In Ruby, a hash is a collection of key-value pairs, also known as associative arrays or dictionaries in other programming languages. Hashes provide a way to store and access data based on a unique key rather than an index, allowing for efficient retrieval and lookup of values.

Hashes in Ruby are defined using curly braces `{}` or the `Hash.new` constructor. Here's an example of creating a hash:

# Using curly braces

person = { "name" => "John", "age" => 30, "city" => "New York" }

# Using Hash.new

person = Hash.new

person["name"] = "John"

person["age"] = 30

person["city"] = "New York"

```

In the example above, we create a hash named `person` with three key-value pairs: "name" => "John", "age" => 30, and "city" => "New York". The keys in the hash are strings, and the corresponding values can be of any data type.

You can access the values in a hash using the square bracket notation with the key:

puts person["name"] # Output: John

puts person["age"] # Output: 30

puts person["city"] # Output: New York

```

You can also modify or add new key-value pairs in a hash by assigning a value to a specific key:

person["age"] = 31 # Update the value for the "age" key

person["occupation"] = "Engineer" # Add a new key-value pair

```

To iterate over the keys and values of a hash, you can use various iteration methods like `each`, `each\_key`, `each\_value`, or `each\_pair`:

person.each do |key, value|

puts "#{key}: #{value}"

end

```

Hashes in Ruby are unordered, meaning the order of the key-value pairs is not guaranteed. If you need to maintain the order, you can use the `OrderedHash` class from the `active\_support` gem or use the `Hash` class with Ruby 1.9 or later, which preserves the insertion order.

Overall, hashes are a powerful data structure in Ruby that provide a flexible way to store and retrieve data based on unique keys.