

```
myTable={}
print("-----table数组方式-----")
myTable[1]="baidu"
myTable[2]="guge"
myTable[3]="taobao"

print(myTable[1])
print("-----table键值对方式-----")
myTable2={}

myTable2[' baidu' ]="www.baidu.com"
myTable2[' 360' ]="www.360.com"
print(myTable2[' baidu' ])
```

迭代器方式遍历table

```
for key,value in ipairs(表名) do
print(key,value)
end
```

//数组方式

```
for key,value in ipairs(myTable) do
    print(key,value)
```

```
end
```

```
//键值对方式
```

```
for key,value in pairs(myTable2) do  
    print(key,value)
```

```
end
```

长度

```
table.getn
```

<3>table 长度

table.getn(表名)

返回 table 表的长度。

这个方式适合“数组模式”，不能用于“键值对模式”。

键值对就用：迭代器迭代，然后累加一个变量的方式获得长度。

```
index=0
```

```
for key,value in pairs(myTable2) do  
    print(key,value)  
    index=index+1
```

```
end
```

```
print(index)
```

增加元素

```
table.insert
```

```
table.insert(myTable, 1, "yahoo")
```

2.table 相关方法

<1>增加元素

`table.insert(表名, [位置], 值)`

往指定的位置增加元素，如果不写位置，默认往最后一个位置增加。

这个方式适合“数组模式”，不太适合“键值对模式”。

键值对就用：表名[‘键’] = 值 的方式添加即可。

移除元素

<2>移除元素

`table.remove(表名, [位置])`

白超神之神 如果不写位置，默认移除最后一个元素，如果位置值超出范围，不会报错，也不会有元素被移除。

这个方式适合“数组模式”，不能用于“键值对模式”。

键值对就用：表名[‘键’] = nil 的方式移除即可。

模块

2. 模块基本使用

1. 创建模块

- ① 创建一个新的 lua 脚本，并命名；
- ② 初始化模块；[模块其实也是 table 的代码格式，说白了就是初始化一个 table]
- ③ 在模块中定义变量和函数；
- ④ 模块的最后要写 `return 模块名`

备注：

在定义变量和函数的时候，名称前面必须加“模块名.”；

最终格式：模块名.变量名 模块名.函数名。

require “ ”

元表

1. 元表介绍

在本套课程的第 13 课讲解 table 的时候，我们都是对一个 table 进行操作，
用 table 这种结构模拟“数组”，“键值对”结构。

如果想同时操作两个 table，就需要让两个 table 之间发生“关联”，这里需要
用到一个新的知识点，这个知识点叫做：元表。

元表（metatable）就是让两个表之间产生“附属”关系，只需要操作主表，
就可以间接的操作元表。

在 Lua 语言中模拟“面向对象”，元表也是一个很关键且不可或缺的知识点。

小白超神之神

```
tableA = {}  
tableB = {}  
--tableB是tableA的元表  
setmetatable(tableA, tableB)  
--判断tableA是否有元素
```

```
print(getmetatable(tableA))
```

```
tableC={name="Jack", age=10}
```

```
tableD={gender="男", adress="USA"}
```

```
setmetatable(tableC, tableD)
```

```
tableD.__index=tableD--设置元表的index索引
```

```
print(tableC.name)
```

```
print(tableC.gender)
```