

[Courseware \(/courses/BerkeleyX/CS169.1x/2013\\_Spring/courseware/\)](/courses/BerkeleyX/CS169.1x/2013_Spring/courseware/)[Course Info \(/courses/BerkeleyX/CS169.1x/2013\\_Spring/info/\)](/courses/BerkeleyX/CS169.1x/2013_Spring/info/)[Syllabus \(/courses/BerkeleyX/CS169.1x/2013\\_Spring/syllabus/\)](/courses/BerkeleyX/CS169.1x/2013_Spring/syllabus/)[Textbook & VM \(/courses/BerkeleyX/CS169.1x/2013\\_Spring/textbook\\_vm/\)](/courses/BerkeleyX/CS169.1x/2013_Spring/textbook_vm/)[Tutorials & Resources \(/courses/BerkeleyX/CS169.1x/2013\\_Spring/tutorials\\_resources/\)](/courses/BerkeleyX/CS169.1x/2013_Spring/tutorials_resources/)[Discussion \(/courses/BerkeleyX/CS169.1x/2013\\_Spring/discussion/forum/\)](/courses/BerkeleyX/CS169.1x/2013_Spring/discussion/forum/)[Wiki \(/courses/BerkeleyX/CS169.1x/2013\\_Spring/course\\_wiki/\)](/courses/BerkeleyX/CS169.1x/2013_Spring/course_wiki/)[Progress \(/courses/BerkeleyX/CS169.1x/2013\\_Spring/progress/\)](/courses/BerkeleyX/CS169.1x/2013_Spring/progress/)

## HW 1-5: ADVANCED OOP, METAPROGRAMMING, OPEN CLASSES AND DUCK TYPING

In lecture, we saw how `attr_accessor` uses metaprogramming to create getters and setters for object attributes on the fly.

Define a method `attr_accessor_with_history` that provides the same functionality as `attr_accessor` but also tracks every value the attribute has ever taken. The following example shows the basic behavior of the new accessor:

```
class Foo
  attr_accessor_with_history :bar
end

f = Foo.new      # => #<Foo:0x127e678>
f.bar = 3        # => 3
f.bar = :wowzo   # => :wowzo
f.bar = 'boo!'   # => 'boo!'
f.bar_history    # => [nil, 3, :wowzo, 'boo!']
```

Here are some hints and guidelines to get you rolling:

- The first thing to notice is that if we define `attr_accessor_with_history` in class `Class`, we can use it as in the snippet above. This is because, as ELLS mentions, a class in Ruby is simply an object of class `Class`. (If that makes your brain hurt, don't worry about it for now. It'll come eventually.)
- The second thing to notice is that Ruby provides a method `class_eval` that takes a string and evaluates it in the context of the current class, that is, the class from which you're calling `attr_accessor_with_history`. This string will need to contain a method definition that implements a `setter-with-history` for the desired attribute `attr_name`.
- `#bar_history` should always return an `Array` of elements, even if no values have been assigned yet.
- Don't forget that the very first time the attribute receives a value, its history array will have to be initialized.
- Don't forget that instance variables are referred to as `@bar` within getters and setters, as explained in Section 3.4 of ELLS.
- Although the existing `attr_accessor` can handle multiple arguments (e.g. `attr_accessor :foo, :bar`), your version just needs to handle a single argument. However, it should be able to track multiple instance variables per class, with any legal class names or variable names, so it should work if used this way:

```
class SomeOtherClass
  attr_accessor_with_history :foo
  attr_accessor_with_history :bar
end
```

- History of instance variables should be maintained separately for each object instance. That is, if you do

```
f = Foo.new
f.bar = 1
f.bar = 2
f = Foo.new
f.bar = 4
f.bar_history
```

then the last line should return `[nil, 4]` rather than `[nil, 1, 2, 4]`.

Here is the skeleton to get you started:

```
class Class
  def attr_accessor_with_history(attr_name)
    attr_name = attr_name.to_s      # make sure it's a string
    attr_reader attr_name           # create the attribute's getter
    attr_reader attr_name+"_history" # create bar_history getter
    class_eval "your code here, use %Q for multiline strings"
  end
end

class Foo
  attr_accessor_with_history :bar
end
```

Example test case:

```
f = Foo.new
f.bar = 1
f.bar = 2
f.bar_history # => if your code works, should be [nil, 1, 2]
```

[Choose Files](#) No file chosen

**Check**

Show Discussion

**New Post**



[Find Courses \(/courses\)](/courses)   [About \(/about\)](/about)   [Blog \(http://blog.edx.org/\)](http://blog.edx.org/)   [Jobs \(/jobs\)](/jobs)   [Contact \(/contact\)](/contact)



(<http://youtube.com/user/edxonline>)



(<https://plus.google.com/108235383044095082735>)



(<http://www.facebook.com/EdxOnline>)



(<https://twitter.com/edXOnline>)

---

© 2013 edX, some rights reserved.

[terms of service \(/tos\)](/tos)

[privacy policy \(/privacy\)](/privacy)

[honor code \(/honor\)](/honor)

[help \(/help\)](/help)