输出print(不换行),puts,会换行)，p(也是输出)，pp(格式化输出)：print(‘value’)

Print v1,v2,v3 : 会分别输出

Print “#{variable}” 输出变量

注释 #

符号：对象，作为名称标签使用,特征是前面有:

Sym = :foo 用于散列的key

可以和字符串替换 sym.to\_s “foo”.to\_sym

数组[] ,

循环：names.each do |n|

Puts n

end

散列{}

{:key1 => “v1”,:key2 => “v2”} or {:key1 : “v1”,:key2 : “v2”}

Address.each do |key,value|

Puts “#{key},#{value}”

End

方法的定义

def 方法名

逻辑

end

库的引用

Require 库名: require hello(文件名)

类：

Numeric(数值)，String(字符串)，Array(数组)，Hash(散列)，Regexp(正则表达式)，File(文件)，Symbol(符号)

变量

局部变量：英文小写字母开头

全局变量：以$开头, 以文件为界限，java是以类为界限，尽量不用。

实例变量: 以@开头

类变量： 以@@开头

伪变量：nil,true,false,self

常量：

以大写字母开头

多个赋值操作

a,b, \*c = 1,2,3,4,5

P [a,b,c] #=> [1,2,[3,4,5]]

Or

Ary = [1,2]

A,b = ary 数组赋值

变量命名

变量名和方法名，使用\_隔开各个单词

类名和模块名使用 sortList, 驼峰名

条件判断

真：false,nil以外的所有对象

假：false,nil

约定方法以？结尾对于返回真假值的情况。

逻辑运算符号：&&，||，！

If语句

If 条件 then

逻辑

End

If 条件 then

逻辑

Elsif 条件 then

else

End

unless语句 : 条件为假才执行

Unless 条件 then

处理

End

Case 语句，case使用===来判断的

Case 比较对象

When 值1 then

Logic

When 值2 then

Logic 2

Else

处理

End

循环

Times,while,each,for,until,loop 六种方式

Times语句

循环次数.times do [|i|] []代表是否可选

end

For 语句

For i in from..to

End

Or

For 变量 in 对象（一般是数组） do

End

while语句

While 条件 do (do可以省略)

end

Until 语句，为假的才执行

Until 条件 do (do可以省略)

End

Each方法将对象集合中的对象逐个取出

对象.each do |变量|

End

Loop 语句

没有终止的条件

Loop do

~

End

Break:结束循环

Next:下面不执行，开始下一次循环

**方法 参数没有的话，可以不用写括号**

Def 方法名

End

实例方法：对象.方法名

类方法：类本身方法

函数式方法：比如print,sleep,没有对象

参数：

1 默认参数 参数名=值

1. 剩余参数 \*arge: 可以传递多个参数

Def fun(a,\*b,c)

1. 关键参数

Def func(key1:value1,key2:value2,\*\*args) \*\*args传递多个散列参数

数组和散列的传递

数组 \*list : 各个元素被依次传递个方法

args = [2,3]

foo(1,\*args)

散列

散列传入，可以省略{}

返回值

Return

Return 可以省略，返回最后一个表达式的结果

类

判断是那个类 ary.class

判断对象是否属于某个类：str.instance\_of?

定义：

Class 类名

定义

End

构造方法： initialize方法，实例变量用@表示，可以在实例内部使用

存取器：

全局变量 name

Def name

Name

End

Def name=(value)

@name = value

End

简单方式 attr\_accesspr :name在类里定义，意思为 读写（定义）

特殊变量self：作为实例内部的接收者

类方法定义

class << 类名 ~end的方式

例子

Class << HelloWorld

Def hello(name)

end

End

Or

Def HelloWorld.hello(name)

or

def self.hello(hello)

end

使用HelloWorld.hello

常量

Class HelloWorld

Version = 1

End

使用

HelloWorld::Version

类变量

Class HelloCount

@@count = 0 类变量，可以修改

Def HelloCount.count

@@count

End

End

方法修饰符public,private,protected

Class test

Public

以下方法为public方法

Def pub

End

Private

以下为private 方法

Def priv

End

end

继承

Class 类名< 父类名

类定义

End

Instance\_methods : 看类的实例方法

模块 ruby

模块表现的是事物的行为部分

模块不能拥有实例

模块不能被继承

Module MyModule

常量，方法

Version = 1.0

Def hello(name)

End

module\_function :hello 指定hello为模块函数

End

在class中使用

HelloModule:: Version

Class MyClass

Include MyModule

MyModule.hello

end

Myclass.hello() 外部使用

extend方法

class MyClass

定义类方法

extend ClassMethods

定义实例方法

include InstanceMethods 都是Module

end

逻辑运算符的应用

1. 表达式的执行顺序是从左到右
2. 如果逻辑表达式的真假已经可以确定，则不会再判断剩余的表达式
3. 最后一个表达式的值为整体逻辑表达式的值

条件运算符

条件？表达式1：表达式2

范围运算符

.. 或者... 范围

1. .10).to\_a

异常处理

begin

逻辑处理

rescue

发生异常后的处理

ensure

都会处理的结果

end

块

对象.方法名(参数列表) do |块变量|

处理

end

Or

对象.方法名(参数列表) { |块变量|

希望循环的处理

}

yield 关键字的作用就是执行方法的块

数组

1. 创建

nums = []

Array.new

Array.new(5) or new(5,1) 默认值为1

获取元素

a[n] n可以为负数，从最后开始取

a[n..m] or a[n...m]

a[a,len]

操作

push : 追加元素

pop： 删除元素

last： 引用元素

替换数组元素

a.collect!{|item|} !表示改变原来的数组，没有！表示返回新数组的方法

b.map{|item| }

迭代

each

each\_with\_index do |elem,i|

散列类

创建{}

使用Hash.new

设定 store(“key”,”value”)

获取 fetch(“key”)

IO类

$stdin 标准输入

$stdout 标准输出

$stderr 错误输出

Time类和Date类