# 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” }

疑问：times是关键字？

**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)