Ruby学习日志

基础语法:

**注释**：

“#注释内容”或者

“=begin

这里是有注释很多很多

=end”

**输出**：

Print，puts，p都可以

变量：

本地变量： time 或者是 \_time

实例对象： @time

类： @@time

全局变量： $time

数据类型：

Numeric，String，Symbol，Boolean，Array，Hash。

变量技巧：

“hello #{name}” a,b = b,a; 3.times{puts “hello”} “hello” \* 3

这些分别是什么意思？

if：

这里标注一下特殊的 if a > 5 then puts a end

特殊的第二版（超简洁） puts a if a > 5

Unless

他不再是else，这个关键字的作用等同于if（！XXX） === unless（XXX）

If elsif else：

这里不啰嗦，注意一下关键字 elsif 即可。

Switch：

改变后的switch完全不一样。

Case xxxx

When “keyValue” then “result”

When ….

Else “result” #这里等同于defualt

End

注意：首字母不用大写，这里是word排版默认大写，我未作更改。

Loop：

**循环：**

3.times{ puts “hello world” }

搞定：等同于 ×3

**For：**

For x in 数组

Puts

End

**While：**

While i > 5 do

I -= 1

End

等同于

I -= 1 while i > 5

**Untile:**

语法通while。

使用loop do 代替 while ture

Loop do puts “OK” end

中断循环关键字

Break，next，redo，retry。

疑问：redo，retry么意思？

方法：

def关键字

def plus（x，y）

z = x + y

return z

end

调用

Plus（3,4）

**简洁版**

**Def plus x，y**

**X+y**

**End**

**调用**

**Plus 3,4**

只有overriding 没有loading，这个是什么意思？

带有block的方法：

Def hello

Yield

End

Hello {puts “hello,block”}

另外一种形式，但是不完全一致

Def hello &block

Block.call

End

Hello {puts “hello,block”}

Lambda 和 Proc

Lambda{}

Proc.new{}

区别：

Proc中的return会跳出当前作用域

Lambda中的return不会跳出当前作用域

Class：

正常定义，使用new 实例化调用

属性访问级别：attr\_accessor:name 读写

Attr\_reader:name

Attr\_writer:name

继承

关键字 “<”

Class A < B

Def xxx

End

End

A继承自B

Module 更强大的interface

Module Eat

Def eat

P “I can eat”

End

End

Class Pig

Include Eat

End

Pig.new.eat

常量访问：

Module Math

PI = 3.24

End

调用 Math::PI

Module命名空间：

在module外面再嵌套一层就认为是命名空间了

比如

Module Foo

Module Bar

Def self.say

P “sss0”

End

End

End

调用：Foo::Bar.say

备注：当然module中也可以嵌套class

module的特殊用法singleton:

module Item extend self

def name

p “I’m item”

end

end

调用： Item.name

扩展方法

class Fixnum

def plus n

self + n

end

end

1.plus 3

更多:

[Fiber](https://gist.github.com/1050566)

[Class](https://gist.github.com/872171)

[System](https://gist.github.com/822091)

[Proc](<https://gist.github.com/874438>)

ruby中，一切都是对象，比如 “Lampard”.reverse。

特点：在ruby中可以省略（），{}等。

irb：irb命令是启动ruby

p puts print 的共性与区别

共性：都是向屏幕打印东西

区别：

print:打印一行内容，不追加换行符，其他两个命令会追加换行符

p：不解释“”中间的字符，原封不动的输出

puts：对字符串中的\n等转义字符做了解释

网上有资料说p无法打印中文，在2.0.0版本无此问题

格式控制符：

％　　格式说明引导符。

－　　指定左对齐输出。

０　　指定空位填零。 eg. %0md，m目标是位数

m.n 指定输出域宽度及精度。 eg. %m.nd，m目标是位数，n表示小数位数。

格式字符：指定输出项的数据类型和输出格式。

d　　有符号十进制整数。

o　　无符号八进制数。

x　　无符号十六进制数。(小写的x格式中用小写字母a,b,c,d,e,f来表示10到15之间的数，大写的X则用大写的ABCDEF来表示10到15之间的数)

u　　不带符号的十进制整数。

c 　 输出一个字符。

s　　输出一个字符串。

e　　以指数形式输出实型数。

f 以小数形式输出实型数。

g　　 自动决定输出格式为e和f中较短的一种，不打印无效的零。

%　　输出%。

注：“+” 拼接时要注意两边的变量时同类型或者是可以隐式转换的。

输出段落可以使用 <<PARAGRAPH 段落文字 PARAGRAPH

在ruby中 get与get.chomp() 的区别

gets和gets.chomp()都表示读入用户的输入并用于输出，但两者还是有所不同，其中gets是得到的内容后，在输出时后面接着换行；而gets.chmop()得到的内容输出时后面不带空格和换行。

在ruby中 include，require，expend，load四个好兄弟。

*require，load用于包含文件，include则用于包含的模块。*

*require加载一次，load可加载多次。*

*require加载Ruby代码文件时可以不加后缀名，load加载代码文件时必须加后缀名。*

*require一般情况下用于加载库文件，而load用于加载配置文件。*

*————引子http://bbs.chinaunix.net/thread-3625799-1-1.html*

没有搞明白他们的用途和用法，待练习

文件操作

Close – 保存并且关闭文件

read – 读取文档全部内容

readline – 读取一行

truncate – 清空

write(stuff) – 写入内容

在ruby中存在这些关键字，比较有趣

and ，那么请问and和&&有什么区别？

通过字面理解一下（肯定不一样，要不然我也不会问了），and是“和”的意思，&&是“并且”，好像也没差啥啊。那么请回答我 puts true and false 的输出结果（true），接着告诉我 puts false and true 的输出结果（false）。××××，我要讲脏话了，这是什么情况。

想知道答案请猛击下面连接

<http://zhaoningbo.iteye.com/blog/1982761>

看了这个地址虽然有些懵懵懂懂，但是还是不能完全明白，

后来通过http://stackoverflow.com/questions/2802494/i-true-and-false-in-ruby-is-true

完全明白了。感谢教练！

Array

::[]

::new

::try\_convert

#&

#\*

#+

#-

#<<

#<=>

#==

#[]

#[]=

#assoc

#at

#bsearch

#clear

#collect

#collect!

#combination

#compact

#compact!

#concat

#count

#cycle

#delete

#delete\_at

#delete\_if

#drop

#drop\_while

#each

#each\_index

#empty?

#eql?

#fetch

#fill

#find\_index

#first

#flatten

#flatten!

#frozen?

#hash

#include?

#index

#initialize\_copy

#insert

#inspect

#join

#keep\_if

#last

#length

#map

#map!

#pack

#permutation

#pop

#product

#push

#rassoc

#reject

#reject!

#repeated\_combination

#repeated\_permutation

#replace

#reverse

#reverse!

#reverse\_each

#rindex

#rotate

#rotate!

#sample

#select

#select!

#shift

#shuffle

#shuffle!

#size

#slice

#slice!

#sort

#sort!

#sort\_by!

#take

#take\_while

#to\_a

#to\_ary

#to\_h

#to\_s

#transpose

#uniq

#uniq!

#unshift

#values\_at

#zip

#

if使用规则

1、每一个if都必须有一个else，

2、哪怕这个else永远都执行不到，那么就在else里面加上一个die方法，打印出错误信息

3、千万不要使用超过两个if，如果有，请移驾到另外一个函数里面

4、把一个if段当做一个段落来处理，上下空一行

5、if里面的bool应该尽量保持简单，如果很复杂，请使用一个变量

while使用规则

1、只有在永不停止的循环中才使用while

2、其他类型的循环都是用for。

Keywords

alias

and

BEGIN

begin

break

case

class

def

defined?

do

else

elsif

END

end

ensure

false

for

if

in

module

next

nil

not

or

redo

rescue

retry

return

self

super

then

true

undef

unless

until

when

while

yield

true

false

nil

constants

strings

numbers

ranges

arrays

hashes

\\

\'

\"

\a

\b

\f

\n

\r

\t

\v

ruby符号

::

[]

\*\*

-(unary)

+(unary)

!

~

\*

/

%

+

-

<<

>>

&

|

>

>=

<

<=

<=>

==

===

!=

=~

!~

&&

||

..

…

Hash

ruby中声明Hash的格式

stuff = {:name => "Rob", :age => 30, :height => 5\*12+10}

基本格式： key=>value

ruby:dict

ruby:initialize

gem 工具

ruby中的try。。catch

begin

//TODO

rescue XXXXError

//TODO

疑问列表：

1、ruby如何实现 多态 封装 继承<

2、ruby 中 :: 是什么意思？

答：namespace的引用，常量的调用都可以使用::，不过一般在使用namespace的时候使用::的。

3、ruby 测试用例

4、ruby中没有main函数，那么怎么确定那里是入口呢？

因为是解释型语言，所以不需要入口，当请求某个方法时，系统就加载对应的模块进行解释。

5、×XXX想这种类似C语言中指针的用法是怎个意思？

答：如果你想要一个函数的参数可以接受任意多个参数，那么只需要在普通的参数前面加上×即可

6、区间作为条件

ruby 小技巧：

<https://ruby-china.org/topics/18512>

=～ 匹配正则表达是，如果串中存在，则返回起始位置，否则返回nil （在bool表达式中 ，nil等同false，）

“”.sub(/正则/，‘替换值’） 把“”中的第一个匹配正则的内容替换成 替换值 （gsub是替换所有）

Block 与 yield

block：

{}一行处理使用{}

do end 多行处理使用do end

block\_given?

可以使用在yield之前，如果有block被传入则执行，否则不执行

eg。 if block\_given？

yield

总结：yield相当于event

block相当于 delegate

public protected privite：三种访问级别

“”.dup 拷贝字符串一份。

Ruby attr\_accesstor 与 cattr\_accesstor

<http://lee2013.iteye.com/blog/1098914>

ruby collect 中的inject方法

收集第一个作为基数，第二个开始迭代

“string”.squeeze!(“ ”)清除空格

“string”.scan(/正则/) 同样用法的split（/正则/）

ruby正则表达：

变量 $&：保存匹配字符

变量 $`:保存满足匹配之前的字符

变量$':保存满足匹配之后的字符