Laurea in Ingegneria Gestionale

Corso di Fondamenti di Informatica
A.A. 2017/2018

DIPARTIMENTO DI INGEGNERIA INFORMATICA AUTOMATICA E GESTIONALE ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Variables and memory usage in Python*

in Python variables work like tags: When you do an assignment in Python, it tags the value with the variable name.

```
a = 1
```

…and if you change the value of the variable, it just changes the tag to the new value in memory.

```
a = 2
```

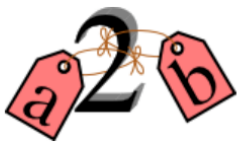
You dont need to do the housekeeping job of freeing the memory here. Python's Automatic Garbage Collection does it for you. When a value is without names/tags it is automatically removed from memory.

* Slide tratte da: http://foobarnbaz.com/2012/07/08/understanding-python-variables/

Variables and memory usage in Python

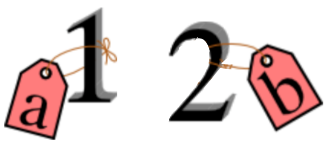
Assigning one variable to another makes a new tag bound to the same value as show below.

```
b = a
```



Now, if in the next instruction we assign a new value to `a`, `b` does not change

```
a = 1
```

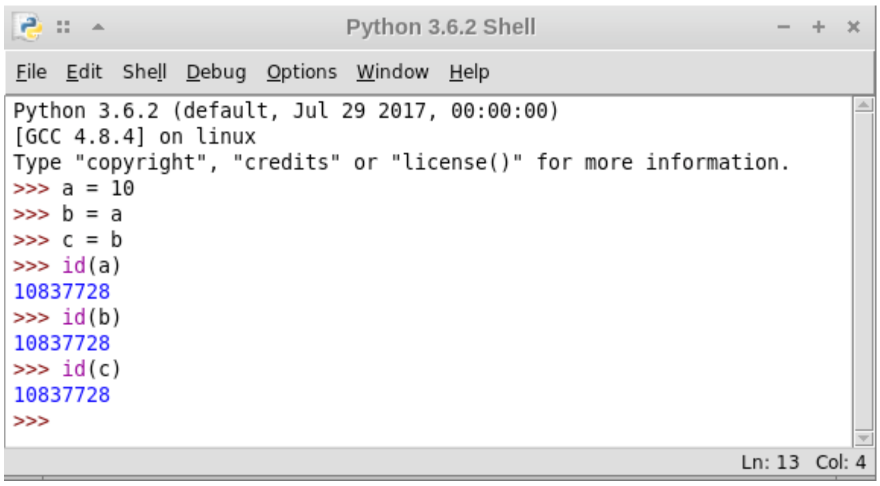


Variables and memory usage in Python

Consider the following code:

```
a = 10
b = a
c = a
```

As we have seen before, the value 10 has only one copy in memory and all the variables `a`, `b`, `c` refer to this memory location. This can be checked using he function `id()`, which returns a object memory address



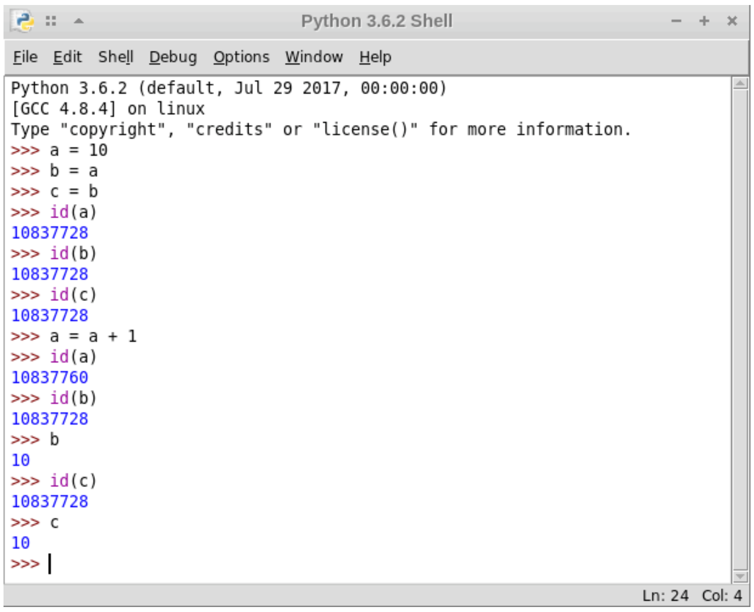
```
Python 3.6.2 (default, Jul 29 2017, 00:00:00)
[GCC 4.8.4] on linux
Type "copyright", "credits" or "license()" for more information.
>>> a = 10
>>> b = a
>>> c = b
>>> id(a)
10837728
>>> id(b)
10837728
>>> id(c)
10837728
>>>
```

Variables and memory usage in Python

If the next instruction is

    a = a + 1

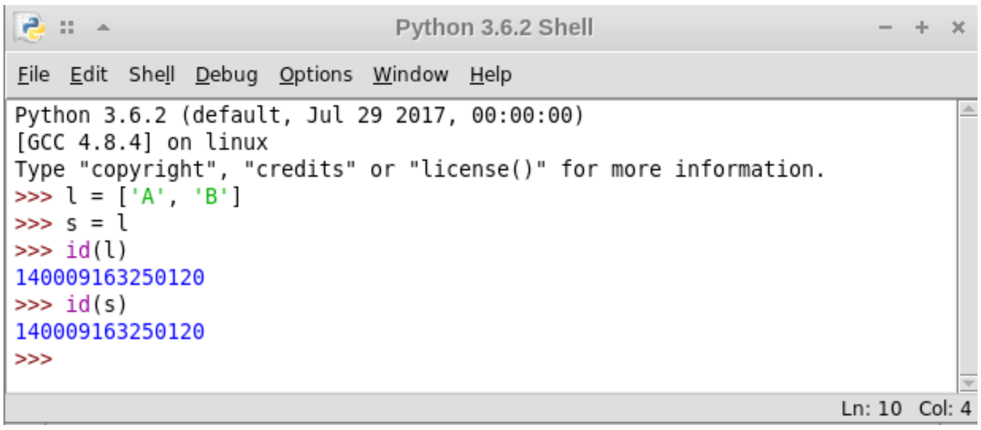
a new value `11` will be allocated in memory and `a` will point to this, whereas `b` and `c` will be unchanged (same memory address and value as before)



```
Python 3.6.2 (default, Jul 29 2017, 00:00:00)
[GCC 4.8.4] on linux
Type "copyright", "credits" or "license()" for more information.
>>> a = 10
>>> b = a
>>> c = b
>>> id(a)
10837728
>>> id(b)
10837728
>>> id(c)
10837728
>>> a = a + 1
>>> id(a)
10837760
>>> id(b)
10837728
>>> b
10
>>> id(c)
10837728
>>> c
10
>>> 
```

Variables and memory usage in Python

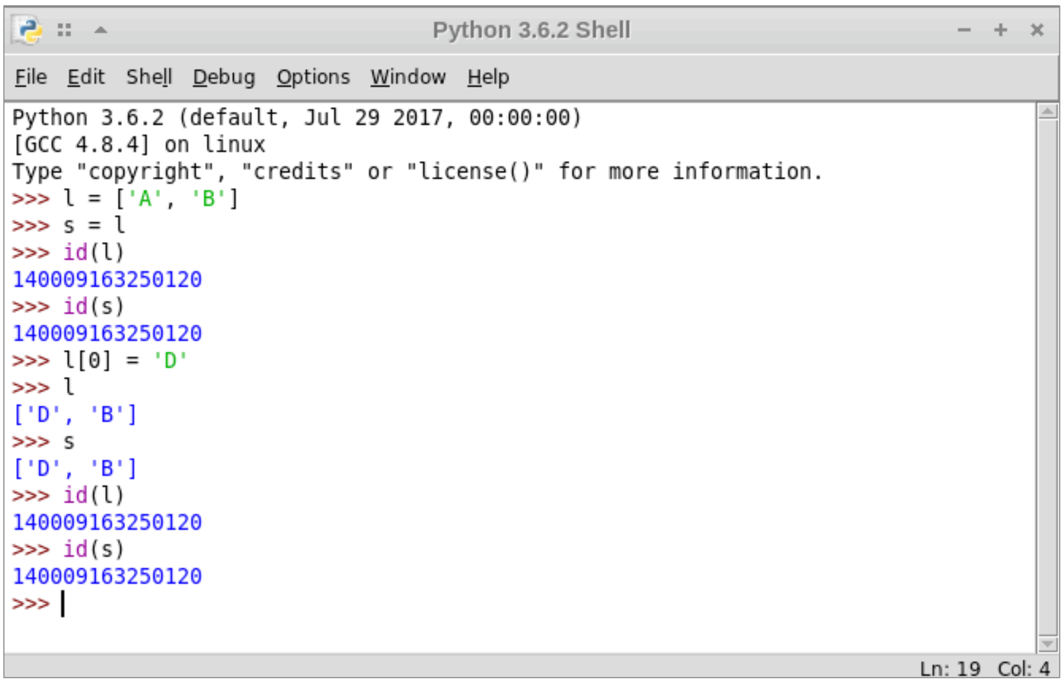
Now consider variables assigned to lists. We have the same situation as before.



```
Python 3.6.2 (default, Jul 29 2017, 00:00:00)
[GCC 4.8.4] on linux
Type "copyright", "credits" or "license()" for more information.
>>> l = ['A', 'B']
>>> s = l
>>> id(l)
140009163250120
>>> id(s)
140009163250120
>>> 
```

Variables and memory usage in Python

Let’s modify the content of the list `l`. Then also the content of `s` wil be changed



```
Python 3.6.2 (default, Jul 29 2017, 00:00:00)
[GCC 4.8.4] on linux
Type "copyright", "credits" or "license()" for more information.
>>> l = ['A', 'B']
>>> s = l
>>> id(l)
140009163250120
>>> id(s)
140009163250120
>>> l[0] = 'D'
>>> l
['D', 'B']
>>> s
['D', 'B']
>>> id(l)
140009163250120
>>> id(s)
140009163250120
>>> 
```