

Ruby - Bug #13237

Behavior for #dup and #clone on Rational/Complex/BigDecimal differs from Integer/Float

02/20/2017 08:52 PM - stomar (Marcus Stollsteimer)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		
ruby -v:	ruby 2.4.0p0 (2016-12-24 revision 57164) [x86_64-linux]	Backport: 2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN
Description		
Since the implementation of feature #12979, #dup and #clone on Integer and Float do not raise a TypeError anymore, and silently return self. Rational and Complex still raise an exception. I'm not sure whether this inconsistent behavior is intended or only an oversight. I guess all Numeric classes should behave in a similar way? Additionally, what is the intention regarding the returned object for non-immediate numeric values, should they return self or a new object? At the time being, BigDecimal (which already did allow #dup/#clone before the change) returns a new object, while Bignum integers return self. Current behavior: <pre>RUBY_VERSION # => "2.4.0" 1.dup # => 1 1.clone # => 1 1.5.dup # => 1.5 1.5.clone # => 1.5 Rational(1).dup rescue \$! # => #<TypeError: can't copy Rational> Rational(1).clone rescue \$! # => #<TypeError: can't copy Rational> Complex(1).dup rescue \$! # => #<TypeError: can't copy Complex> Complex(1).clone rescue \$! # => #<TypeError: can't copy Complex> require "bigdecimal" BigDecimal(1).dup # => 0.1e1 BigDecimal(1).clone # => 0.1e1 d = (1<<64) [d.object_id, d.dup.object_id, d.clone.object_id] # => [5134140, 5134140, 5134140] d = BigDecimal(1) [d.object_id, d.dup.object_id, d.clone.object_id] # => [5133040, 5132900, 5132840]</pre> Old behavior: <pre>RUBY_VERSION # => "2.3.3" 1.dup rescue \$! # => #<TypeError: can't dup Fixnum> 1.clone rescue \$! # => #<TypeError: can't clone Fixnum> 1.5.dup rescue \$! # => #<TypeError: can't dup Float> 1.5.clone rescue \$! # => #<TypeError: can't clone Float> Rational(1).dup rescue \$! # => #<TypeError: can't copy Rational> Rational(1).clone rescue \$! # => #<TypeError: can't copy Rational> Complex(1).dup rescue \$! # => #<TypeError: can't copy Complex> Complex(1).clone rescue \$! # => #<TypeError: can't copy Complex> require "bigdecimal" BigDecimal(1).dup # => #<BigDecimal:101e270,'0.1E1',9(27)></pre>		

```
BigDecimal(1).clone # => #<BigDecimal:101dfa0,'0.1E1',9(27)>
```

Related issues:

Has duplicate Ruby - Feature #13985: Avoid exception for #dup/#clone on Ratio...

Closed

Associated revisions

Revision 31ef3124a9db534abcc3e323f5d3cb696eda3bf5 - 02/22/2017 02:02 AM - nobu (Nobuyoshi Nakada)

numeric.c: Numeric#clone and #dup

- numeric.c (num_clone, num_dup): no longer raises TypeError, returns the receiver instead as well as Integer and Float. [ruby-core:79636] [Bug #13237]
- object.c (rb_immutable_obj_clone): immutable object clone with freeze optional keyword argument.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57682 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 31ef3124 - 02/22/2017 02:02 AM - nobu (Nobuyoshi Nakada)

numeric.c: Numeric#clone and #dup

- numeric.c (num_clone, num_dup): no longer raises TypeError, returns the receiver instead as well as Integer and Float. [ruby-core:79636] [Bug #13237]
- object.c (rb_immutable_obj_clone): immutable object clone with freeze optional keyword argument.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57682 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 02/22/2017 02:02 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset r57682.

numeric.c: Numeric#clone and #dup

- numeric.c (num_clone, num_dup): no longer raises TypeError, returns the receiver instead as well as Integer and Float. [\[ruby-core:79636\]](#) [Bug [#13237](#)]
- object.c (rb_immutable_obj_clone): immutable object clone with freeze optional keyword argument.

#2 - 02/22/2017 04:17 PM - matsuda (Akira Matsuda)

Will this be backported to 2.4? (IOW is this 2.4.0 bug or 2.5 new feature?)

#3 - 10/19/2017 11:09 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Feature #13985: Avoid exception for #dup/#clone on Rational and Complex added