在 Lua 中实现面向对象

在 pil 中，lua 的作者推荐了一种方案来实现 OO，比较简洁，但是我依然觉得有些繁琐。

这里给出一种更漂亮一点的解决方案：为了贴代码和修改方便，我把它贴在了 wiki 上。

Lua 中实现面向对象

在这个方案中，只定义了一个函数 class(super) ，用这个函数，我们就可以方便的在 lua 中定义类：

```
base_type=class()       -- 定义一个基类 base_type

function base_type:ctor(x)  -- 定义 base_type 的构造函数
    print("base_type ctor")
    self.x=x
end

function base_type:print_x()    -- 定义一个成员函数 base_type:print_x
    print(self.x)
end

function base_type:hello()  -- 定义另一个成员函数 base_type:hello
    print("hello base_type")
end
```

以上是基本的 class 定义的语法，完全兼容 lua 的编程习惯。我增加了一个叫做 ctor 的词，作为构造函数的名字。

下面看看怎样继承： test=class(basetype) -- 定义一个类 test 继承于 basetype

```
function test:ctor()    -- 定义 test 的构造函数
    print("test ctor")
end

function test:hello()   -- 重载 base_type:hello 为 test:hello
    print("hello test")
end
```

现在可以试一下了：

`a=test.new(1)` -- 输出两行，`base_type ctor` 和 `test ctor` 。这个对象被正确的构造了。

`a:print_x()` -- 输出 `1` ，这个是基类 `base_type` 中的成员函数。

`a:hello()` -- 输出 `hello test` ，这个函数被重载了。

其实，实现多重继承也并不复杂，这里就不再展开了。更有意义的扩展可能是增加一个 dtor :)

~~ps. 这里用了点小技巧，将 self 绑定到 closure 上，所以并不使用 a:hello 而是直接用 a.hello 调用成员函数。这个技巧并不非常有用，从效率角度上说，还是不用为好。~~