1. GIAO DIỆN DÒNG LỆNH IRB

Chạy trên cmd   
và thoát bằng lệnh exit

1. BIẾN TRONG RUBY

Biến trong ruby sẽ tự nhận kiểu dữ liệu bằng đối số truyền vào.

Để nhận đối số từ bàn phím sữ dụng : gets.chomp.to\_i

Gồm 4 loại biến

* Biến local (chỉ dung trong phương thức) : biến được viết thường và có giá trị trong phương thức “tuoi = 19”
* Biến global(sử dụng trong và ngoài class): Được khai báo ngoài class và bắt đầu bằng dấu đô la “ $pi = 3.14”
* Biến hằng (chỉ được khai báo giá trị 1 lần và không đổi): được khai báo trong class và viết In “ PI = 3,14 “
* Biến instance(sử dụng trong class): được sử dụng trong class chứa nó và bắt đầu bằng dấu @ “@gioitinh = nam”

1. MỘT SỐ CÂU LỆNH

a>=18 && puts”Du 18 tuoi”

* Câu điều kiện

If điềukiện

end

hoặc

If điềukiện

elseif

end

else

end

1. SỐ VÀ PHÉP TOÁN

Kiểu dữ liệu

Để xem kiểu dữ liệu của biến : tênbiến.class

Để chuyển kiểu dữ liệu :

* tênbiến.to\_i (thực sang nguyên)
* tênbiến.to\_f (nguyên sangthực)

PHÉP TOÁN

Các phép toán tương tự các ngôn ngữ khác

Toán mủ : 2 mủ 4 : 2e4

Các phép so sánh tương tự ngôn ngữ khác

Đặc biệt : a <=> b (trả về 1 a>b; -1 b>a; 0 a=b)

So sanh kiểu dữ liệu: tênbien.equal?(10.0)

Làm tròn tenbien.round(số dữ số sau dấu phẩy)

Hoặc thư viện require “bigdecimal”

A = BigDecimal.new(‘0.9’)

B = BigDecimal.new(‘0.8’)

1. CHUỖI

Để tạo ra chuỗi

s = String.new(“abc”) hoặc s = “abc” hoặc s = <<ST …..ST

In giá trị chuỗi hay biến ra màn hình

Với

* puts “abc #{s}” : khi in ra sẽ đọc biến s để in ra
* puts ‘abc #{s}’ : khi in ra sẽ KHÔNG đọc biến s để in ra
* Để dánh dấu ký tự đạc biệt ta sử dụng như sau

puts “ Hi I\”m Duy”

* Để viết nhiều chuỗi trên nhiều hàng ta thưc hiện

puts %\* abvc cdsfsfd fdsfsddvv

Fsddsf ffdsfdf dsfdsfds fdsfdfd

Fdsfds fdsfs fdsfds

Fdsfs fdsff

\*

So sánh chuỗi

A = B

A != B

A > B

….

Để xem có rỗng không : A.empty?  true or false

Nối chuỗi A+B

Phép gán chuỗi A << B  tương đương nối chuỗi và gán cho A

Các hàm theo tác với chuỗi

c.upcase

c.downcase

c.capitalize : “Hello duy”

c.swapcase : ký tự up <=> down

c.reverse : đảo ngược chuỗi

c.length hoặc c.size : ký tự trong chuỗi

c.include?(“Kítucantim”) : tìm kiếm ký tự hay gì đó trong chuỗi , trả về true/false

c.sub(“Hello”, “Hi”)  Thay thế hello = hi

c.split(“kýtutach”)  tách các ký tự của chuỗi theo điều kiện truyền vào, trả về mảng

ĐỂ gán cho biến thao tác thì them dầu “!” vào cuối

1. MẢNG (ARRAY)

Các cách khai báo mảng

* Arr = Array.new  mảng rỗng
* Arr = Array.new(7)  7 phần tử
* Arr = [1,2,3,4,5,6,7,8]

Các thao tác với mảng

* arr.push(5) hoặc arr << 5 hoặc arr += 5 thêm vào cuối số 5
* arr.pop  lấy ra 1 phần tử cuối ; arr.pop(2)  lấy ra 2 phần tử cuối
* arr.unshift(5,6)  thêm vào đầu 5,6
* arr.shift  lấy ra phần tử đầu
* arr.size hoặc arr.length  số lượng phần tử trong arr
* arr.index(5)  tìm vị trí index của 5
* arr.flatten  chuyển phần tử là mảng thảnh các phần tử con của mảng chính
* arr.concat(array)  nối 2 chuỗi
* arr.slice(2,1)  lấy ở vị trí index 2, 1 phần tử
* arr.last or arr.first  trả về phần tử cuối , đầu của mảng
* arr.uniq  xóa các phần tử lập lại
* arr & array  trả về mảng phẩn tử giao nhau của 2 mảng

SẮP XẾP

* arr.sort  xắp sếp từ nhỏ đến lớn
* arr.sort.reverse  trả về sắp xếp từ lớn tới nhỏ (đảo ngược của sắp xếp)
* Hoặc sắp xếp theo kiểu function :

Arr.sort{|a,b| a<=>b}

EACH

arrs.each { |arr| puts arr}  chãy qua từng phần tử của mảng và thực hiện lệnh puts

ALL

arrs.all? {|arr| arr >0}  coi toàn bộ mảng có lớn hơn không không? Trả về true/flalse

arrs.any? {|arr| arr >0}  coi có phần tử nào lớn hơn 0 hay không, có 1 phần tử lón hơn không sẽ trả về true , ko thì false

SELECT

arrs.select {|arr| arr>15}  trả về mảng các phần tử thỏa điều kiện arr>15 trong mảng arrs

Ngược với select ta có reject

arrs.reject {|arr| arr>15}  trả về mảng các phần tử không thỏa điều kiện arr>15 trong mảng arrs

MAP

Trả ra mảng mới với các phần tử đã được thao tác trong function

arr = [1,2,3,4,5]

arr.map {|a| a\*2}  mảng mới là [2,4,6,8,10]

1. HASHES

Hash chứa các keys chỉ vị trí của các value

Hash có dạng h= {“name”=>”Duy”,”gender”=>”male”,”age”=>”21”}

Để khai báo : h = Hash.new or h = {}

Hiển thị các khóa : h.keys

Hiển thị các giá trị : h.values

Hiển thị giá trị khóa xác định: h[“name”]

Hiển thị khóa qua giá trị xác định : h.key(“Duy”)

Hiễn thị hashes dễ nhìn : require “awesome\_print”

THAO TÁC VỚI HASHES

* h.has\_key?(“name”)  kiểm tra key có trong hash h hay không , trả về true fale
* h.each {|key,value| puts “….”}
* h.each\_key {|key|}
* h.each\_value {|value|}
* h.select {|key,value| key.size > 5} trả về hash
* h.reject {|key,value| key.size > 5} trả về hash
* h.find {|key,vlue | key >5 } trả về mảng, chỉ 1 thg đầu tìm thấy
* h.find\_all {|key,vlue | key >5 } trả về mảng, tất cả thg tìm thấy
* h.all? {|key,value| key.size >5} tra ve trueflase
* h.any? {|key,value| key.size >5} tra ve trueflase
* h.map {}….

1. Câu điều kiện

Ngoài if… elseif …..else..end và unless….else..end

Đặc biệt

Vd : puts “Dung ” if x==100  puts nếu như x bằng 100

Puts “Dung” unless x==100  làm nếu x không bằng 100

Cả 2 cách viết trên 1 dòng

Cấu trúc case…when..

Vd1

X = 10

case x

when 0..4 #xét giá trị x từ 0 đến 4

puts “tu 0 den 4”

when 5

puts “bang 5”

when 6

puts “ bang6”

else

puts “Khong biet”

end

Vd2

X = 10

case x

when x>=0 && x<=4 #xét giá trị x từ 0 đến 4

puts “tu 0 den 4”

when x==5

puts “bang 5”

when x==6

puts “ bang6”

else

puts “Khong biet”

end

1. Vòng lặp

Cấu trúc while …end

Vd

X=1

while x<=10

puts x

x+=1

end

Cấu trúc until..end

Vd

X=1

until x>=10 #condition sai thì làm

puts x

x+=1

end

Cấu trúc for in…end

Vd 1

array= [1,2,3,4,5,6,7]

for i in array  I chạy qua các phần tử trong array

Puts “i”

end

Vd2

for i in 0..5 do  Chay qua số lần biết trức qui dinh

puts i

end

Cú pháp next if điều hướng

vd

for i in 0..5 do

next if i==2  bỏ qua i=2

puts i

end

Cú pháp break if

vd

for i in 0..5 do

break if i==2  Khi i=2 thì break

puts i

end

CÁC CÚ PHÁP ĐẶC BIỆT

* 5.times { do sth} : lập 5 lần
* 10.downto(1) { do sth } : lập từ 10 đến 1
* (2..8).each { do sth} : lập từ 2 đến 8
* loop do

do sth….

end

vd:

loop do

puts “ nhap mat khau “

matkhau = gets.chomp

if(matkhau == “abc”)

end

1. HÀM

Khai báo hàm

def tênham()

do sth

return

end

vd:

def dientich(a=0,b=0)  nếu ko có đối số thì sẽ dung a b mặc định 0

return a\*b

end

1. LỚP

attr\_accessor : get va set

attr\_reader : get

attr\_writter : set

Cấu trúc

Vd

class Nguoi

attr\_accessor :name, :id, :gender, :nangluong

def initialize(name,id,gender)

@name = name

@id = id

@gender = gender

end

def thucday

@nangluong = 100

end

def an

@nangluong += 5

end

def lamviec

@nangluong -= 5

end

end

nhatduy = Nguoi.new(“DuyPhan”,”0013”,”male”)

KẾ THỪA TRONG RUBY

Vd :

class Giaovien < Nguoi

attr\_accessor :idGV

def initialize(name,id,gender,idGV)

@name = name

@id = id

@gender = gender

@idGV = idGV

End

def to\_s

puts “Hello”

end

end

1. Cài đặt sử dụng thư viện khác trong ruby

* Rubu-toolbox
* rubyGem

vd :

require “httpclient” # yêu cầu thư viện

cr = HTTPClient.new #khoi tao HTTPclient

cr.post “ link ” { :user => “abc”, :pass => “xyz”} #gui thong tin user va pass đến link

rs = cr.get”link” #lấy thông tin trong link

puts rs.body #tra về body của rs

1. BLOCKS

Block trong ruby giúp gọi các câu lệnh

Vd

def testFunction (&block)  block đầu vào

if block\_given?  xem có block được truyền ko

block.call  gọi block

else

puts “ko tim thấy block”

end

end

testFunction{puts “Hello”} block được truyền qa {}

vd2

def testFunction(name) nuế gọi bằng yield ko gần đầu vào block

if block\_given?

yield name  Hiển thị yield như 1 block và name

else

puts “Khng co lock”

end

end

testFunction(“Duy”) {|name| puts”Hello #{name}”} gọi hàm với đầu vào name = Duy và block thể hiện name

1. Proc và Lambda

Proc

n = [1,2,3]

double\_proc = Proc.new { |item\_block| item\_block\*2 }

puts n.map(&double\_proc)

Lambda

n = [1,2,3]

double\_lambda = lambda{ |item\_block| item\_block\*2 }

puts n.map(&double\_lamda)

Cả 2 tương tự nhau : chỉ khác nhau là

Proc cho phép gọi nhiều đối số đầu vào và ko dung tới

Lambda không cho phép gọi đối số đầu vào và không dung tới

Trong 1 phương thức nếu cả 2 đều gọi có block gọi return thì

Proc xem đó là return và return cái method dg đứng

Lambda thì không xem đó là return và ko trả về return

1. FILE

Đọc File.read “ten6file.txt”

Ghi File.write “tenfile.txt”,”Noidung”

Sửa tên File.rename “tenfile.txt”,”tenxua”

Xóa file File.delete“tenfile.txt”

Hiển thị các phương thức của file File.method