**Array**

Create:

array = [ ] arr = Array.new arr = Array.new(size[,size\_num\_of\_same\_value]) arr = Array.new(size){ size\_num\_of\_same\_value } %w()会以空格为分界将内面分解成串数组

arr = Array.new(aotherArr) arr.new(size){ |index| block} 会做遍历 arr1 & arr2交集(去重)

Get:

Arr[sizeAsIndex] Arr[index](index可取负) arr[out\_of\_index] = nil arr[index, num] arr[index…index] arr.at(index)

Arr.fetch(index, ‘value\_when\_out’)取index下的值，超过取越界取后 arr.first/last arr.take/drop(size\_of\_first/last)

Information:

Length/size/count empty? Include?(value) arr.count(value/&block)可以直接用这个计算出它里面的要求数的数量如奇数 根据block的t/f

Add:

Arr.flattern(级数)拆分数组中的数组元素

Arr.push(obj) == arr << obj (becareful!!以元素形式放入) Arr.contact(arr2)(真正的数组连接) arr1 + arr2 arr.unshift(value)/shift() arr.insert(index, \*value)

Remove:

Arr.(reject=drop\_while)/(delete\_if) &block (delete\_if = reject! &block=>t/f)

Arr.pop(后)/shift(前) arr.delte\_at(index) arr1 – arr2 arr.delete(value) arr.compact 去空 Arr.uniq arr.clear

Value:

Arr = Array.new{2,Hash.new} => arr[0][‘catName’] = ‘Tom’ arr[1][‘ratName’] = ‘Jerry’

**String**

Operation

<=> str.casecmp(str2)(忽略大小写的比较) == str[] str[‘str\_part’]=nil/str\_part可判断是否包含str\_part str[index, size]

str[/rex\_with\_group/, group\_name/num]可以做提取 str.capitalize对整个串的首字母单词变大写， str.center(total\_size, [‘fill\_str’])将原字符串在新的total\_size中居中以fill\_str或空格填充

str.chop削去最后一个字符 str.downcase str.each\_char/byte/

Create

Str = ‘value’/”value”/ %q/Q()/<>/||/!!/[]/{} !!! << EOF

im string too

EOF

!!!EFO要这么写也会生成一个字符串目前碰到的

**Hash**

Hash中:key和’key’两个不是一种类型

Hash.invert

Get:

Ha[key] ha.each {|key,value| } /ha.each\_value/key fetch同array hash.flattern(num)将hash转成数组也能继续拆其中的数组

Create:

Hash[arr](加偶数数量的数组,其可变成键值形式的hash) Hash[[key, value], …] Hash[key => value, …]

Hash.try\_convert(obj)尝试将对象转化成数组

Hash.new(default\_value)

Message:

Hash.default=value后取hash[‘haven’t\_defined’] = value empty? Ha.has\_key/value?()=ha.member?=ha.include?

Ha.length

Option:

同样有select,reject,shift

== 同时键值对相同对返回true ha.merge!(h2)合并两个hash相同的键默认以h2替换h1可以加块去规则它{ |key, odlvalue, newvalue| block }

Key:

Ha.invert键值对反转

Key本身不应该是一个值，但可以把它变成一个值freeze (但实际好像并没什么用)

Ha.key(100)可以获取值所对应的键，如果有相同的值则取前面的那个键

Ha.assoc(‘key’)会返回键值数组

Delete

Ha.delete(‘key’)返回被删除的值(value)否则返回default或nil 有block时可自行设置

Ha.delete(‘Akey’){|key| “#{key} not found”}这里的参数应该加括号跟在delete后面

Ha.delete\_if {|key, value| }

**Dir**

Dir.foreach Dir.mkdir

**File**

File.open(能加代码块)/new(filename, mode) fileName.puts File.ftype(fileName) File.exist?(fileName)

File.atme/ctime(目录)/mtime(文件本身)

**Time**

Time.mktime/gm(yyyy, MM, dd, hh, mm, ss) Time.at(ss)

Time.new/now.wday~~~Time.new.strftime(“%a/%A”) Time.new.strftime(“%W/%U”)

Time.new.strftime(“%Y-%-%d %H:%m:%S”)

**fileutil**

mkdir(dir, options) mkdir\_p(dir, options) rmdir(dir, options) FileUtils.mv/cp不能跨磁盘分区移动

chmod(mode, list, options) mode->0(管理员 (可读1可写1可执行1)=7) (创建者 (可读1可写1可执行1)=7) (非创建者 (可读1可写1可执行1)=7) =>0755