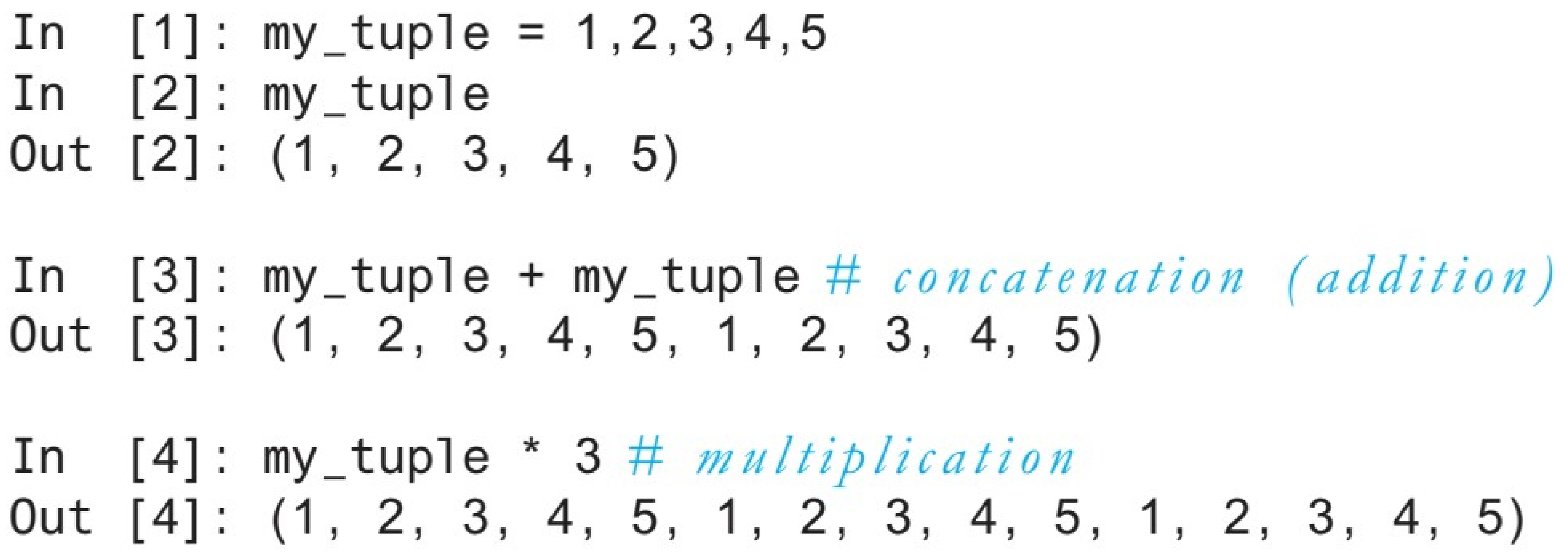
- Tuples are simply **immutable** lists
- Dibuat dengan notasi (…, …, …)



```
In  [1]: 10,12 # Python creats a tuple
Out [1]: (10, 12)

In  [2]: tup = 2,3 # assigning a tuple to a variable
In  [3]: tup
Out [3]: (2, 3)

In  [4]: (1) # not a tuple, a grouping
Out [4]: 1

In  [5]: (1,) # comma makes it a tuple
Out [5]: (1,)
```

Functions & Operators pada tuples

Hampir semua yang bisa diterapkan pada list bsia diterapkan pada tuple. Kecuali, yang mengubah nilai object.



```
In  [1]: my_tuple = 1,2,3,4,5
In  [2]: my_tuple
Out [2]: (1, 2, 3, 4, 5)

In  [3]: my_tuple + my_tuple # concatenation (addition)
Out [3]: (1, 2, 3, 4, 5, 1, 2, 3, 4, 5)

In  [4]: my_tuple * 3 # multiplication
Out [4]: (1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
```

```
In  [5]: my_tuple[1] # indexing
Out [5]: 2

In  [6]: my_tuple[:3] # slicing
Out [6]: (1, 2, 3)

In  [7]: my_tuple[1:3]
Out [7]: (2, 3)

In  [8]: my_tuple[-1]
Out [8]: 5

In  [9]: 2 in my_tuple # membership (in)
Out [9]: True
```

Tuple? motivation

Buatlah sebuah fungsi max_min yang menerima sebuah list lalu **mengembalikan indeks elemen terbesar dan indeks elemen terkecil sekaligus**!

```
def max_min_position(lst):
  ....
```

List of tuples

Misal, (follower, followee) direpresentasikan sebagai list of tuples:

```
[('alfan', 'rudi'), ('rudi', 'ani'), ('ani', 'alfan'), ...]
```

Implementasikan fungsi yang menerima data follower-followee dan sebuah nama akun; lalu mengembalikan daftar follower dari akun tersebut!

```
def who_follows(data, name):
  ....
```

```
>> data = [('ani', 'anto'), ('rio', 'anto'),('ani', 'rudi')]
>> who_follows(data, 'anto')
['ani', 'rio']
```

List Comprehension

- `[e for e in range(0, 10)]`
- `[e for e in range(0, 10) if e%2 == 0]`

- `Lst = [1,2,3,4,5]`
- `[e*e for e in lst]`