

[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 1-6: ADVANCED OOP, METAPROGRAMMING, OPEN CLASSES AND DUCK TYPING

**Part A — Currency conversion (ELLS 3.11):** Extend the currency-conversion example from lecture so that code such as the following will work:

- `5.dollars.in(:euros)`
- `10.euros.in(:rupees)`

You should support the currencies `dollars`, `euros`, `rupees`, `yen` where the conversions are:

- `rupees` to `dollars`, multiply by 0.019,
- `yen` to `dollars`, multiply by 0.013,
- `euro` to `dollars`, multiply by 1.292.

Both the singular and plural forms of each currency should be acceptable, e.g. `1.dollar.in(:rupees)` and `10.rupees.in(:euro)` should work.

You can use the code shown in lecture as a starting point if you wish. It is repeated below:

```
class Numeric
  @@currencies = {'yen' => 0.013, 'euro' => 1.292, 'rupee' => 0.019}
  def method_missing(method_id)
    singular_currency = method_id.to_s.gsub(/s$/, '')
    if @@currencies.has_key?(singular_currency)
      self * @@currencies[singular_currency]
    else
      super
    end
  end
end
```

**Part B — Palindromes:** Adapt your solution from the "palindromes" question so that instead of writing `palindrome?("foo")` you can write `"foo".palindrome?` (Hint: this should require fewer than 5 lines of code.)

**Part C — Palindromes again:** Adapt your palindrome solution so that it works on `Enumerables`. That is:

```
[1,2,3,2,1].palindrome? # => true
```

It's not necessary for the collection's elements to be palindromes themselves--only that the top-level collection be a

palindrome. (Hint: this should require fewer than 5 lines of code.) Although hashes are considered `Enumerables`, your solution does not need to work with hashes, though it should not error.

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>)