**Ruby 学习笔记**

一 .ruby 简介

* 1. ruby 是一种跨平台，面向对象的解释性语言。
  2. Ruby特性

1. 格式灵活。
2. 大小写敏感。
3. 单行注释以 “#”开头，多行注释包括在 “=begin” 和“=end”中。
4. 分行符：多行语句写在同一行时，必须用分号隔开，行尾可以不写，换 行符相当于分号。单行语句书写到多行时，要使用“\”作为结尾。
5. 关键字. ruby中有38个保留关键字，和其他语言一样，不能把他们作为变量，类或者方法名。
6. null => true

false,nul => false

* 1. 安装

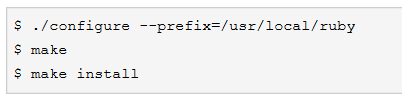
①下载

wget <https://cache.ruby-lang.org/pub/ruby/2.5/ruby-2.5.1.tar.gz>



②解压 tar -zxvf

③安装 这里路径指定为 /usr/local/bin. 以免还需要设置路径。



1. 基本语法

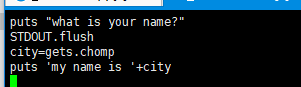
2.0变量

变量的命令有一定规则：以小写字母或下划线开头，变量中只能包含字母，数字和下划线。关键字不能做变量名使用。

2.1 输入与输出

puts:输出。 gets和chomp输出。

Gets方法得到输入字符串，包括\n回车符。Chomp方法去掉回车。



[root@localhost proj]# ruby test

what is your name?

zs

my name is zs

2.2 定义函数

|  |  |
| --- | --- |
| def smud(a=99,b=a+1)  [a,b]  end  puts smud  99  100 | def smud(\*a)  a.each do | words|  puts words  end  end  smud("one","two")  one  two |

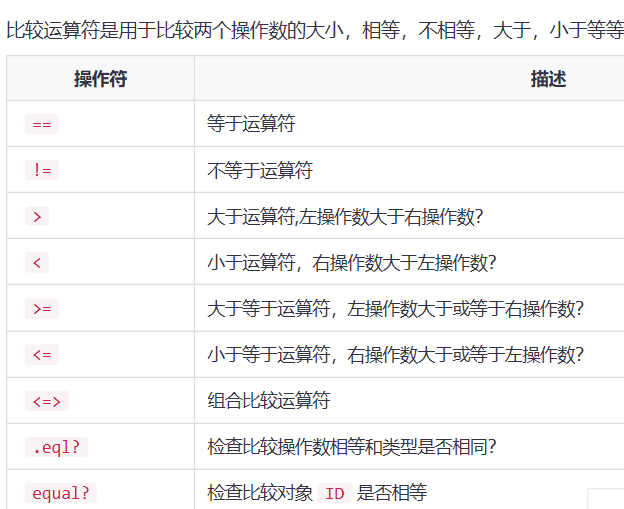
1. 方法体需要在def 和end 之间。参数列表需要括在括号里。
2. 方法的返回值就是该方法的最后一行。
3. 参数可以带默认形参值。调用时实参也可不用括号。
4. Def method(\*my\_string) 表示可选参数，可以传多个，也可不传。通过‘\*’，会把传过来的参数放到数组my\_string中。
5. 重磅方法。带“！”的方法叫重磅方法。，意为比较危险，执行不带“！”方法后，会返回一个全新的对象，如果时带“！”的方法，返回的时原来的对象。

2.3 控制语句

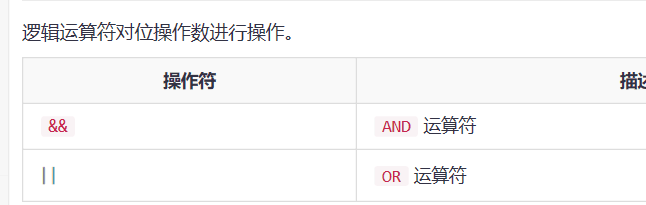
If-elsif-else-end if-else-end (除了nil和false是假外，其他所有的都是真的)

|  |  |
| --- | --- |
| def med( a)  if a.nil?  puts "a 不存在"  elsif a.empty?  puts "a 未空"  else  puts "a 存在且不为空 #{a.length}"  end  end  a 存在且不为空 11 | a =6  if a>4  puts ">4"  else  puts "<=4"  end  >4 |

1. 常用运算操作



1. 常用逻辑操作



2.4 循环语句

|  |  |
| --- | --- |
| var=0  while var<2  puts var.to\_s  var+=1  End  0  1 | for i in 0..5  if i < 3 then  next # break,redo  end  puts "局部变量的值为 #{i}"  end  局部变量的值为 3  局部变量的值为 4  局部变量的值为 5 |

1. next :跳到循环的下一个迭代,不继续往下.。快内则终止程序。
2. break: 直接跳出循环。如果是函数内部break,则返回nil。
3. Redo:重新开始循环。
4. 要点

3.1 字符串

1)字符串拼接

|  |  |
| --- | --- |
| str1="zs"  str2="is a real man"  a=8  str3=" #{str1} not #{str2} #{8}"  puts str3  zs not is a real man 8  注：#{}拼接 | str1="zs"  str2="is a real man"  str3= str1+str2  puts str1  str4= str1.concat(str2)  puts str1  puts str3  puts str4  zs  zsis a real man  zsis a real man  zsis a real man  注： .concat为在末尾追加。 |
|  |  |

1. 字符串分割

str.split(pattern=$;, [limit])

基于分隔符，把str分成子字符串，并返回这些子字符串的数组。

①如果pattern 是字符串，如例2，将他作为分隔符使用。

②如果是正则表达式，置于（/..../）两条斜线之中。如例一。

③限制如果省略，一直分隔到结尾。如果是正数，分割n-1次。如果是负数，数量不限制，可能会超出内存。

|  |  |  |
| --- | --- | --- |
| **str2="is a233 re44al?tman"**  **a = str2.split(/[\s?]/)**  **puts a**  **is**  **a233**  **re44al**  **tman** | **str2="isa233re44al?tman"**  **a = str2.split("23")**  **puts a**  **isa**  **3re44al?tman** | **str2="isa233re44al?tman"**  **a = str2.split("a",2)**  **puts a**  **is**  **233re44al?tman** |

1. 字符串替换

**str.gsub(pattern, replacement)**

**str2="2018\_10\_19"**

**str3=str2.gsub("\_","-")**

**puts str3**

**2018-10-19**

1. 长度和子串

|  |  |
| --- | --- |
| str2="2018\_10\_19"  puts str2.size  puts str2.length  10  10 |  |

5)常用方法

3.2 哈希（hash）

1） 哈希是类似”key” => “value” 这样的键值对集合。哈希类似于一个数组，只不过它的索引不局限于使用数字。哈希的索引可以是任何类型。Hash没有顺序。

2）创建hash，在使用不存在的索引时都会返回创建时的默认值。

H1=Hash.new # 默认值是 nil

H2=Hash.new(“mo”) # 默认值是 “mo”

H3={“one”=>”zs”,”two”=>”ls”}

3）添加键值对（修改）

H2[“three”]=”ww”

1. 删除键值对

H2.delete(“three”)

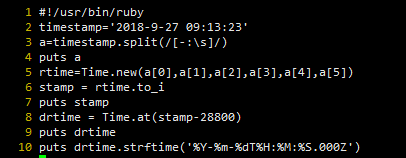
1. 遍历
2. 常用方法(首字母均小写)

|  |  |
| --- | --- |
| size() | 返回Hash 长度。 |
| Length() | 返回Hash对象的长度。 |
| Include?(key) | 判断指定的Hash对象是否包含指定的key。包含返回true |
| Has\_key?(key) | 判断指定的Hash对象是否包含指定的key。 |
| Delete(key) | 删除Hash对象中指定的对应元素。 |
| Keys() | 返回由Hash对象中全部key组成的数组。 |
| Values() | 返回由Hash对象中全部value组成的数组。 |

3.3 数组

1. 操作实例

4.1时间转换



4.2 字符串数字转换

Ruby的整数，浮点数，字符串的类均提供了 to\_i,to\_f,to\_s 三个函数。

to\_i: 转换成整数

to\_f: 转换成小数

to\_s： 转换成字符串