codecademy

Arrays and Hashes

Ruby Hash

In Ruby, a *hash* is a collection of key-value pairs.

A *hash* is denoted by a set of curly braces (`{}`) which contains key-value pairs separated by commas. Each value is assigned to a key using a hash rocket (`=>`). Calling the *hash* followed by a key name within brackets grabs the value associated with that key.

```
profile = {
  "name" => "Magnus",
  "profession" => "chess player",
  "ranking" => 1,
  "grandmaster?" => true
}

# "name", "profession", "ranking", and
"grandmaster?" are the keys. "Magnus",
"chess player", 1 and true are the values.

puts profile["name"] # => Magnus
```

Ruby Array

In Ruby, an *array* is an ordered collection of Ruby objects separated by commas and enclosed in `[]` . An *array* can contain the same or different types of Ruby objects, such as Integers, Strings, Floats, etc. An *array* can also be empty.

```
numbers = [1, 2, 3, 4, 5]
#An array of Integers

words = ["See", "Spot", "run"]
#An array of Strings

mixed = ["hello", 5, true, 3.0]
#An array with a String, Integer, Boolean,
and Float

empty = []
#An empty array
```

Ruby Hash New

In Ruby, a *hash* can be created through literal notation (because we are literally assigning what `key=>value` pairs we want in the hash) or by assigning a variable equal to `Hash.new` which generates a new, empty hash.

```
#Creating a hash through literal notation:
lunch = {
  "protein" => "chicken",
  "greens" => "lettuce",
  "organic?" => true
}

#Creating a hash through Hash.new
lunch = Hash.new
puts lunch # => {}
```

Ruby Hash Bracket Notation Adding Pairs

In Ruby, a new key-value pair can be added to a *hash* using bracket notation. The new key is bracketed after the name of the hash and then the value is assigned after the equals sign.

```
#Bracket notation applies to any hash,
regardless of how it was initialized
teammates = Hash.new
teammates["forward"] = "Messi"

salary = {
  "starting" => 40000
}
salary["mid-level"] = 60000
```

Ruby Multidimensional Arrays

In Ruby, arrays can be nested within one another representing multi dimensional arrays. An array can hold another array as if it was like any other Ruby object, such as an Integer or a String.

```
multi_array = [[0,1,2,3],[4.5, true,
"hi"]]

# Accessing the array at index 1
puts multi_array[1] # => [4.5, true, "hi"]

# Accessing the element at index 0 within
the array at index 1
puts multi_array[1][0] # => 4.5
```

Ruby Array Index

In Ruby, each item inside of an array is at a numbered position called an *index*. The first item is at index 0, the second item is at index 1, and so on. We can access the `i` th element of an array by putting the index in square brackets after invoking the array's name; this is known as *access by index*

```
example = ["Car", "Boar", 45, 9.9, true]

#For an array named `example`, you can retrieve an item of a particular index by referencing its index.

puts example[2] # => 45
puts example[0] # => Car
```

Ruby Method .Each

In Ruby, the `.each` method is used to iterate over arrays and hashes. This allows each element in an array and each key-value pair in a hash to be iterated.

```ruby
#In this example, the each method iterates over every color in the colors array and prints it to the console.

colors = ["red", "blue", "green", "yellow"]

colors.each { |color| puts color }
#Output
#red
#blue
#green
#yellow

#When iterating over hashes, two placeholder variables are needed to represent each key/value pair.

polygons = {
  "pentagon" => 5,
  "hexagon" => 6,
  "nonagon" => 9
}

polygons.each do |shape, sides|
  puts "A #{shape} has #{sides} sides."
end
#Output
#A pentagon has 5 sides.
#A hexagon has 6 sides.
#A nonagon has 9 sides.
```

Ruby Hash Bracket Notation Value

In Ruby, the values in a *hash* can be accessed using bracket notation. After the *hash* name, type the key in square brackets in order to access the value.

```
my_love = {
  "dog" => "Keanu",
  "breed" => "Shiba Inu",
  "age_in_years" => 1,
}

puts my_love["breed"] # => Shiba Inu
```

Print Share