Arrays

Containers
Making Array objects
Accessing elements in an Array
Other operations

Collections

- So far the objects we’ve seen have been simple pieces of data
  - integers
  - floats
  - strings
- Most programs work on groups of objects
- In programming language terminology, groups of objects are containers
- Common types of containers include lists, arrays, matrices, trees, and graphs
- Today: an introduction to arrays in Ruby
- We’ll look at other types of containers throughout the term

Arrays

- An array is an ordered collection of data
- Elements in an array are accessed according to their position
- If there are $n$ elements in an array, their positions are labeled 0 to $n-1$
  - computer scientists always start counting with 0

Making a New Array

- The simplest way to make an Array object in Ruby is to just write a list of data items enclosed in square brackets:
  >> $a = [1, 1, 2, 3, 5, 8]$
  => [1, 1, 2, 3, 5, 8]
  >> $a.class$
  => Array
  >> $a.length$
  => 6

- Ruby doesn’t care about “white space” between elements
  - you can insert spaces and/or tabs
  - you can spread out the array over several lines
Making a New Array (cont’d)

- The Array class has a method named `new` that creates new Array objects
  - every class in Ruby has a method named `new`
  - this method is known as a **constructor**
  - there may be other constructors besides `new`
- To make a new empty array:
  ```ruby
  >> Array.new
  => []
  ```
- To make an array of `n` items:
  ```ruby
  >> Array.new(5)
  => [nil, nil, nil, nil, nil]
  ```
- To make an array of `n` items, each initialized with a value `x`:
  ```ruby
  >> Array.new(5,0)
  => [0, 0, 0, 0, 0]
  ```

Accessing Array Elements

- Items in an array are accessed according to their location
  - an expression used to specify a location is known as an **index**
- In Ruby (and other languages) write an index in square brackets following the name of an array
  ```ruby
  >> a = [1,1,2,3,5,8]
  => [1, 1, 2, 3, 5, 8]
  >> a[0]
  => 1
  >> a[3]
  => 3
  >> a[-1]
  => 8
  ```

Arrays Can Hold Any Object

- Arrays in Ruby can hold any type of object
- Example: an array of Strings
  ```ruby
  >> phrase = ["Hello","World"]
  => ["Hello", "World"]
  >> phrase[0] = "Goodbye"
  => "Goodbye"
  >> phrase
  => ["Goodbye", "World"]
  ```
- In Ruby (but not all languages) arrays can be mixed
  ```ruby
  >> a = [1,"one",2,"two"]
  => [1, "one", 2, "two"]
  ```

Array Bounds

- As shown in a previous example there is a `length` method for arrays that returns the number of elements in an array:
  ```ruby
  >> phrase = ["Hello","World"]
  => ["Hello", "World"]
  >> phrase.length
  => 2
  ```
- What happens if you supply an index that is “out of bounds”?
  ```ruby
  >> phrase[3]
  => nil
  >> phrase[3] = "Adios"
  => "Adios"
  >> phrase
  => ["Hello", "World", nil, "Adios"]
  ```
Extending Arrays

- The array class has a method named `<<` that can be used to append an item to the end of an array.

```ruby
>> phrase = ["Hello", "World"]
=> ["Hello", "World"]
>> phrase << "!
=> ["Hello", "World", "!"]
>> phrase.length
=> 3
```

Read About Array Methods

- There are lots of useful methods for arrays -- read about them in the Thomas book or the online documentation.

```ruby
>> phrase = ["Goodbye", "World"]
=> ["Goodbye", "World"]
>> phrase.insert(1,"Cruel")
=> ["Goodbye", "Cruel", "World"]
>> phrase.join(" - ")
=> "Goodbye - Cruel - World"
>> phrase.reverse
=> ["World", "Cruel", "Goodbye"]
>> phrase.sort
=> ["Cruel", "Goodbye", "World"]
```

A Note on Terminology

- Since arrays hold any type of object, we can make an array of arrays.
  - A simple array is a 1D array or a vector.
  - An array of arrays is a 2D array or a matrix.

```ruby
>> a = [ [0,1,2], [3,4,5], [6,7,8] ]
=> [[0, 1, 2], [3, 4, 5], [6, 7, 8]]
>> a[1]
=> [3, 4, 5]
>> a[1][1]
=> 4
```