第一章 开始

1. Hello World

```
print(“Hello World”)
```

2.阶乘

```
--定义一个阶乘函数
function fact(n)
    if n==0 then
        return 1
    else
        return n*fact(n-1)
    end
end

print("enter a number")

a=io.read("*number")
print(fact(a))
```

3.分隔符的问题

语句之间不需要分隔符，但可以用分号来分割

如： **a=1;b=a*2** 和 **a=1 b=a*2** 是一样的。

4.运行程序块的方法

1.lua -i 程序块名

2.dofile(“xxxx.lua”) 对测试极有帮助

5.注释

行注释 **-**

块注释 **--[[ ]]** 重启代码 **---[[ ]]** 开头加 **-** 即可

6.特殊值 **nil**

访问一个未初始化的变量会返回 **nil**

删除一个全局变量就把它赋值为 **nil (**后面会看到 全局变量是存在 **table** 中**)**

第 2 章 类型与值

1.基本类型

nil	没有任何有效值
boolean	**false** 和 **nil** 为假 其他为真
number	表示实数 没得整数类型
string	一个字符序列 不可变 转义和 **C** 一样 字母字符串 **a=[==[<html>...]==]** **#** 长度操作符 **print(#”hell”)**
userdata	自定义类型，主要用来表示一种有应用程序或者 **C** 创建的新类型。
function	**lua** 中函数是作为第一类值来看待。这表示函数可以存储在变量中，可以参数传递，可以作为函数返回值。后面会具体介绍。

thread　　线程　以后具体介绍。
table　　实现了关联数组

表 **table**

table 类型实现了关联数组。

关联数组是具有特殊索引方式的数组。可以用整数，字符串或者其它类型的值(除了 **nil**)来索引。不固定大小。

table 是对象。它创建是通过 构造表达式完成的。如：

```
a={}    --创建了一个 table
k="x"
a[k]=10  --新条目 key="x" value=10
```

table 是匿名的。一个 **table** 变量和 **table** 没固定关联性。变量只是引用它。如果没有一个 **table** 的引用时，**lua** 就会删除 **table** 并复用内存

a.x 和 **a[x]**

a.x 表示 **a["x"] x** 作为索引

a[x]表示 **x** 是一个变量，把变量 **x** 的值作为索引

如：
```
a={}
x="y"  --变量 x
a[x]=10  --存入索引为"y"
printf(a[x]) --10
printf(a.x)  --没有定义"x"索引的字段
prinft(a.y)  --10
```

第 3 章 表达式

1.关系操作符

~= 用于不等性的测试 **nil** 只与其自身相等

对于 **table**，**userdata** 和函数，**lua** 做引用的比较。引用同一个对象才相等。

2.逻辑操作符

lua 中的 **and or** 也是短路求值。

3.优先级

表 3-1　Lua 操作符的优先级

^
not　#　　-（一元）
*　　/　　%
+　　-
..
<　　>　　<=　　>=　　~=　　==
and
or

4.table 构造式

初始化数组： days={“Sun”,”Mon”} days[1] Sun 记录风格

a={x=10,y=20} a[‘x’] 10 列表风格

两种风格可以混用。更通用的是：

ope={[‘+’]=”add,[‘-’]=’sub’} 用方括号初始化索引值

如果要以 0 开始索引，days={[0]=”Sun”,”Mon”} 不推荐，内建函数以 1 开始索引

a={x=10,y=45;”one”,”two”} 分号区分列表和记录

第 4 章 语句

1.多重赋值

a,b=10,2*x 和 python 一样

交换变量 x,y=y,x

如果个数不一致，那左边多了就赋值 nil 右边多了就被丢弃

a,b,c=0,1 c 为 nil a,b=a+1,b+1,b+2 b+2 被丢弃

在函数返回多个值的时候需要多个变量来接受 如 a,b=f() a,b 分别接受第一，第二返回值

2.局部变量和块

局部变量只作用于声明它们的那个块。

块是一个控制结构的执行体，或者是一个函数的执行体或者是一个程序块。

如：
```
if  i> 20 then
    local x      --尽可能地使用局部变量
    x=20
    print(x+2)
else
  print(x)
end
print(x)
```

```
local a, b=1,10
if a < b then
  print(a)   --1
  local a
  print(a)   --nil
end
print(a,b)    --1,10
```

3.控制结构

if for while 以 end 结尾

repeat 以 until 结尾

1. if then else --elseif 嵌套的时候用 替代 switch

```
if a<0 then a=0 end
```

2. while

```
 local i=1
 while a[i] do
   print(a[i])
   i = i+1
end
```

3.repeat --与 c 中 do-while 相同

```
repeat
  line=io.read()
until line ~=" "
print(line)
```

4.for -分为数字型和泛型

数字型：

```
for i=10, f(x) do print(i) end
--在列表中查找某个值
local found=nil
for i =1,#a do
 if a[i] < 0 then
      found=i
     break
 end
end
print(found)
```

泛型：--通过迭代器函数来遍历所有值

```
for i,v in ipairs(a) do print(v) end
```

5.break 和 return

break 跳出循环，并不跳出 if (C 中经常出错)

return 从一个函数中返回结果或者简单的结束一个函数的执行