

Ruby - Feature #18915

New error class: NotImplementedYetError or scope change for NotImplementedError

07/14/2022 01:39 PM - Quintasan (Michał Zająć)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	

Description

Abstract

Introduce NotImplementedYetError exception that should be used in case when the codepath has not been implemented by the developer for some reason (maybe they're designing an abstract class or are designing some sort of interface to reuse later on) OR extend the meaning of NotImplementedError to cover those usecases so we don't have to introduce another exception

Background

NotImplementedError is supposed to be raised if the underlying operating system or Ruby runtime does not support them (<https://ruby-doc.org/core-3.1.2/NotImplementedError.html>)

However it appears that many people are misusing this exception by raising this in a superclass from which they later inherit from. I do realize that Ruby promotes duck-typing (the default RuboCop style guide has a cop for this – <https://github.com/rubocop/ruby-style-guide#duck-typing>). However I have seen this being discussed numerous times:

- <https://github.com/rubocop/ruby-style-guide/issues/458>
- <http://chrisstump.online/2016/03/23/stop-abusing-notimplementederror/>
- <https://oleg0potapov.medium.com/ruby-notimplementederror-dont-use-it-dff1fd7228e5>
- <https://gitlab.com/gitlab-org/gitlab/-/issues/354314> (which I'm the author of)
- <https://github.com/rmosolgo/graphql-ruby/issues/2067> (here the author actually confused it with Python's NotImplementedError)
- <https://stackoverflow.com/questions/13668068/how-to-signal-not-implemented-yet>

Proposal

Create NotImplementedYetError exception

OR

Allow raising NotImplementedError in cases other than OS or Ruby runtime incompatibilities

Evaluation

Add NotImplementedYetError

I think a new exception is a better idea than changing the usage of an existing one just because "everyone is using it". That said it would require people to refactor their code which might prevent wider adoption of the new exception.

Change scope of NotImplementedError

This would require the least amount of changes possible (only a documentation change) and I believe there would be no compatibility problems whatsoever.

History

#1 - 11/15/2022 04:10 PM - Quintasan (Michał Zająć)

- Subject changed from New error class: NotImplementedYetError or scope change for NotImplementedYet to New error class: NotImplementedYetError or scope change for NotImplementedError

#2 - 11/15/2022 04:11 PM - Quintasan (Michał Zająć)

- Description updated

#3 - 11/15/2022 10:14 PM - austin (Austin Ziegler)

I think that a PR for a documentation change on this would probably be accepted.

#4 - 10/02/2023 12:51 PM - zdennis (Zach Dennis)

- *File not-implemented-error-docs.patch added*

Attached is a patch to update the documentation for NotImplementedError to expand its scope beyond just missing features on the underlying platform.

The current documentation has been in place since just before Ruby 1.9.2. Historically, it appears that Ruby hasn't limited its usage of NotImplementedError to errors that originate from features missing from the underlying platform (e.g. system calls) since at least 1.8.0. I believe the example provided in the current NotImplementedError documentation was meant as an example, not to suggest the only scope in which this error can be used.

Ruby itself invalidates most of the interpretation provided in the Background section of this ticket. Here are just three examples:

- [numeric.c](#) raises NotImplementedError when a class needs provide its own `<=>` method.
- [delegate.rb](#) raises NotImplementedError to indicate that a `getobj` and `setobj` need to be provided by a subclass.
- [tsort](#) uses this error and documents it expressly as: Should be implemented by a extended class.

There are more examples to be found in Ruby as well.

Outside of Ruby, this pattern is also commonly used as a way to communicate the intent to an application, framework, or library developer. Here are four examples from popular projects:

- [rails uses this pattern](#)
- [minitest uses this pattern](#)
- [rspec uses this pattern](#)
- [dry-rb libraries use this pattern](#)

Additionally, even the latest Programming Ruby 3.2 book by PragPub uses an example of this in Chapter 6: Inheritance and Messages.

#5 - 10/02/2023 01:12 PM - zdennis (Zach Dennis)

- *File not-implemented-error-docs.patch added*

Whoops, last patch upload failed. Patch actually applied here.

#6 - 10/02/2023 03:03 PM - Anonymous

#7 - 10/03/2023 06:29 AM - nobu (Nobuyoshi Nakada)

- *File deleted (not-implemented-error-docs.patch)*

#8 - 10/13/2023 04:19 PM - tenderlovemaking (Aaron Patterson)

I'm guilty of using NotImplementedError for abstract classes. I like the idea of changing the documentation, but on the other hand, since NotImplementedError doesn't inherit from StandardError I kind of wish there was a new exception class.

#9 - 10/13/2023 04:32 PM - Quintasan (Michał Zając)

tenderlovemaking (Aaron Patterson) wrote in [#note-8](#):

I'm guilty of using NotImplementedError for abstract classes. I like the idea of changing the documentation, but on the other hand, since NotImplementedError doesn't inherit from StandardError I kind of wish there was a new exception class.

Hence my proposal to introduce a new class. I didn't want to create a PR for either solution because I would welcome both of them. I'm slightly leaning towards a new exception class for this.

#10 - 10/13/2023 05:06 PM - byroot (Jean Boussier)

since NotImplementedError doesn't inherit from StandardError I kind of wish there was a new exception class

Could you elaborate? For this use case I think not inheriting from StandardError is better, as it's not something you'd want to be rescued broadly.

#11 - 10/13/2023 06:27 PM - tenderlovmaking (Aaron Patterson)

byroot (Jean Boussier) wrote in [#note-10](#):

since NotImplementedError doesn't inherit from StandardError I kind of wish there was a new exception class

Could you elaborate? For this use case I think not inheriting from StandardError is better, as it's not something you'd want to be rescued broadly.

It is something I would like rescued broadly. For example if I'm test driving a concrete implementation, the test framework needs to specifically rescue NotImplementedError in order to report the error. I wouldn't expect a test framework to rescue OutOfMemoryError for example, but definitely rescue NotImplementedError.

#12 - 10/13/2023 06:35 PM - byroot (Jean Boussier)

I wouldn't expect a test framework to rescue OutOfMemoryError for example,

Well Minitest does rescue Exception:

<https://github.com/minitest/minitest/blob/6719ad8d8d49779669083f5029ea9a0429c49ff5/lib/minitest/test.rb#L196>

Pretty sure RSpec does as well. It's one of the rare justifiable cases for doing so.

#13 - 11/08/2023 12:37 AM - mame (Yusuke Endoh)

Discussed at the dev meeting. This is tentative, but [@matz \(Yukihiro Matsumoto\)](#) said:

- I don't want to change the documentation of NotImplementedError.
- NotImplementedYetError is too confusing with NotImplementedError.
- ToBeDefinedError or something would be better.
- I will reply to the ticket.

Please wait for matz's answer.

#14 - 12/20/2023 07:17 AM - matz (Yukihiro Matsumoto)

[#note-13](#) explains my opinion well. What name candidate do you have?

Matz.

#15 - 12/20/2023 08:47 AM - Quintasan (Michał Zając)

matz (Yukihiro Matsumoto) wrote in [#note-14](#):

[#note-13](#) explains my opinion well. What name candidate do you have?

Matz.

How about AbstractMethodError - the same as Java?

#16 - 12/22/2023 03:02 PM - Dan0042 (Daniel DeLorme)

How about raise NoMethodError, "method '#{__method__}' is not implemented in #{self.class}"

#17 - 12/22/2023 03:28 PM - byroot (Jean Boussier)

How about raise NoMethodError

Agreed. If people want an exception that inherits from StandardError, then NoMethodError perfectly fits the bill.

I still think that "you forgot to implement this abstract method" shouldn't inherit from StandardError though, as it risk getting handled by the application and not surface during testing, hence why NotImplementedError already works great for that use case regardless of what the documentation says.

Amusingly Python has the same error class for that purpose exactly: <https://docs.python.org/3/library/exceptions.html#NotImplementedError>

exception NotImplementedError

This exception is derived from RuntimeError. In user defined base classes, abstract methods should raise this exception when they require

derived classes to override the method, or while the class is being developed to indicate that the real implementation still needs to be added.

#18 - 04/10/2024 03:34 AM - nithinbekal (Nithin Bekal)

What name candidate do you have?

What do you think about the name SubclassResponsibilityError? As @citizen428 explains [here](#):

Smalltalk had the idiom of implementing abstract methods with the body self subclassResponsibility which raises an error.

The name gives a pretty clear indication of what is wrong, and it seems fitting considering Ruby's Smalltalk heritage.

#19 - 04/19/2024 01:39 AM - Dan0042 (Daniel DeLorme)

It's a bit off-topic but does anyone know why NotImplementedError doesn't inherit from StandardError? It seems like it should. If the system doesn't support fork() then I'd like to see that as a nice message via Rack::ShowExceptions rather than having to dig through logs. And if you use it for abstract classes then you *definitely* want that displayed by Rack::ShowExceptions

#20 - 04/28/2024 12:01 AM - Quintasan (Michał Zając)

nithinbekal (Nithin Bekal) wrote in [#note-18](#):

What name candidate do you have?

What do you think about the name SubclassResponsibilityError? As @citizen428 explains [here](#):

Smalltalk had the idiom of implementing abstract methods with the body self subclassResponsibility which raises an error.

The name gives a pretty clear indication of what is wrong, and it seems fitting considering Ruby's Smalltalk heritage.

I like this as well as AbstractMethodError

Files

not-implemented-error-docs.patch	1.57 KB	10/02/2023	zdennis (Zach Dennis)
----------------------------------	---------	------------	-----------------------