Buscar si un elemento está en una lista no es difícil:

In [102]:
```
def esta(l,e):
    #l es una lista, e un elemento
    #devuelve true si e está en la lista
    result=False #Hasta que no lo vemos no está
    for x in l:
        if x==e:
            result=True
    return result
```

In [103]:
```
print esta([2,3,4],5)
print esta([3,2,6,7],10)
```

```
False
False
```

Aunque la versión anterior es mejorable, si ya lo hemos encontrado no hace falta seguir. Se impone usar un bucle `while`

In [104]:
```
def esta(l,e):
    #l es una lista, e un elemento
    #devuelve true si e está en la lista
    result=False #Hasta que no lo vemos no está
    i=0
    while i<len(l) and not result:
        if l[i]==e:
            result=True
        i=i+1
    return result
```

Y podemos aprovechar para devolver también la posición del elemento

In [105]:
```
def esta(l,e):
    #l es una lista, e un elemento
    #devuelve true y la posición del elemento si e está en la lista.
    #y False si no esta.
    result=False #Hasta que no lo vemos no está
    i=0
    while i<len(l) and not result:
        if l[i]==e:
            result=True
        i=i+1
    return result,i-1
```

In [106]:
```
esta([2,1,45,12,11],12)
```

Out[106]:
```
(True, 3)
```

Si sabemos que la lista está ordenada, podemos hacer las cosas mejor

In [107]:
```
def esta_ord(l,e):
    #l es una lista ordenada
    #devuelve true y la posición del elemento si e está en la lista.
    #y False y la posición que ocuparía el elemento si no esta.
    i=0
    while i<len(l) and l[i]<e:
        i=i+1
    return i<len(l) and l[i]==e,i
```

In [108]:
```
esta_ord([2,3,7,9,10],12)
```

Out[108]: (False, 5)

Aunque si la lista es grande y esta ordenada hay mejores formas de actúar. Para ver si el `10` está en la lista `l=[2,3,5,7,8,9,11,13,15,23,30,31,33,36,40,100]` podemos actúar así:

- el elemento central es `l[7]=13>10`
- luego el `10` estaría entre las posiciones 0 a 6, es decir miraremos si el `10` está en `l1=l[0:7]=[2,3,5,7,8,9,11]`
- el elemento central de `l1` es `l1[3]=7<10`
- luego el 10 estaría entre las posiciones 4 y 6 de l1, es decir miraremos si el `10` está en `l2=l1[4:7]=[8,9,11]`
- el elemento central de `l2`es `l2[1]=9<10`
- luego `10` estaría entre las posiciones 2 y 2 de `l2`, es decir miraremos si está en `l3=l2[2:3]=[11]`
- el elemeto central de `l3` es `l3[0]=11>10` y la siguiente lesta a considerar es `l4=l3[0:0]=[]` y, como es obvio 10 no está en `l4`

In [109]:
```
def esta_ord_bin_rec(l,e):
    if len(l)==0:
        result =False
    else:
        med=len(l)/2
        if l[med]==e:
            result=True
        elif l[med]>e:
            result=esta_ord_bin_rec(l[:med],e)
        else:
            result=esta_ord_bin_rec(l[med+1:],e)
    return result
```

In [110]:
```
esta_ord_bin_rec([2,3,5,7,8,9,11,13,15,23,30,31,33,36,40,100],9)
```

Out[110]: True

In [111]:
```
def pos_ord_bin_rec(l,e):
    if len(l)==0:
        result =False
        pos=0
    else:
        med=len(l)/2
        if l[med]==e:
            result=True
            pos=med
        elif l[med]>e:
            result,pos=pos_ord_bin_rec(l[:med],e)
        else:
            result,pos=pos_ord_bin_rec(l[med+1:],e)
            pos=pos+med+1
    return result,pos
```

In [112]:
```
pos_ord_bin_rec([2,3,5,7,8,9,11,13,15,23,30,31,33,36,40,100],36)
```

Out[112]: (True, 13)

No controlamos si nos devuelve la primera,segunda ... aparición del elemento en la lista

In [113]:
```
pos_ord_bin_rec([2,3,5,7,8,9,11,13,15,23,30,31,33,36,36,36,36,40,100],36)
```

Out[113]: (True, 14)

In [114]:
```
def pos_ord_bin_rec1(l,e):
    if len(l)==0:
        result =False
        pos=0
    else:
        med=len(l)/2
        if l[med]>=e:
            result,pos=pos_ord_bin_rec1(l[:med],e)
            result=(l[med]==e or result)
        else:
            result,pos=pos_ord_bin_rec1(l[med+1:],e)
            pos=pos+med+1
    return result,pos
```

In [115]:
```
pos_ord_bin_rec1([2,3,5,7,8,9,11,13,15,23,30,31,33,36,36,36,36,40,100],36)
```

Out[115]: (True, 13)

In [116]:
```
def pos_ord_bin_rec2(l,i,j,e):
    if i>j:
        pos=i
        result=False
    else:
        med=(i+j)/2
        if l[med]>=e:
            result,pos=pos_ord_bin_rec2(l,i,med-1,e)
            result=(l[med]==e or result)
        else:
            result,pos=pos_ord_bin_rec2(l,med+1,j,e)
    return result,pos
```

In [117]:
```
l=[2,3,5,7,8,9,11,13,15,23,30,31,33,36,36,36,36,40,100]
print pos_ord_bin_rec2(l,0,len(l)-1,36)
print pos_ord_bin_rec2(l,0,len(l)-1,200)
```

```
(True, 13)
(False, 19)
```

In [118]:
```
len(l)
```

Out[118]:
```
19
```

In [119]:
```
def pos_ord_bin_iter(l,e):
    i=0
    j=len(l)-1
    while i<=j:
        med=(i+j)/2
        if l[med]>=e:
            j=med-1
        else:
            i=med+1
    return (i<len(l) and l[i]==e),i
```

In [120]:
```
print pos_ord_bin_rec2(l,0,len(l)-1,36)
print pos_ord_bin_rec2(l,0,len(l)-1,200)
```

```
(True, 13)
(False, 19)
```

In [17]:
```
def ordena_sel(l):
    #ordena la lista l
    def menor_el(l,j):
        #l lista, j<len(l)
        #devuelve el índice del menor elemento de l desde j hasta el final
        menor=l[j]
        ind=j
        for k in range(j+1,len(l)):
            if l[k]<menor:
                menor=l[k]
                ind=k
        return ind
    for i in range(len(l)-1):
        k=menor_el(l,i)
        l[i],l[k]=l[k],l[i]
```

In [122]:
```
l=[3,2,1,45,10,45,123]
ordena_sel(l)
```

In [123]:
```
l
```

Out[123]: [1, 2, 3, 10, 45, 45, 123]

In [18]:
```
def ordena_inser(l):
    def pos_bin(l,inic,fin,e):
        i=inic
        j=fin
        while i<=j:
            med=(i+j)/2
            if l[med]>=e:
                j=med-1
            else:
                i=med+1
        return i

    for i in range(1,len(l)):
        j=pos_bin(l,0,i-1,l[i])
        el=l[i]
        for k in range(i,j,-1):
            l[k]=l[k-1]
        l[j]=el
```

In [125]:
```
l=[3,2,1,45,10,45,123]
ordena_inser(l)
```

In [126]:
```
l
```

Out[126]: [1, 2, 3, 10, 45, 45, 123]

In [19]:
```
def ord_burbuja(l):
    cambiado=True
    k=0
    while k<len(l) and cambiado:
        cambiado=False
        for i in range(len(l)-k-1):
            if l[i+1]<l[i]:
                l[i],l[i+1]=l[i+1],l[i]
                cambiado=True
```

In [7]:
```
l=[3,2,1,45,10,45,123]
ord_burbuja(l)
```

In [8]:
```
l
```

Out[8]:
```
[1, 2, 3, 10, 45, 45, 123]
```

In [20]:
```
def mezcla(l,i,j,med):
    #l ordenada en i...med-1 y med...j
    #ordena l entre i..j
    aux=[]
    in1=i
    in2=med
    while in1<=med-1 and in2<=j:
        if l[in1]<l[in2]:
            aux.append(l[in1])
            in1=in1+1
        else:
            aux.append(l[in2])
            in2=in2+1
    for k in range(in1,med):
        aux.append(l[k])
    for k in range(in2,j+1):
        aux.append(l[k])
    for k in range(len(aux)):
        l[i+k]=aux[k]
```

In [17]:
```
l=[1,5,8,10,4,7,9,11,12]
mezcla(l,0,8,4)
```

In [18]:
```
l
```

Out[18]:
```
[1, 4, 5, 7, 8, 9, 10, 11, 12]
```

In [21]:
```
def ordena_mezcla(l,i,j):
    if j>i:
        medio=(i+j)/2+1
        ordena_mezcla(l,i,medio-1)
        ordena_mezcla(l,medio,j)
        mezcla(l,i,j,medio)
```

In [26]:
```
l=[8,5,2,10,1,3]
ordena_mezcla(l,0,len(l)-1)
```

In [27]:
```
l
```

Out[27]: [1, 2, 3, 5, 8, 10]

In [22]:
```
def pivotar(l,i,j):
    #recoloca los elementos de l entre i..j
    #coloca l[i] en la posicion p
    #los menores que l[i] van en las posiciones i..p-1, y los mayores entre p+1..j
    e=l[i]
    p=i
    b=j
    #lo que sigue se cumple antes de entrar y al terminar cada paso del bucle:
    #l[p]=e
    #para i<=k<p l[k]<=e
    #para b<k<=j l[k]>e
    while b>p:
        if l[p+1]>e:
            l[p+1],l[b]=l[b],l[p+1]
            b=b-1
        elif l[p+1]<e:
             l[p],l[p+1]=l[p+1],l[p]
             p=p+1
        else:
            p=p+1
    return p
```

In [3]:
```
l=[7,2,8,9,1,1,7]
pivotar(l,0,len(l)-1)
```

Out[3]: 4

In [4]:
```
l
```

Out[4]: [2, 7, 1, 1, 7, 9, 8]

In [5]:
```
pivotar(l,0,3)
```

Out[5]: 2

In [6]:
```
l
```

Out[6]: [1, 1, 2, 7, 7, 9, 8]

In [7]:
```
pivotar(l,5,len(l)-1)
```

Out[7]: 6

In [8]:
```
l
```

Out[8]: [1, 1, 2, 7, 7, 8, 9]

In [23]:
```
def quicksort(l,i,j):
    if i<j:
        p=pivotar(l,i,j)
        quicksort(l,i,p-1)
        quicksort(l,p+1,j)
```

In [12]:
```
l=[7,4,7,10,1,2,3,9,3,5]
quicksort(l,0,len(l)-1)
```

In [13]:
```
l
```

Out[13]:
```
[1, 2, 3, 3, 4, 5, 7, 7, 9, 10]
```

In [32]:
```
import random
import time
def aleatoria(n,max):
    result=[0]*n
    for i in range(n):
        result[i]=random.randint(0,max)
    return result
l=aleatoria(100000,100000)
l1=l[:]
t0=time.time()
ordena_sel(l1)
print "seleccion:",time.time()-t0
l1=l[:]
t0=time.time()
ordena_inser(l1)
print "inserccion:",time.time()-t0
l1=l[:]
t0=time.time()
ord_burbuja(l1)
print "burbuja:",time.time()-t0
l1=l[:]
t0=time.time()
ordena_mezcla(l1,0,len(l1)-1)
print "mezcla:",time.time()-t0
l1=l[:]
t0=time.time()
quicksort(l1,0,len(l1)-1)
print "quicksort:",time.time()-t0
```

```
seleccion: 358.783027887
inserccion: 303.308477879
burbuja: 1609.74829292
mezcla: 0.77033996582
quicksort: 0.661063909531
```

In [ ]:
```

```