Sun*

Array

Outline

Sun*

1. Array introduction

- ❖ Arrays are ordered collections of objects, integer-indexed.
- ❖ Array class inherits from **Object** class and includes **Enumerable** module.
- ❖ Array Index starts at 0.
- ❖ A negative index is assumed to be relative to the end of the array.

Sun*

2. Creating array

- Using the literal constructor []

```
arr = [1, "two", 3.0]          # => [1, "two", 3.0]
```

- Using class method

```
arr = Array.new                # => []
Array.new [8, 9]               #=> [8, 9]
Array.new 3                    # => [nil, nil, nil]
Array.new 3, true              # => [true, true, true]
Array.new 3, "hello"           # => ["hello", "hello", "hello"]
```

- Other

```
animals = %w(monkey dog cat)   # => ["monkey", "dog", "cat"]
```

Sun*

3. Accessing array

Using the :[] method:

```
> arr = [1, 2, 3, 4, 5, 6]
> arr[2]                      # => 3
> arr[100]                    # => nil
> arr[-3]                     # => 4
> arr[2, 3]                   # => [3, 4, 5]
> arr[0, 0]                   # => []
> arr[1..4]                   # => [2, 3, 4, 5]
> arr[1..-3]                  # => [2, 3, 4]
```

Using other methods

```
> arr = [1, 2, 3, 4, 5, 6]
> arr.at 0                    # => 1
> arr.first                   # => 1
> arr.last                    # => 6
> arr.take 3                  # => [1, 2, 3]
```

4. Get information about an array

Using other methods (continue)

```
> numbers = ["one", "two", "three", "four"]
> numbers.length                              # => 4
> numbers.empty?                              # => false
> numbers.include? "ten"                      # => false
```

Sun*

5. Array manipulation

Items can be added to the end of an array by using either **push** or **<<**

```
> arr = [1, 2, 3, 4]
> arr.push 5                                       # => [1, 2, 3, 4, 5]
> arr << 6 << 7 << 8                               # => [1, 2, 3, 4, 5, 6, 7, 8]
> a1 = [1]; a2 = [2, 3]; a3 = [4, 5, [6, 7]]
> a = a1 << a2 << a3                               # => [1, [2, 3], [4, 5, [6, 7]]]
> a[1]                                             # => [2, 3]
> a[1][0]                                          # => 2
> a[2][2][0]                                       # => 6
> a.flatten                                        # => [1, 2, 3, 4, 5, 6, 7]
```

Sun*

5. Array manipulation

❖ **unshift** will add new items to the beginning of an array.

```
➔   arr.unshift -1, 0              # => [-1, 0, 1, 2, 3, 4, 5, 6]
```

❖ With **insert** you can add a new element to an array at any position.

```
➔   arr.insert 3, "apple"          # => [0, 1, 2, 'apple', 3, 4, 5, 6]
```

❖ The method **pop** removes the last element in an array and returns it.

```
➔   arr =  [1, 2, 3, 4, 5, 6]
➔   arr.pop                        # => 6
➔   arr                            # => [1, 2, 3, 4, 5]
```

❖ To retrieve and at the same time remove the first item, use **shift.**

```
➔   arr.shift                      # => 1
➔   arr                            # => [2, 3, 4, 5]
```

❖ To delete an element at a particular index.

```
➔   arr.delete_at(2)               # => 4
➔   arr                            # => [2, 3, 5]
```

Sun*

6. Iterating over array

- ❖ Array has **each** method, which includes from Enumerable module.

```
arr = [1, 2, 3, 4, 5]
```

```
➔    arr.each {|e| puts e}                # => print: 1 2 3 4 5
```

- ❖ Sometimes useful iterator is **reverse_each** which will iterate over the elements in the array in reverse order.

```
arr = [1, 2, 3, 4, 5]
```

```
➔    arr.reverse_each {|e| puts e}        # => print: 5 4 3 2 1
```

- ❖ The **map** method can be used to create a new array based on the original array.

```
arr = [1, 2, 3, 4, 5]
➔    arr.map {|a| 2 * a}                  # => [2, 4, 6, 8, 10]
➔    arr                                  # => [1, 2, 3, 4, 5]
➔    arr.map! {|a| a ** 2}                # => [1, 4, 9, 16, 25]
➔    arr                                  # => [1, 4, 9, 16, 25]
```

Sun*

7. Other array methods

- compact
- concat
- index
- count
- sample
- select
- shuffle
- uniq
- ...

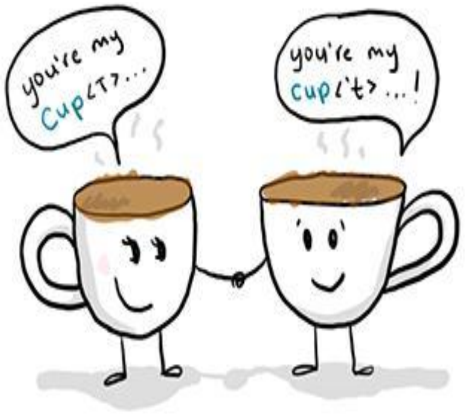
References

- http://ruby-doc.org/core-3.1.0/Array.html
- http://zetcode.com/lang/rubytutorial/arrays/
- https://github.com/awesome-academy/RubyExample_TFW

Sun*

Question & Answer?

THANK YOU


you're my cup<T>...
you're my cup<'t>...!