# Ruby master - Bug #18187

**Float#clamp() returns ArgumentError (comparison of Float with 1 failed)**

09/22/2021 05:23 PM - SouravGoswami (Sourav Goswami)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td>ruby 3.0.2p107 (2021-07-07 revision 0db68f0233) [x86_64-linux]</td>
</tr>
<tr>
<td>Backport:</td>
<td>2.6: UNKNOWN, 2.7: UNKNOWN, 3.0: UNKNOWN</td>
</tr>
</tbody>
</table>

**Description**

When I have a Float::NAN as a number, I expect all the method to work properly.

For example, Float::NAN - 1 gives NAN. But Float::NAN.to_i raises FloatDomainError.

But in case of clamp(), Float::NAN.clamp(0, 100) returns ArgumentError (comparison of Float with 1 failed)

This error doesn't explain what's actually wrong. I didn't write the comparison to compare Float with 1. I didn't pass any invalid argument either. This error is a reflection of what's going on in the C level, which shouldn't appear to the user.

If I write a vanilla clamp() in ruby:

```ruby
Float.define_method(:clamp2) { |min, max| self < min ? min : self > max ? max : self }
```

In this case, I can call it like this:

```ruby
> 8.0.clamp2(10, 100)
=> 10
> 80.0.clamp2(10, 100)
=> 80.0
> 800.0.clamp2(10, 100)
=> 100
> Float::NAN.clamp2(10, 100)
=> NaN
```

As you can see, it just returns NAN. But in case of the built-in clamp, it raises the ArgumentError, even though my arguments are just correct. So this should handle this clamp() correctly, either returning the min value or Float::NAN.

**History**

**#1 - 09/22/2021 09:04 PM - mame (Yusuke Endoh)**

On my machine, the code raises comparison of Float with 0 failed, instead of ... with 1 failed. I have no idea where 1 comes from.

```ruby
$ ruby -ve 'Float::NAN.clamp(0, 100)'
ruby 3.0.2p107 (2021-07-07 revision 0db68f0233) [x86_64-linux]
-e:1:in `clamp': comparison of Float with 0 failed (ArgumentError)
```

BTW, I have no opinion about what Float::NAN.clamp(0, 100) should return or raise.

**#2 - 09/22/2021 09:30 PM - SouravGoswami (Sourav Goswami)**

Hi, sorry, yes it's comparison of Float with 0 failed, probably there was some typo.

I agree it should raise or return. But it shouldn't raise ArgumentError. Anyway, it probably has the lowest priority because it doesn't cause any issue so far.

The best would be returning Float::NAN? Because that's the behaviour you get when you write comparison in Ruby.

Apparently it will waste some CPU time on isnan(RFLOAT_VALUE(self)) though...

**#3 - 08/26/2022 05:55 PM - jeremyevans0 (Jeremy Evans)**

I submitted a pull request to make Float::NAN.clamp return the receiver: [https://github.com/ruby/ruby/pull/4884](https://github.com/ruby/ruby/pull/4884)
However, like @mame (Yusuke Endoh), I'm not sure if it is more desirable to raise in this case.

#4 - 10/19/2021 04:48 AM - mrkn (Kenta Murata)
I think it's OK to return NaN for all the cases of Float::NAN.clamp.

#5 - 10/19/2021 11:12 AM - nobu (Nobuyoshi Nakada)
What about Float#clamp?

diff --git i/numeric.c w/numeric.c
index db2b2eb2793..12edb0f6006 100644
--- i/numeric.c
+++ w/numeric.c
@@ -2844,6 +2844,13 @@ num_step(int argc, VALUE *argv, VALUE from)
     return from;
 }
+static VALUE
+flo_clamp(int argc, VALUE *argv, VALUE x)
+{
+    if (isnan(RFLOAT_VALUE(x))) return x;
+    return rb_call_super(argc, argv);
+}
+
+static char *
+out_of_range_float(char (*pbuf)[24], VALUE val)
+{
@@ -5789,6 +5796,7 @@ Init_Numeric(void)
         rb_define_method(rb_cFloat, "finite?", rb_flo_is_finite_p, 0);
         rb_define_method(rb_cFloat, "next_float", flo_next_float, 0);
         rb_define_method(rb_cFloat, "prev_float", flo_prev_float, 0);
+        rb_define_method(rb_cFloat, "clamp", flo_clamp, -1);
     }

#6 - 10/21/2021 08:11 AM - matz (Yukihiro Matsumoto)
I vote for keeping NaN raises exceptions.

Matz.

#7 - 10/21/2021 02:14 PM - jeremyevans0 (Jeremy Evans)
- Status changed from Open to Rejected