Just some rubbish, ignore it

In [1]:

```
%cat algotrial.py
```

```
def f():
    return "hellgddddddggggo"
```

In [2]:

```
import algotrial

print algotrial.f()
```

```
hellgddddddggggo
```

In [3]:

```
x = ['a']
x.insert(3, 'b')
x
```

Out[3]:

```
['a', 'b']
```

In [4]:

```
x
```

Out[4]:

```
['a', 'b']
```

In [5]:

```
y = []
y.insert(0, 'a')
y
```

Out[5]:

```
['a']
```

In [6]:

```
z = ['a','c']
z.insert(1,'b')
z
```

Out[6]:

```
['a', 'b', 'c']
```

In [7]:

```
z = ['a','c'].reverse()
```

In [8]:

```
p = ['a', 'b']
p.pop()
```

Out[8]:

```
'b'
```

In [9]:

```

```

In [9]:

```
%%HTML

<p class="algolab-warn">
WARNING: <code>delete <i>t</i></code> is a command that asks the environment to phys
ically deallocate
    the memory occupied by object <i>t</i>, and it is different from the method <cod
e>delete(TREE t)</code>
    (notice the parenthesis) which is something defined by the user to perform more
sophisticated cleaning
    procedures (in this case, going through connected useless nodes and deallocate t
hem one by one)!
</p>


<p class="algolab-important">
IMPORTANT: While coding to Python, you can often ignore the pseudo code command <cod
e>delete</code>
        like in <code>delete <i>t</i></code> !
        The reason is commands like <code>delete</code> are mostly thought for langu
ages where you
        have to manually deallocate memory once you don't need it anymore (like in <
i>C</i>). Luckily for us, Python
        manages memory for us  - that is, now and then Python garbage collector runs
 and
        whenever an object is not referenced by any pointer, it gets automatically r
emoved.
</p>


<p class="algolab-question">
QUESTION: Given the above, do you really need to implement the <code>delete(TREE t)<
/code> method ?</p>
</p>
```

WARNING: `delete` *t* is a command that asks the environment to physically deallocate the memory occupied by object *t*, and it is different from the method `delete(TREE t)` (notice the parenthesis) which is something defined by the user to perform more sophisticated cleaning procedures (in this case, going through connected useless nodes and deallocate them one by one)!

IMPORTANT: While coding to Python, you can often ignore the pseudo code command `delete` like in `delete` *t* ! The reason is commands like `delete` are mostly thought for languages where you have to manually deallocate memory once you don't need it anymore (like in *C*). Luckily for us, Python manages memory for us - that is, now and then Python garbage collector runs and whenever an object is not referenced by any pointer, it gets automatically removed.

QUESTION: Given the above, do you really need to implement the `delete(TREE t)` method ?