1. #!/usr/bin/ruby
3. puts 'Hello everyone'
4. puts "Hello everyone"

Accessing string elements

You can access Ruby string elements in different parts with the help of square brackets []. Within square brackets write the index or string.

**Example:**

1. #!/usr/bin/ruby
3. msg = "This tutorial is from JavaTpoint."
5. puts msg["JavaTpoint"]
6. puts msg["tutorial"]
8. puts msg[0]
10. puts msg[0, 2]
11. puts msg[0..19]
12. puts msg[0, msg.length]
13. puts msg[-3]

Multiline string

Writing multiline string is very simple in Ruby language. We will show three ways to print multi line string.

* String can be written within double quotes.
* The % character is used and string is enclosed within / character.
* In heredoc syntax, we use << and string is enclosed within word STRING.

**Example:**

1. puts "
2. A
3. AB
4. ABC
5. ABCD"
7. puts %/
8. A
9. AB
10. ABC
11. ABCD/
13. puts <<STRING
14. A
15. AB
16. ABC
17. ABCD
18. STRING

Variable Interpolation

Ruby variable interpolation is replacing variables with values inside string literals. The variable name is put between #{ and } characters inside string literal.

**Example:**

1. #!/usr/bin/ruby
3. country = "India"
4. capital = "New Delhi"
6. puts "#{capital} is the capital of #{country}."

**next →← prev**

# Ruby Arrays

Ruby arrays are ordered collections of objects. They can hold objects like integer, number, hash, string, symbol or any other array.

Its indexing starts with 0. The negative index starts with -1 from the end of the array. For example, -1 indicates last element of the array and 0 indicates first element of the array.

## Creating Ruby Arrays

A Ruby array is created in many ways.

* Using literal constructor []
* Using new class method

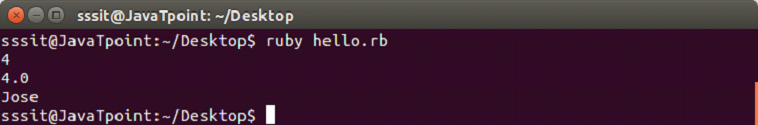
## Using literal construct []

A Ruby array is constructed using literal constructor []. A single array can contain different type of objects.

For example, following array contains an integer, floating number and a string.

1. exm = [4, 4.0, "Jose", ]
2. puts exm

Output:



## Using new class method

A Ruby array is constructed by calling ::new method with zero, one or more than one arguments.

**Syntax:**

1. arrayName = Array.new

To set the size of an array,

**Syntax:**

1. arrayName = Array.new(10)

Here, we have mentioned that array size is of 10 elements.

To know the size of an array, either size or length method is used.

**Example:**

1. #!/usr/bin/ruby
3. exm = Array.new(10)
4. puts exm.size
5. puts exm.length
6. days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
7. puts days[0]
8. puts days[10]
9. puts days[-2]
10. puts days[2, 3]
11. puts days[1..7]
12. days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
13. puts days.fetch(10, "oops")

Removing Items from Array

Ruby array elements can be removed in different ways.

* pop
* shift
* delete
* uniq

insert

Using insert, a new element can be added at any position in an array. Here, first we need to mention the index number at which we want to position the element.

**Example:**

1. days = ["Fri", "Sat", "Sun"]
2. puts days.insert(2, "Thursday")

pop

Using pop, items can be removed from the end of an array. It returns the removed item.

**Example:**

1. days = ["Fri", "Sat", "Sun"]
2. puts days.pop

**first and last method**

The first and last method will return first and last element of an array respectively.

**Example:**

1. days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
2. puts days.first
3. puts days.last

**slice method**

The slice method works similar to #[] method.

**fetch method**

The fetch method is used to provide a default value error for out of array range indices.

**Example:**

1. days = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"]
2. puts days.fetch(10)

shift

Using shift, items can be removed from the start of an array. It returns the removed item.

**Example:**

1. days = ["Fri", "Sat", "Sun"]
2. puts days.shift

unshift

Using unshift, a new element can be added at the beginning of an array.

**Example:**

1. days = ["Fri", "Sat", "Sun"]
2. puts days.unshift("Today")

delete

Using delete, items can be removed from anywhere in an array. It returns the removed item.

**Example:**

1. days = ["Fri", "Sat", "Sun"]
2. puts days.delete("Sat")

uniq

Using uniq, duplicate elements can be removed from an array. It returns the remaining array.

**Example:**

1. days = ["Fri", "Sat", "Sun", "Sat"]
2. puts days.uniq

Cases I

**str.capitalize**

Return a copy of str capitalized # "string" => "String"

**str.downcase**

Return a lowercase copy of str # "STRING" => "string"

**str.swapcase**

Return a swapped case copy of str # "String" => "sTRING"

**str.upcase**

Return a uppercase copy of str # "string" => "STRING"

Search / Replace I

**str.end\_with?(str2)**

Returns true if str ends with str2

**str.include?(str2)**

Return true if str includes str2

**str.index(str2)**

Return index of first occurrence of str2 in str

**str.rindex(str2)**

Return the index of the last occurrence of str2 in str

**str.start\_with?(str2)**

Return true if str starts with str2

Modification I

**str + str2**

Return a new string adding str2 to str # "one" + "two" => "onetwo"

**str << str2**

Modify str in place by appending str2 to the end

**str[index] = str2**

Replace str at index with str2

**str[start,length] = str2**

Replace str from start to length with str2

**str[regexp] = str2**

Replace str using regexp with str2

**str.chomp**

Return a new string with carriage return characters removed from str # "hello\r\n" => "hello"

**str.chop**

Return a new string with last character removed from str # "string" => "strin"

**str.insert(index, str2)**

Insert str2 before the character at the given index, modifying str

**str.succ**

Return successor to str (increment characters starting from rightmost alphachar in string) # "abcd" => "abce"

**str.prepend(str2)**

Prepend str2 to str

**str.reverse**

Return a new string with the characters from str in reverse order # "abcd" => "dcba"

Substrings I

**str \* integer**

Copy - return a new String containing integer copies of str.

**str[start,length]**

Element Reference - pass a start and a length, return a substring of length characters from start

**str[regexp]**

Element Reference - pass a Regexp and return the matching portion of the str

**str[range]**

Element Reference - pass a range using beginning and end as offsets delimiting the substr to be returned

**str.chars**

Return an array of characters in str

**str.chr**

Return a one-character string at the beginning of str

**str.lines**

Return an array of lines in str

**str.split(str2)**

Return array of substrings by dividing str based on str2

Whitespace I

**str.center(width, padstr)**

Center str in width padded with padstr

**str.ljust(length, str2)**

Return a new string of length with str left justified and padded with str2

**str.lstrip**

Return a copy of str with leading whitespace removed

**str.lstrip!**

Modify str with leading whitespace removed

**str.rjust(length, str2)**

Return a new string of length with str right justified and padded with str2

**str.rstrip**

Return a copy of str with trailing whitespace removed

**str.rstrip!**

Modify str by removing trailing whitespace

**str.strip**

Return a copy of str with leading and trailing whitespace removed

**str.strip!**

Modify str by removing leading and trailing whitespace

Comparison I

**str <=> str2**

Compare - return -1, 0, +1 or nil depending on str is less than, eq to, or greater than str2

**str == obj**

Equality - return whether string is equal to obj

**str =~ obj**

Match - if obj is a Regexp, use it as a pattern to match against str, and return the position.

**str.eql?(str2)**

Return true if str is equal to str2 by length and content

Conversion I

**str.hash**

Return a hash based on str's length, content, and encoding

**str.hex**

Return number of str as hexadecimal digits

**str.oct**

Return octal number of str

**str.to\_c**

Convert str to complex number

**str.to\_f**

Convert str to float

**str.to\_i**

Convert str to integer

**str.to\_r**

Convert str to rational number

**str.to\_sym**

Convert str to symbol

Search / Replace II

**str.gsub(regexp, str2)**

Return a copy of str with all occurrences of regexp substituted with str2

**str.gsub(regexp, other\_hash)**

Return a copy of str with all occurrences of regexp substituted with values of matched keys in other\_hash

**str.gsub(str1, str2)**

Return a copy of str with all occurrences of str1 replaced with str2

**str.gsub!(str1, str2)**

Modify str by replacing all occurrences of str1 with str2

**str.index(regexp)**

Return index of first occurrence of regexp in str

**str.match(pattern)**

Convert pattern to regexp and invoke its match method on str

**str.partition(str2)**

Search str2 in str and return part before it, the match, and part after in an array

**str.partition(regexp)**

Search regexp in str and return part before it, the match, and the part after it in an array

**str.sub(regexp, str2)**

Return copy of str with first occurrence of regexp replaced by str2

Cases II

**str.capitalize!**

Modify str by converting the first character to uppercase and the remainder to lowercase

**str.casecmp(str2)**

Case-insensitive compare of str with str2

**str.downcase!**

Modify str by replacing all uppercase letters with lowercase letters

**str.swapcase!**

Modify str by swapping case of all characters

**str.upcase!**

Modify string by replacing lowercase characters with uppercase characters

Search / Replace III

**str.replace(str2)**

Replace the contents and taintedness of str with str2

**str.rpartition(str2)**

Search str2 in str from end of str and return part before it, the match, and part after in an array

**str.scan(regexp)**

Return an array of strings matching regexp of str

**str.sub!(regexp, str2)**

Modify str by replacing first occurrence of regexp with str2

**str.tr(from\_str, to\_str)**

Return copy of str with characters in from\_str replaced by characters in to\_str

**str.tr!(from\_str, to\_str)**

Translate str in place by replacing from\_str with to\_str

**str.tr\_s(from\_str, to\_str)**

Translate str replacing from\_str with to\_str then remove duplicate characters in translated regions

Inspection I

**str[index]**

Element Reference - pass a single index, and return a substring of one character at that index

**str.ascii\_only?**

Return true for a str which has only ASCII characters

**str.count(str2)**

Count of str2 characters in str

**str.empty?**

Return true if str has a length of zero

**str.encoding**

Return the encoding of str

**str.inspect**

Return a printable version of str, surrounded by quote marks, with special characters escaped

**str.length**

Return character length of str (option 1)

**str.size**

Return the character length of str (option 2)

**str.sum**

Return a basic checksum of the characters in str

**str.valid\_encoding?**

Returns true if str is encoded correctly

Iteration I

**str.each\_byte**

Pass each byte in str to a block

**str.each\_char**

Pass each character in str to a block

**str.each\_codepoint**

Pass Integer ordinal of each character in str to a block

**str.each\_line**

Pass each line of string to a block

**str.upto(str2)**

Iterate through successive values of str, starting at str and ending at str2 inclusive

Modification II

**str % arg**

Format - use str as a format specification, and return the result of applying it to arg

**str.chomp!**

Modify str in place by removing carriage return characters from end of str

**str.chop!**

Modify str in place by removing last character

**str.clear**

Make str empty

**str.delete(str2)**

Return a copy of str with all characters of str2 deleted

**str.delete!(str2)**

Modify str by deleting all characters of str2 set

**str.succ!**

Modify str by incrementing characters starting from right most aphanumeric char (successor)

**str.reverse!**

Modify str by reversing it in place

**str.scrub(char)**

Replace invalid bytes with given replacement char

**str.squeeze**

Return copy of str where runs of the same character are replaced by a single character

Conversion II

**str.b**

Return a copied string whose encoding is ASCII-8BIT

**str.codepoints**

Return an array of the Integer ordinals of the characters in str

**str.crypt(str2)**

Apply a one-way cryptographic hash to str with given salt str2

**str.dump**

Produce a version of str with all non-printing characters replace and all special characters escaped

**str.encode(str2)**

Return a copy of str transcoded to encoding str2

**str.intern**

Return symbol corresponding to str, creating the symbol if it did not previously exist

**str.ord**

Return integer ordinal of one-character string

**str.unpack(str2)**

Return array of decoded str using str2 as format

Bytes I

**str.bytes**

Return an array of bytes in str

**str.bytesize**

Return the length of str in bytes

**str.byteslice(index)**

Byte Reference - return a substring of one byte at index of str

**str.byteslice(start, length)**

Byte Reference - return substring from start to length of str

**str.byteslice(range)**

Byte Reference - return substring containing bytes at offsets given by range of str

**str.getbyte(index)**

Return the indexth byte of str as an integer

**str.setbyte(index, integer)**

Modify the indexth byte as integer