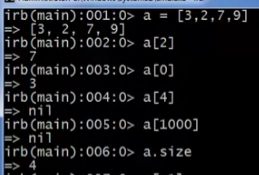
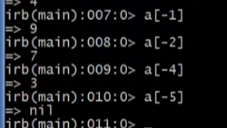
Ruby Arrays

* Lots of special syntax and many provided methods for the Array class
* Can hold any number of other objects, indexed by a number
  + Get via a[i]
  + Set via a[i] = e
* Compared to arrays in many other languages
  + More flexible and dynamic
  + Fewer operations are errors
  + Less efficient
* “The standard collection” (like lists were in ML and Racket)

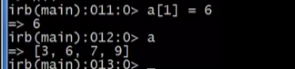
Examples:



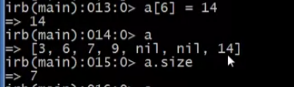
* Out of bounds index will not result to an error
  + Will only return ‘nil’



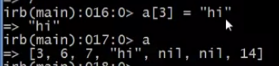
* But using negative numbers as index will not return nil
  + It will return the element counting from the end to the start of the array



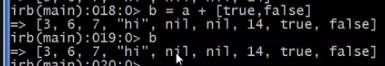
* Updating an element inside the array



* Assigning to an out of bounds index will assign to that index and populate the skipped indices with ‘nil’ in order to make room to a[6]

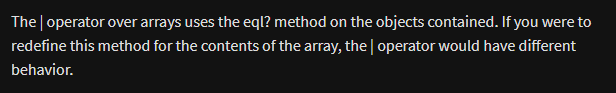


* We can put any type/class



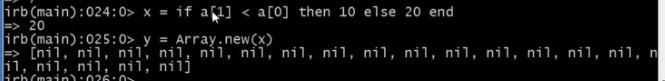
* We can use the + operator in arrays
  + It will append the operands according to the position
    - First operand on the first elements
    - Second operand on the last elements



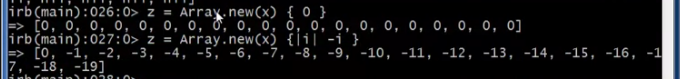
* Pipe operator
  + Like + operator but will remove duplicates
* 



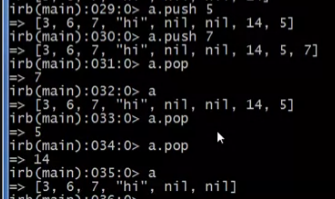
* Creation of triple/tuple …



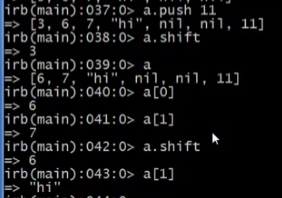
* Creation of new array with predetermined size
  + Array.new(*size*)



* Creation of new array with predetermined size and value
  + Array.new(*size*) { *value/exp* }

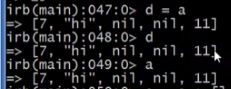


* Creation of stacks
  + Pop
    - Removing the last element of the array
    - Returns the last element removed
    - *Array*.pop
  + Push
    - Addition of an element on the last position in an array
    - Returns the new array
    - *Array*.push *element*

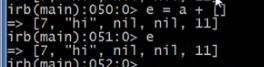


* Creation of queues
  + Shift
    - Removing the first element of the array
    - Returns the first element removed
    - *Array.*shift
  + Push
    - Addition of an element on the last position in an array
    - Returns the new array
    - *Array*.push *element*
  + Unshift
    - Addition of an element on the first position in an array
    - Returns the new array
    - *Array*.push *element*

Array Aliasing

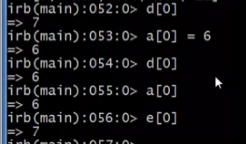


* d and a are aliases here

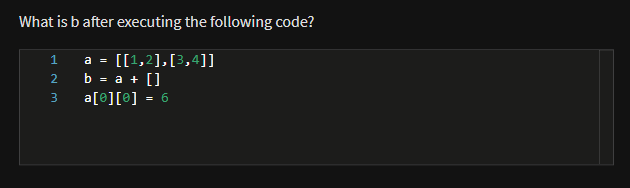


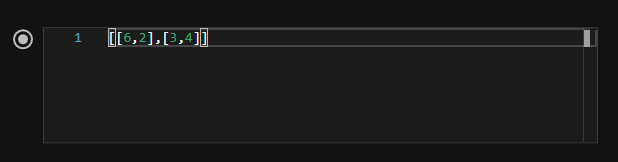
* but e here is not an alias of d and a since it came from a return value of the + operator which is another object
  + even if they have the same elements

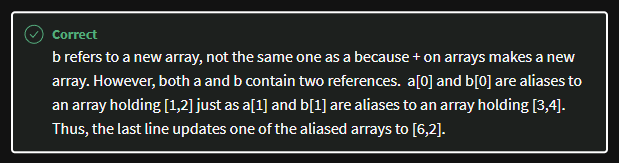
Example:



Strange occurrence in 2 dimensional arrays





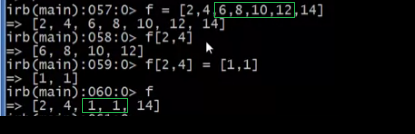


Slicing arrays

* *Array*[*starting* index, *number of elements you want to get*]



* You can also assign/replace the elements that you sliced:



.each

* Loop through elements

