Chapter 6: The Dynamic Typing Interlude

Extractos de codigo del libro Learning Python 5th Ed. by Mark Lutz

OBJETOS INMUTABLES

In [2]:

```
a = 3
b = a       # b es 3
print("b es ", b)

a = 3
b = a
a = 'spam'  # b es 3
print("b es ", b)
```

```
b es  3
b es  3
```

Los objetos `integer` no son mutables:

In [3]:

```
a = 3
b = a
a = a + 2       # b es 3
print("b es ", b)
```

```
b es  3
```

OBJETOS MUTABLES

Listas

In [6]:

```
L = [2, 3, 4]       # A mutable object
M = L                       # Make a reference to the same object
L[0] = 24                   # An in-place change

print("L = ", L)                   # [24, 3, 4]
print("M = ", M)                   # [24, 3, 4]
```

```
L =  [24, 3, 4]
M =  [24, 3, 4]
```

Copia de objetos en vez de referencia:

```
In [7]: L = [2, 3, 4]
        M = L[:]                        # Make a copy of L (or list(L), copy.
        copy(L), etc.)
        L[0] = 24

        print("L = ", L)    # [24, 3, 4]

        print("M = ", M)    # [2, 3, 4]  M is not changed
```

```
        L =  [24, 3, 4]
        M =  [2, 3, 4]
```

Shared References and Equality

- Operador `==` igualdad de valores.
- Operador `is` identidad de objetos.

```
In [11]: L = [1, 2, 3]
         M = L                          # M and L reference the same object

         L == M                         # True: Same values => operador igual
         dad de valores
         L is M                         # True: operador identidad de objeto
         s: compara las referencias (los punteros)
```

```
Out[11]: True
```

```
In [15]: L = [1, 2, 3]
         M = [1, 2, 3]      # M and L reference different objects

         print(
             "L == M",
             L == M                          # True: Same values
         )

         print(
             "L is M",   # False: operador identidad de objetos: compara l
         as referencias (los punteros)
             L is M
         )
```

```
         L == M True
         L is M False
```

CACHE

Because small integers and strings are cached and reused, though, is tells us they reference the same single object.

In [ ]:

```
X = 42
Y = 42              # Should be two different objects

X  == Y         # True
X is Y              # True: Same object anyhow: caching at wor
k!!!!!!!!
```

Averiguar el numero de refencias a un objeto:

In [16]:

```
import sys
sys.getrefcount(1)              # number of pointers to this shar
ed piece of memory
```

Out[16]:

```
3514
```