Nool Usage (version 2.0.1)

First of all, i should mention that this it not the real version. It is a kind of beta version for testing purposes. This version has many problems with scope declarations and recursivity.

But we are working on it. There are two ways of using it :

- In an interactive way
- Or calling the executable with a file, where you have your code.

The interactive way is not complete yet, of course. But you can perform some mathematical operations or boolean ones.

When using script file you can create classes or functions. Please do not create complicate ones for the momment.

Syntax

The syntax is similar to python in a certain way.

For function declarations :

```
def functon_name(parameters) :
    body
```

example :

```
def foo() :
    print(self.name())
```

For class declarations :

```
Class class_name(parent_class_name) :
    Body
```

Example :

```
class Person(Object) :
    x = 'jean'
    def name() :
        return x
```

Nb : In fact everything is like in python except for certain things. Semicolons can be add at the end of every statement if you want.

Predefined functions

There are couple of define functions :

- ➔ print : for printing. Example : print(x,y,z)
- ➔ pprint : same as print with a difference in the way of printing.
- ➔ name : gives the name of the caller.

Example :

name() ;// will print the name of the current package

obj.name() ;//will print the name of obj, you can redefine it if you want when building the class of obj.

- ➔ class : gives the class of an object
- ➔ package : gives the package name where an object was build or created
- ➔ read : capture a value for basic types.
  Example :
  X = read('enter a name : ') ;

Nb : There are many other things but i don't have time now to write all the documentation.

Please excuse me for the moment. We are working on a new version with more stuff and this one will come with the documentation.