Hubert Test1

1.

Theo em hiểu là tất cả các giá trị, các thực thể đều biểu diễn dưới dạng đối tượng. Trong ruby mọi thứ như array, string, hash, number, module đều là đối tượng. Thì mỗi đối tượng thuộc về 1 lớp.
Em ví dụ: number = 30 → number.class -> kết quả là Integer (Đối tượng lớp Integer) …

Riêng đối với block thì không theo nguyên tắc này

2.
my_hash = { name: “this is my name”, ‘age’ => 15 } ?
để lấy giá trị name thì ta làm làm như sau: **my_hash [:name]**

3. sự khác nhau của class method và instance method

Theo như em hiểu thì: **Phương thức class** là phương thức của lớp. Lớp đó có thể truy cập vào method đó. Nhưng đối tượng thuộc lớp đó không thể truy cập được. Và được khai báo dưới dạng có self.method_class
vd: def self.say_hello
…
end
Còn phương thức instance (method thể hiện): Đc hiểu nó là phương thức thuộc các đối tượng. Nó được gọi trên 1 đối tượng cụ thể của một class. Nhưng class thì không và được khai báo không có từ khóa **self** phía trước.

4. sự khác nhau của phương thức map và each
map: Nó là 1 phương thức dùng để ánh xạ các phần tử trong block và trả về 1 mảng mới, có độ dài bằng với độ dài ban đâu. Thường được sử dụng trong việc thực hiện 1 tính toán, thay đổi nào đó trong các phần tử. Map nó cũng giống như phương thức collect trong ruby
vd: array = [2,3,5,12,4]
array.map {|num| num*2}
kết quả: [4,6,10,24,8]

each: Được hiểu nó như dùng để lặp qua và thực hiện xử lý các phần tử bên trong block đó và kết quả trả về không tạo ra 1 mảng mới như map.

vd: array = [2,3,5,12,4]

array.each {|num| print num}
#kết quả: 2 3 5 12 4

5. sự khác nhau của string và symbol
String: thường được sử dụng để biểu diễn biến dưới dạng chuỗi hoặc các kí tự.
vd: name = “Hubert”
Symbol: giá trị của nó thường được đại diện cho tên, các symbol không phải là string
vd: name = :hubert
Một điểm khác nữa là string thì giá trị có thể thay đổi được, còn symbol thì không

6. **sự khác nhau của lambda và proc**
Nó được hiểu như là method ẩn, trả về giá trị và được lưu vào 1 biến cụ thể.
Em thấy sự khác nhau chính của lambda và proc là cách sử lí tham số truyền vào.
ở lambda thì sẽ kiểm tra xem đối số truyền vào có trùng khớp với tham số đã định nghĩa ban đầu hay không, nếu số lượng không khớp sẽ gây ra lỗi. Còn proc thì không xét, nó sẽ bỏ qua việc đó.
Về cú pháp thì lambda: sử dụng từ khóa lambda hoặc —> còn proc thì sử dụng Proc.new hoặc proc.
cú pháp lambda:
tính tổng 2 số
sum = lambda {|a,b| a+b}
sum.call(1,2)
# kết quả: 3

sum = Proc.new {|a,b| a+b}
sum.call(1,2)
# kết quả: 3

7. Sự khác nhau của include và extend
Thì ở ruby thì không có tính chất đa thừa kế. Cũng không có interface như những ngôn ngữ khác (java, c#).
Theo như em hiểu thì nó sử dụng để bao gồm, hay mở rộng phương thức, đối tượng.

Đối với Include: Có nghĩa là chúng ta có 1 module chứa các phương thức, biến … và em muốn sử dụng các phương thức từ module này. Thì ở class đó

em include module đó vào. Từ đó các phương thức của module đó sẽ trở thành các phương thức instance của class đó. Và các đối tượng thuộc class đó có thể truy cập vào. Bởi vì đây là method instance nên là class đó không thể truy cập trực tiếp được.

Còn đối với extend thì nó có khác thì nó là mở rộng phương thức. Có nghĩa là ở **class đó em sử dụng extend từ module(extend bên trong class đó luôn)** thì các phương thức của module sẽ trở thành các phương thức class thì class đó có thể truy cập vào các phương thức đó. Nhưng mà các đối tượng thì không truy xuất được các phương thức đó.

Có thể mở rộng thêm với đối tượng
ví dụ: em có module A (có phương thức x, y, z ..)
person = Class_Name.new
person.extend(A)
person.A

extend không ảnh hưởng đến đối tượng mà chỉ mở rộng chức năng của một đối tượng cụ thể.