

Objetos mutables e inmutables

Si necesitamos trabajar con horas, podemos hacerlo usando tres variables. Por ejemplo si son las 14:30:00 podemos asignar tres variables:

```
In [1]: h, m, s=14, 30, 0
```

Si ahora ha pasado un cierto periodo de tiempo *ds*, expresado en segundos nuestra hora será ¿?

Merece la pena tomarse la molestia de escribir una función para resolver el problema

```
In [2]: def inc_time(h,m,s,ds):  
        """  
        Function that increments the time h,m,s in ds seconds.  
        @type h,m,s:int  
        @rtype :(int,int,int)  
        @precondition: 0<=h<24 and 0<=m<60 and 0<=s<60  
        """  
        h1,m1,s1=h,m,s+ds  
        if s1>59:  
            m1=m+s1/60  
            s1=s1%60  
            if m1>59:  
                h1=h+m1/60  
                m1=m1%60  
                if h1>24:  
                    h1=h1%24  
        return h1,m1,s1
```

Así podemos actualizar nuestras variables para representar la hora al pasar 240 segundos.

```
In [3]: h,m,s=inc_time(h,m,s,240)  
        h,m,s
```

```
Out[3]: (14, 34, 0)
```

Usando listas podemos *unir* las cosas en una única variable.

The logo for Cartagena99 features the text 'Cartagena99' in a stylized, blue, serif font. The '99' is significantly larger and more prominent than the 'Cartagena' part. The text is set against a light blue background with a subtle gradient and a soft shadow effect.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

```
In [5]: def inc_time1(t,ds):
        """
        Function that increments the time represented by t in ds second
        s.
        This function changes the list t
        @type t:[int,int,int]
        @precondition: 0<=t[0]<24 and 0<=t[1]<60 and 0<=t[2]<60
        """
        t[2]=t[2]+ds
        if t[2]>59:
            t[1]=t[1]+t[2]/60
            t[2]=t[2]%60
            if t[1]>59:
                t[0]=t[0]+t[1]/60
                t[1]=t[1]%60
                if t[0]>24:
                    t[0]=t[0]%24
```

```
In [6]: inc_time1(time,240)
        time
```

```
Out[6]: [14, 34, 0]
```

Observa dos cosas: * `inc_time1` no tiene `return` * no hay asignación en la instrucción anterior. La función `inc_time1` se encarga de cambiar la lista a la que se refiere la variable `time` (ya hemos hecho cosas parecidas con funciones que transformaban imágenes).

¡Parece cómodo!, ¿podíamos haber hecho lo mismo con `inc_time`?

```
In [7]: def bad_inc_time(h,m,s,ds):
        """
        Function that increments the time h,m,s in ds seconds.
        @type h,m,s:int
        @precondition: 0<=h<24 and 0<=m<60 and 0<=s<60
        """
        s=s+ds
        if s>59:
            m=m+s/60
            s=s%60
            if m>59:
                h=h+m/60
                m=m%60
```

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

```
Out[8]: (14, 30, 0)
```

La respuesta es no: los nombres de los parámetros y las variables de las funciones tienen *ámbito local*.

La razón de conseguir el efecto en el caso de la lista es que **las listas son mutables**.

Podemos utilizar la siguiente herramienta on-line para visualizar lo que ocurre:

<http://pythontutor.com/visualize.html#mode=edit>

Otra interesante herramienta para visualizar las variables en una ejecución se encuentra en:

<http://www.uuhistle.org/tryout.php>

Hay otro tipo de situaciones en que hay que tener cuidado con el hecho de que las variables son referencias.

Para representar una matriz construimos una lista de listas:

```
In [9]: matrix=[[0,0,0],[0,0,0],[0,0,0]]
```

```
In [10]: matrix[0][2]=23
matrix
```

```
Out[10]: [[0, 0, 23], [0, 0, 0], [0, 0, 0]]
```

```
In [11]: def bad_zero_matrix(n,m):
          """
          Function that returns a zero matrix, with n filas y m columnas
          @type n,m:int
          @rtype: list of lists of int
          @precondition: n,m>0
          """
          return [[0]*n]*m
```

```
In [12]: m=bad_zero_matrix(3,4)
m
```

```
Out[12]: [[0, 0, 0], [0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

```
In [13]:
```

The logo for Cartagena99 features the text 'Cartagena99' in a stylized, blue, serif font. The '99' is significantly larger and more prominent than the word 'Cartagena'. The text is set against a light blue background with a subtle gradient and a soft shadow effect.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70

```
In [15]: def zero_matrix(n,m):
        """
        Function that returns a zero matrix, with n filas y m columnas
        @type n,m:int
        @rtype: list of lists of int
        @precondition: n,m>0
        """
        result=[]
        for i in xrange(n):
            result.append([0]*n)
        return result
```

```
In [16]: m1=zero_matrix(3,4)
```

```
In [17]: m1
```

```
Out[17]: [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

```
In [18]: m1[0][0]=1
```

```
In [19]: m1
```

```
Out[19]: [[1, 0, 0], [0, 0, 0], [0, 0, 0]]
```

```
In [19]:
```

The logo for Cartagena99 features the text 'Cartagena99' in a stylized, blue, serif font. The '99' is significantly larger and more prominent than the 'Cartagena' part. The text is set against a light blue background with a subtle gradient and a soft shadow effect.

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS
CALL OR WHATSAPP:689 45 44 70