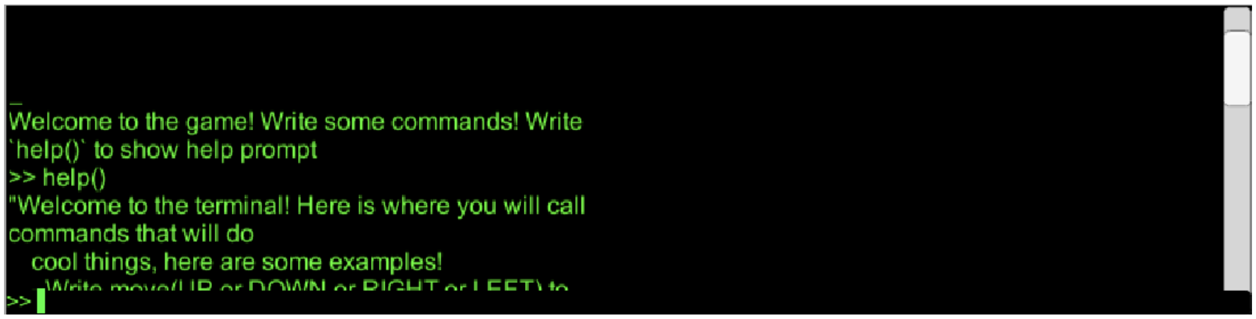
Terminal Feature

Created	@Jan 3, 2021 8:36 PM
Created By	Gabriel Villalonga
Last Edited Time	@Jan 3, 2021 9:31 PM
Participants	
Type	

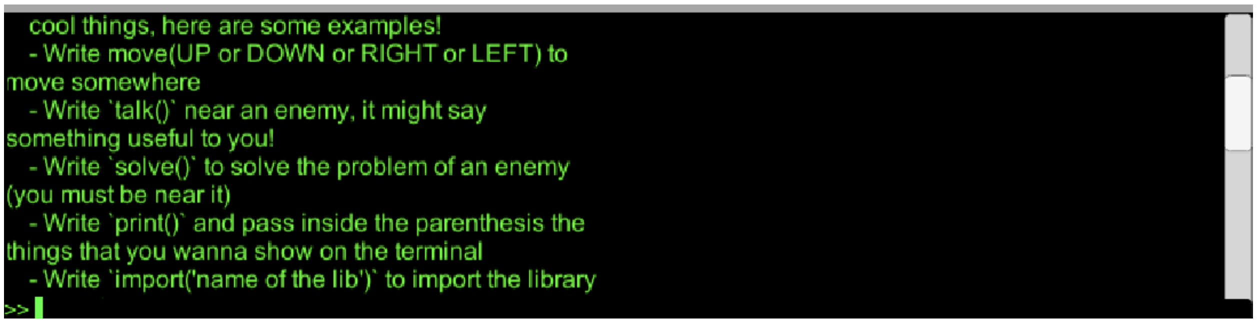
This is a showcase of the terminal inside the game and multiple language features.

When you open the terminal it will prompt you information and to write the method `help()` to show a big starter pack text



```
_
Welcome to the game! Write some commands! Write
`help()` to show help prompt
>> help()
"Welcome to the terminal! Here is where you will call
commands that will do
   cool things, here are some examples!
>>
```

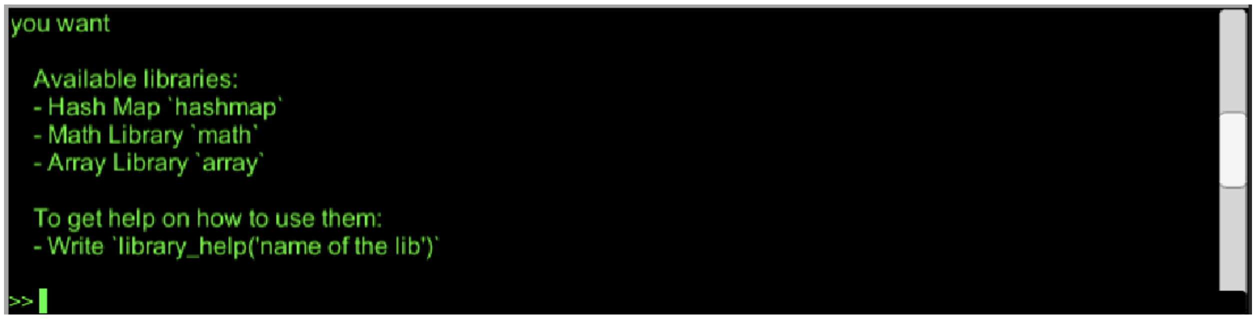
When you scroll through the terminal you will find this:



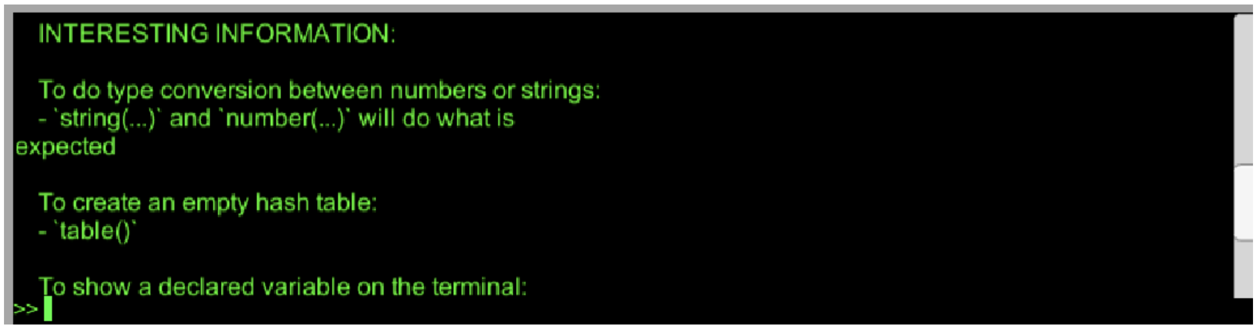
```
   cool things, here are some examples!
   - Write move(UP or DOWN or RIGHT or LEFT) to
move somewhere
   - Write `talk()` near an enemy, it might say
something useful to you!
   - Write `solve()` to solve the problem of an enemy
(you must be near it)
   - Write `print()` and pass inside the parenthesis the
things that you wanna show on the terminal
   - Write `import('name of the lib')` to import the library
>>
```

So the player will know how to do basic stuff, and below there are more instructions that are more advanced like importing libraries and getting more information about them (so veteran programmers can advance through the game faster)

Or also a way to read tutorials for the programming language, but do not worry, these tutorials will also come up when the player advances through the game, so if it is a beginner user, it will be able to go through the game with a pace that is good for learning.



```
you want

   Available libraries:
   - Hash Map `hashmap`
   - Math Library `math`
   - Array Library `array`

   To get help on how to use them:
   - Write `library_help('name of the lib')`

>>
```



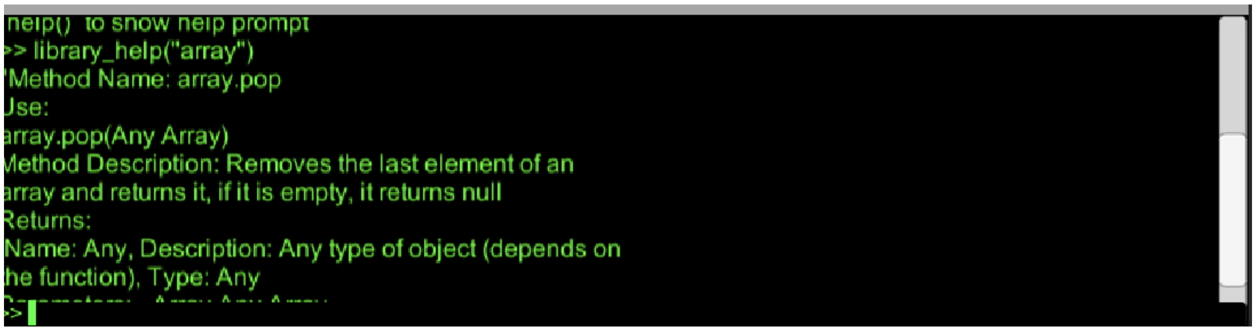
```
   INTERESTING INFORMATION:

   To do type conversion between numbers or strings:
   - `string(...)` and `number(...)` will do what is
expected

   To create an empty hash table:
   - `table()`

   To show a declared variable on the terminal:
>>
```

Etc.

When you write

```
library_help("array");
```

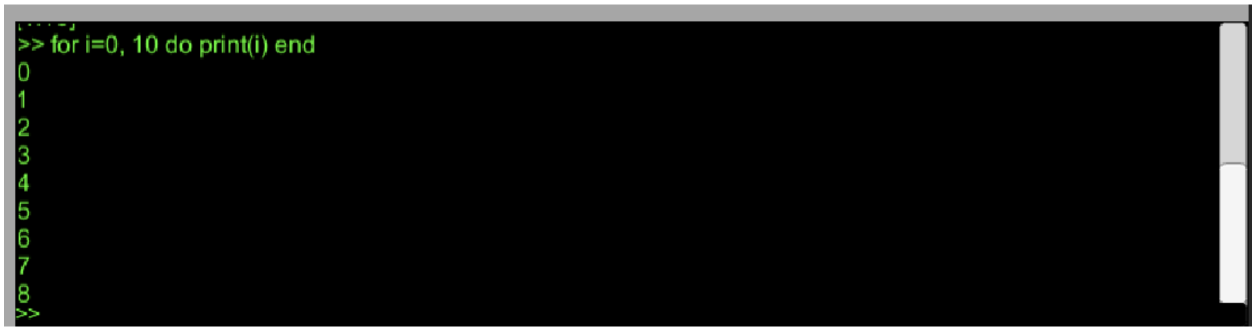
you will get a message like this, showing the available documentation added to that library:



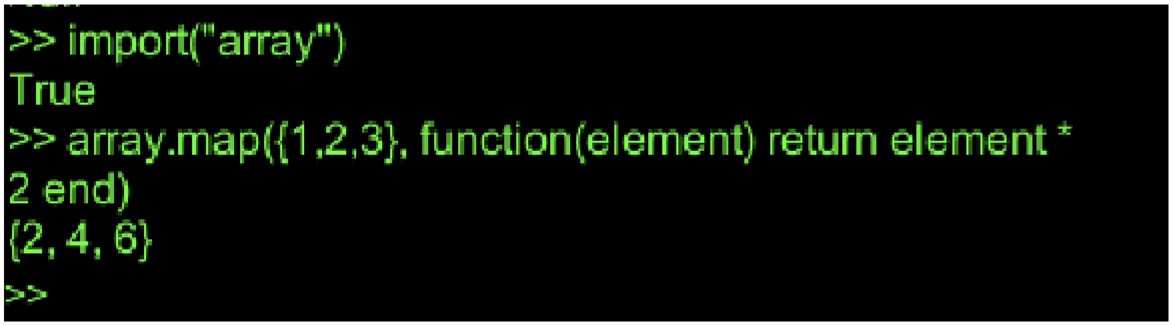
```
help() to show help prompt
>> library_help("array")
'Method Name: array.pop
Use:
array.pop(Any Array)
Method Description: Removes the last element of an
array and returns it, if it is empty, it returns null
Returns:
Name: Any, Description: Any type of object (depends on
the function), Type: Any
>>
```

Also, you can run any kind of code on the terminal

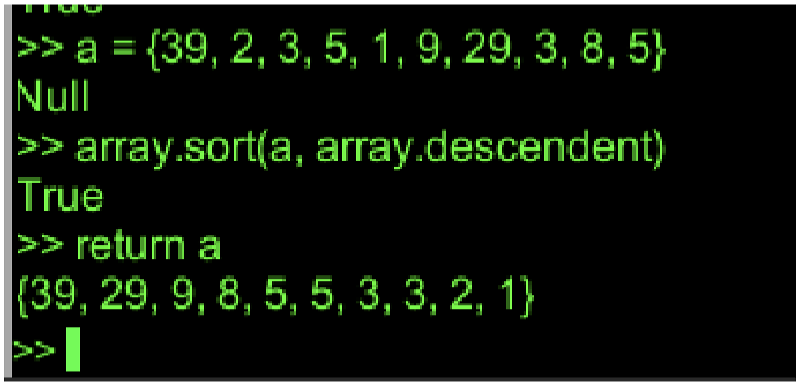
Some examples:



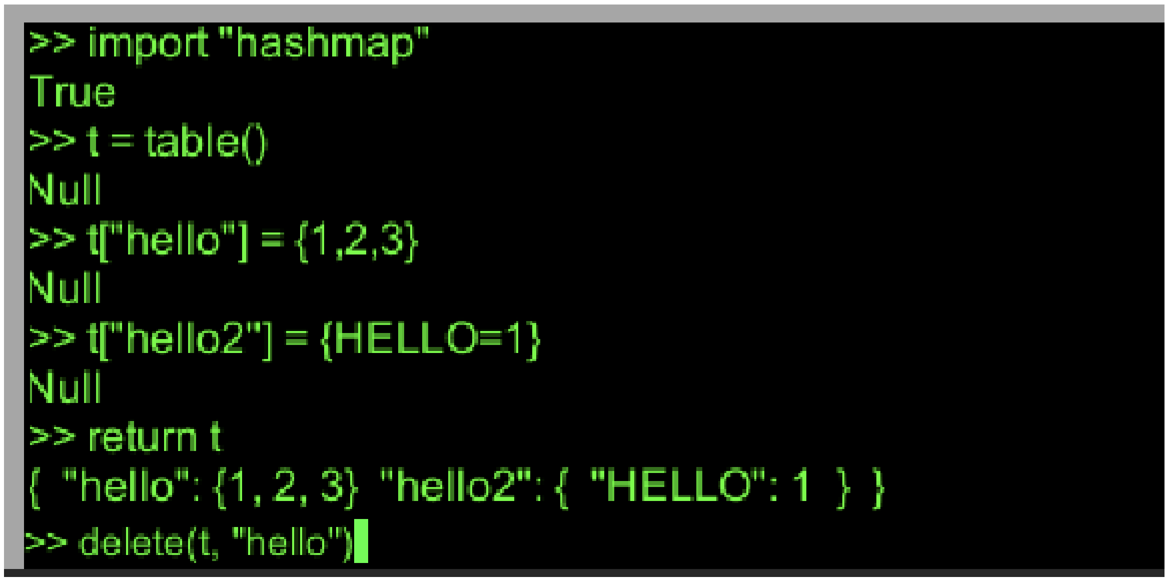
```
>> for i=0, 10 do print(i) end
0
1
2
3
4
5
6
7
8
>>
```



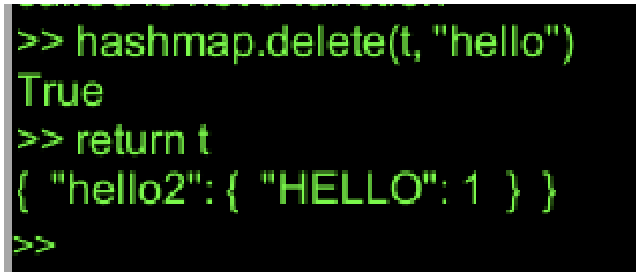
```
>> import("array")
True
>> array.map({1,2,3}, function(element) return element *
2 end)
{2, 4, 6}
>>
```



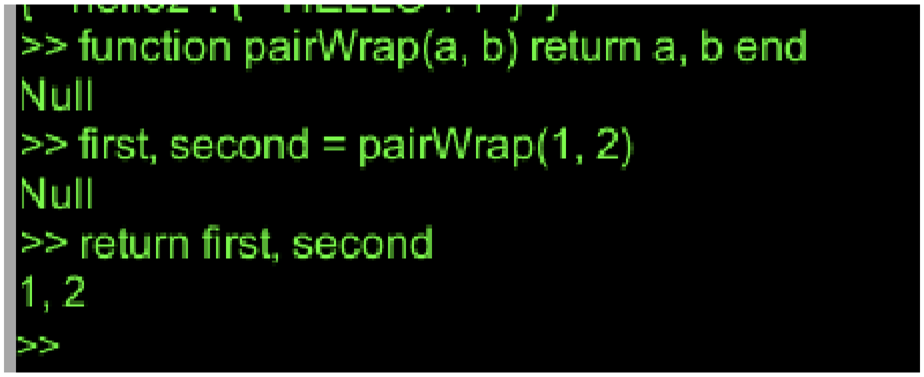
```
>> a = {39, 2, 3, 5, 1, 9, 29, 3, 8, 5}
Null
>> array.sort(a, array.descendent)
True
>> return a
{39, 29, 9, 8, 5, 5, 3, 3, 2, 1}
>>
```



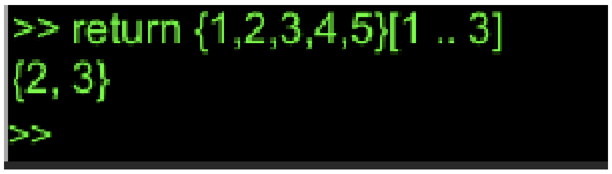
```
>> import "hashmap"
True
>> t = table()
Null
>> t["hello"] = {1,2,3}
Null
>> t["hello2"] = {HELLO=1}
Null
>> return t
{  "hello": {1, 2, 3}  "hello2": {  "HELLO": 1  }  }
>> delete(t, "hello")
```



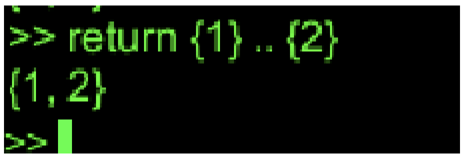
```
>> hashmap.delete(t, "hello")
True
>> return t
{  "hello2": {  "HELLO": 1  } }
>>
```



```
>> function pairWrap(a, b) return a, b end
Null
>> first, second = pairWrap(1, 2)
Null
>> return first, second
1, 2
>>
```



```
>> return {1,2,3,4,5}[1 .. 3]
{2, 3}
>>
```

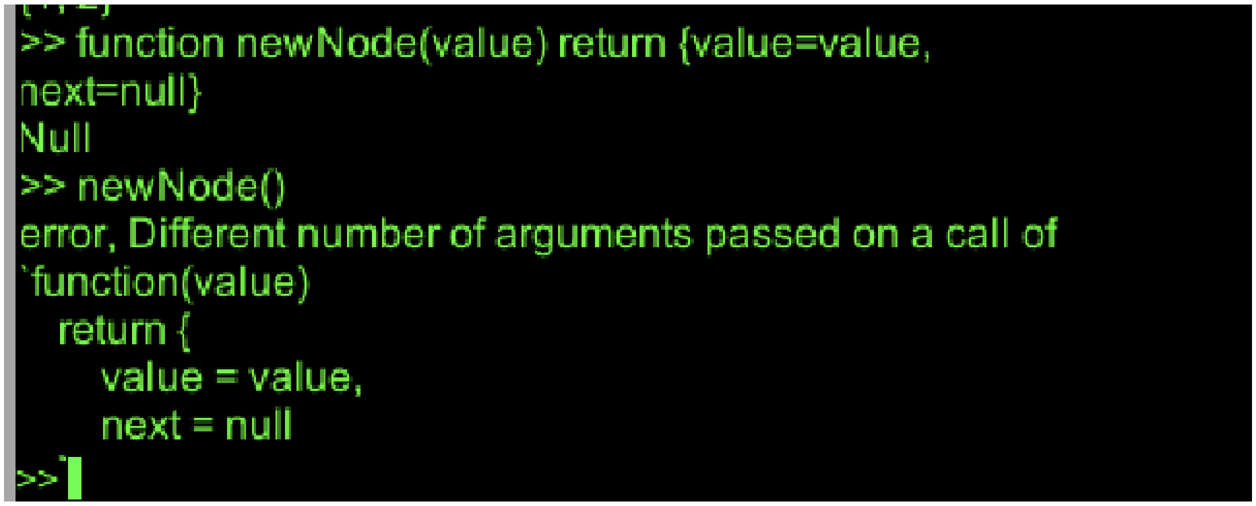


```
>> return {1} .. {2}
{1, 2}
>>
```

For examples on how to use Lua, please refer to the folder parser_testing, there is a whole file with Lua code.

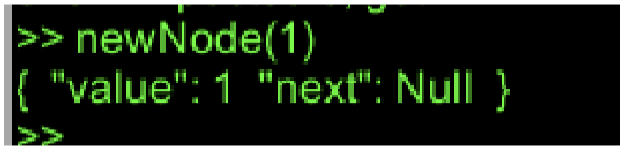
We will be developing more throughly documentation and more library features, at the moment, to see all the available things in the language you must go to the folder called `libs` and skim through the Csharp classes.

At the moment we got really good errors, for example consider that we create a constructor for a linked list node, and we call with too few values!



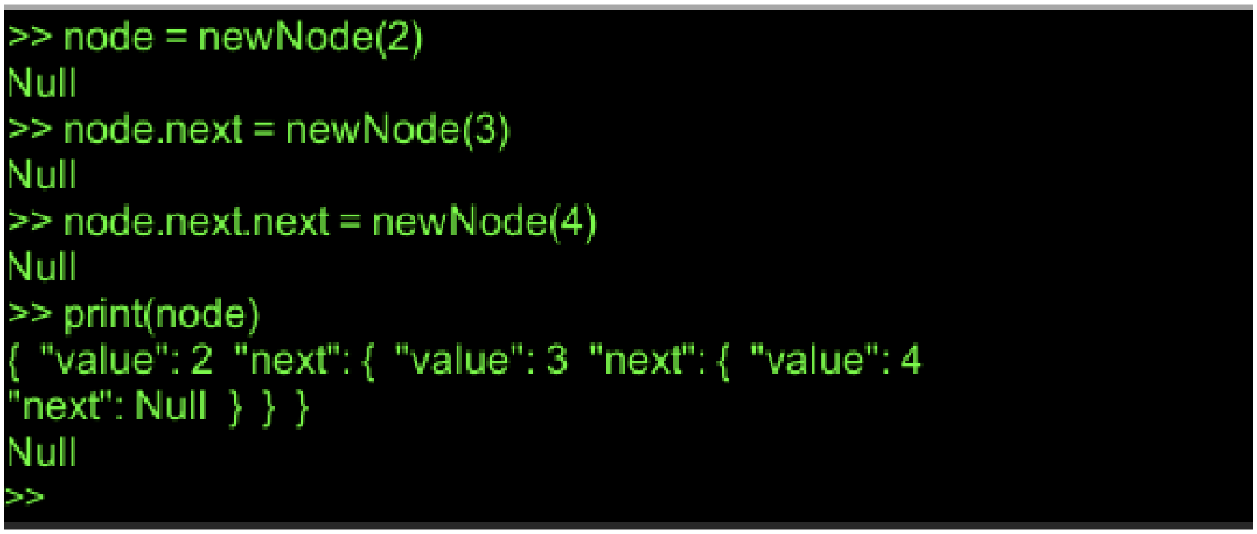
```
>> function newNode(value) return {value=value,
next=null}
Null
>> newNode()
error, Different number of arguments passed on a call of
`function(value)
   return {
      value = value,
      next = null
>>`
```

Now with calling fine:



```
>> newNode(1)
{ "value": 1  "next": Null  }
>>
```

We also got recursive printing:



```
>> node = newNode(2)
Null
>> node.next = newNode(3)
Null
>> node.next.next = newNode(4)
Null
>> print(node)
{  "value": 2  "next": {  "value": 3  "next": {  "value": 4
"next": Null  }  }  }
Null
>>
```

We got lots of things, and we can't wait to start creating problems so we can see cool examples on how to use the language :).