Python vs. Ruby for metaprogramming [closed]

Asked 16 years, 2 months ago Modified 6 years, 4 months ago

Viewed 35k times

90

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 12 years ago.

I'm currently primarily a D programmer and am looking to add another language to my toolbox, preferably one that supports the metaprogramming hacks that just can't be done in a statically compiled language like D.

I've read up on Lisp a little and I would love to find a language that allows some of the cool stuff that Lisp does, but without the strange syntax, etc. of Lisp. I don't want to start a language flame war, and I'm sure both Ruby and Python have their tradeoffs, so I'll list what's important to me personally. Please tell me whether Ruby, Python, or some other language would be best for me.

Important:

1. Good metaprogramming. Ability to create classes, methods, functions, etc. at runtime. Preferably, minimal distinction between code and data, Lisp style.
2. Nice, clean, sane syntax and consistent, intuitive semantics. Basically a well thought-out, fun to use, modern language.
3. Multiple paradigms. No one paradigm is right for every project, or even every small subproblem within a project.
4. An interesting language that actually affects the way one thinks about programming.

Somewhat important:

1. Performance. It would be nice if performance was decent, but when performance is a real priority, I'll use D instead.
2. Well-documented.

Not important:

1. Community size, library availability, etc. None of these are characteristics of the language itself, and all can change very quickly.
2. Job availability. I am not a full-time, professional programmer. I am a grad student and programming is tangentially relevant to my research.

3. Any features that are primarily designed with very large projects worked on by a million code monkeys in mind.

python ruby lisp metaprogramming

Share

Improve this question

Follow

edited Jul 26, 2018 at 13:03

community wiki
3 revs, 2 users 69%
dsimcha

1 +1 For a generally well-asked question. – Kenan Banks May 6, 2011 at 1:23

Sadly the high-rated answers all basically say "it depends" and the lower answers are basically a Python vs Ruby flame war. – Dan Burton Jun 10, 2011 at 6:31

Have you considered Clojure? – Mark Thomas Jul 3, 2013 at 13:04

The answer is Julia! It just didn't exist back then: bit.ly/julia_meta ...rejoice! Julia covers all the important points listed by @dsimcha, plus **performance** and all the other points are becoming true as Julia keeps on maturing.
– HarmonicaMuse Dec 27, 2015 at 13:07

69

> I've read up on Lisp a little and I would love to find a language that allows some of the cool stuff that Lisp does, but without the strange syntax, etc. of Lisp.

Wouldn't we all.

> minimal distinction between code and data, Lisp style

Sadly, the minimal distinction between code and data and "strange" syntax are consequences of each other.

If you want easy-to-read syntax, you have Python. However, the code is *not* represented in any of the commonly-used built-in data structures. It fails—as most languages do—in item #1 of your 'important' list. That makes it difficult to provide useful help.

You can't have it all. Remember, you aren't the first to have this thought. If something like your ideal language existed, we'd all be using it. Since the real world falls short of your ideals, you'll have to re-prioritize your wish list. The "important" section has to be rearranged to identify what's *really* important to you.

Share Improve this answer Follow

edited Aug 11, 2013 at 15:15

community wiki
2 revs, 2 users 81%
S.Lott

3 I've found that many languages implement Lisp-like macros in non-Lispy syntaxes and what ends happening is that writing macros there is very hard, because nobody naturally knows the data structures in which the code is represented, thus writing macros gets too hard and nobody does it. – Pablo Fernandez Oct 27, 2008 at 15:05

11 People find Lisp hard to read because they're not familiar with the syntax. I find Lisp much easier to read than C# (but harder than Python). – Jules Dec 27, 2008 at 23:43

1 I largely agree with the point, but my experience of both Ruby and Lisp shows that former is as good for metaprogramming as it gets *without* the parentheses. Other alternatives that come close could be TCL and JavaScript but I would not favor them for other reasons. – inger Sep 14, 2011 at 20:33

Long long agi there was a language called Dylan which should be exactly that. – Friedrich Mar 18, 2018 at 12:23

17

Honestly, as far as metaprogramming facilities go, Ruby and Python are a lot more similar than some of their adherent like to admit. This review of both language offers a pretty good comparison/review:

- http://regebro.wordpress.com/2009/07/12/python-vs-ruby/

So, just pick one based on some criteria. Maybe you like Rails and want to study that code. Maybe SciPy is your thing. Look at the ecosystem of libraries, community, etc, and pick one. You certainly won't lose out on some metaprogramming nirvana based on your choice of either.

Share Improve this answer Follow

answered Jul 30, 2009 at 7:48

community wiki
ars

that blog entry seems to have more to do with personal preferences (fair enough but beauty is in the eye of the beholder), rather than metaprogramming - which was the main point of the OP. – inger Sep 14, 2011 at 20:38

16

Disclaimer: I only dabble in either language, but I have at least written small working programs (not just quick scripts, for which I use Perl, bash or GNU make) in both.

Ruby can be really nice for the "multiple paradigms" point 3, because it works hard to make it easy to create domain-specific languages. For example, browse online and look at a couple of bits of Ruby on Rails code, and a couple of bits of Rake code. They're both Ruby, and you

can see the similarities, but they don't look like what you'd normally think of as the same language.

Python seems to me to be a bit more predictable (possibly correlated to 'clean' and 'sane' point 2), but I don't really know whether that's because of the language itself or just that it's typically used by people with different values. I have never attempted deep magic in Python. I would certainly say that both languages are well thought out.

Both score well in 1 and 4. [Edit: actually 1 is pretty arguable - there is "eval" in both, as common in interpreted languages, but they're hardly conceptually pure. You can define closures, assign methods to objects, and whatnot. Not sure whether this goes as far as you want.]

Personally I find Ruby more fun, but in part that's because it's easier to get distracted thinking of cool ways to do things. I've actually used Python more. Sometimes you don't want cool, you want to get on with it so it's done before bedtime...

Neither of them is difficult to get into, so you could just decide to do your next minor task in one, and the one after that in the other. Or pick up an introductory book on each from the library, skim-read them both and see what grabs you.

edited Sep 28, 2008 at 0:20

Share Improve this answer

Follow

community wiki
2 revs
Steve Jessop

15

There's not really a huge difference between python and ruby at least at an ideological level. For the most part, they're just different flavors of the same thing. Thus, I would recommend seeing which one matches your programming style more.

Share Improve this answer

Follow

answered Sep 28, 2008 at 0:31

community wiki
Jason Baker

31 There're most definitely *not* the same thing. The look similar on the surface, but one you exercise the most powerful features of Ruby you understand that Python is just no match. For an example try to write a DSL in Ruby vs writing one Python, or creating function, methods, classes, etc. at run-time. It's much more straight-forward in Ruby. – FelipeC Feb 15, 2010 at 22:01

14 It's not rare that you need to do metaprogramming, it's just rare that it's done. All but the most trivial program have repeating patterns that don't fall to the usual refactoring tools but could be slain readily by metaprogramming.
– Wayne Conrad Feb 22, 2010 at 23:43

10 Ruby and Python are hugely different even at ideas that govern their design. Python they want one and hopefully one obvious way to do things. That generally makes the language not as expressive as Ruby, but it makes it more consistent. Ruby comes a little more from the Perl way of things where there are many ways to do things. Also Ruby makes some things super easy and actually includes the idea of private members. Python on the other hand at the most just makes some thing harder to do, so you have to be more explicit (like adding or overriding behavior on classes). – Sean Copenhaver Sep 13, 2010 at 13:53

5 You should probably do a crash course in both, but for easy metaprogramming it appears that Ruby is more suited. I don't have very much experience though, so take that with a grain of salt. – Sean Copenhaver Sep 13, 2010 at 13:55

7 Ruby and Python are *only* similar in the fact that they preach "beautiful code". They just have totally different views on that beauty thing (which IMO is good) – Gabi Purcaru Jul 13, 2011 at 17:16

15

Have you considered Smalltalk? It offers a very simple, clear and extensible syntax with reflectivity and introspection capabilities and a fully integrated development environment that takes advantage of those capabilities. Have a look at some of the work being done in Squeak Smalltalk for instance. A lot of researchers using Squeak hang out on the Squeak mailing list and #squeak on freenode, so you can get help on complex issues very easily.

Other indicators of its current relevance: it runs on any platform you'd care to name (including the iPhone); Gilad

Bracha is basing his Newspeak work on Squeak; the V8 team cut their teeth on Smalltalk VMs; and Dan Ingalls and Randal Schwartz have recently returned to Smalltalk work after years in the wilderness.

Best of luck with your search - let us know what you decide in the end.

Share Improve this answer Follow

answered Sep 28, 2008 at 8:54

community wiki
mykdavies

14

Lisp satisfies all your criteria, including performance, and it is the only language that doesn't have (strange) syntax. If you eschew it on such an astoundingly ill-informed/wrong-headed basis and consequently miss out on the experience of using e.g. Emacs+SLIME+CL, you'll be doing yourself a great disservice.

Share Improve this answer Follow

answered Sep 29, 2008 at 21:34

community wiki
user23611

4 Or you can try Clojure, which I find very nice.
– Pablo Fernandez Oct 27, 2008 at 15:07

1 Strong agree. If you want the power of Lisp, just dive in and have it! It's actually quite easy to get used to the parens; they're not as big a deal as most people make them out to be. – Dan Burton Jun 10, 2011 at 6:26

11

Your 4 "important" points lead to Ruby exactly, while the 2 "somewhat important" points ruled by Python. So be it.

Share Improve this answer Follow

answered Sep 28, 2008 at 8:36

community wiki
Neo

11

You are describing Ruby.

> - Good metaprogramming. Ability to create classes, methods, functions, etc. at runtime. Preferably, minimal distinction between code and data, Lisp style.

It's very easy to extend *and* modify existing primitives at runtime. In ruby everything is an object, strings, integers, even functions.

You can also construct shortcuts for syntactic sugar, for example with class_eval.

> - Nice, clean, sane syntax and consistent, intuitive semantics. Basically a well thought-out, fun to use, modern language.

Ruby follows the principle of less surprise, and when comparing Ruby code vs the equivalent in other language many people consider it more "beautiful".

> - Multiple paradigms. No one paradigm is right for every project, or even every small subproblem within a project.

You can follow imperative, object oriented, functional and reflective.

> - An interesting language that actually affects the way one thinks about programming.

That's very subjective, but from my point of view the ability to use many paradigms at the same time allows for very interesting ideas.

I've tried Python and it doesn't fit your important points.

Share Improve this answer Follow

answered Sep 29, 2008 at 0:42

community wiki
FelipeC

11 -1 I'm using Python and it fits perfectly, differences between Python and Ruby lays in other aspects. Fanboy-ish noisemaker, you are. – gorsky Jan 25, 2010 at 12:44

6 A lot of talk but no walk. Care to provide an example in Python of adding a method dynamically to say, the String class? – FelipeC Feb 5, 2010 at 20:19

6 @john That's precisely my point; it's very complicated and ugly (not to mention *impossible* for the String class). OTOH in Ruby it's very simple: "self.class.send(:define_method, :method_name) { method_code }" – FelipeC Mar 3, 2010 at 18:49

2 Ruby may *try* to follow the PoLS, but I wouldn't say it *does*. For example, the `lambda` / `Proc.new` mess has been called "surprising behavior" and "highly counterintuitive" here on SO. :-) Any language as big and complex as Ruby is bound to have such confusing areas. – Ken Aug 26, 2010 at 14:32

2 @Token here's an example of monkey-patching `String` in Ruby for metaprogramming purposes: coldattic.info/shvedsky/pro/blogs/a-foo-walks-into-a-bar/posts/…. Subclassing wouldn't do; however, a simple two-argument function would. – P Shved May 19, 2011 at 22:32

8

Compare code examples that do the same thing (join with a newline non-empty descriptions of items from a `myList` list) in different languages (languages are arranged in reverse-alphabetic order):

Ruby:

```
myList.collect { |f| f.description }.select { |d| d !=
```

Or

```
myList.map(&:description).reject(&:empty?).join("\n")
```

Python:

```
descriptions = (f.description() for f in mylist)
"\n".join(filter(len, descriptions))
```

Or

```
"\n".join(f.description() for f in mylist if f.descrip
```

Perl:

```
join "\n", grep { $_ } map { $_->description } @myList
```

Or

```
join "\n", grep /./, map { $_->description } @myList;
```

Javascript:

```
myList.map(function(e) e.description())
      .filter(function(e) e).join("\n")
```

Io:

```
myList collect(description) select(!="") join("\n")
```

Here's an Io guide.

Share Improve this answer

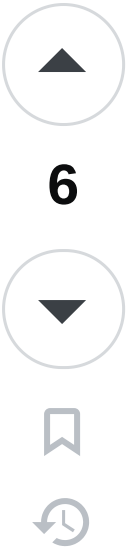
Follow

edited Dec 28, 2008 at 10:39

community wiki
2 revs
J.F. Sebastian

3 (format nil "~{~a~^~%~}" (remove nil (mapcar #'description mylist))) – Rainer Joswig Oct 16, 2010 at 14:33

nice, but where is metaprogramming here? it seems to be some slightly functional style, remotely related to the question. – inger Sep 14, 2011 at 20:25



Ruby would be better than Lisp in terms of being "mainstream" (whatever that *really* means, but one realistic concern is how easy it would be to find answers to your questions on Lisp programming if you were to go with that.) In any case, I found Ruby very easy to pick up. In the same amount of time that I had spent first learning Python (or other languages for that matter), I was soon writing *better* code much *more efficiently* than I ever had before. That's just one person's opinion, though; take it with a grain of salt, I guess. I know much more about Ruby at this point than I do Python or Lisp, but you should know that I was a Python person for quite a while before I switched.

Lisp is definitely quite cool and worth looking into; as you said, the size of community, etc. can change quite quickly. That being said, the size itself isn't as important as the *quality* of the community. For example, the `#ruby-lang` channel is still filled with some incredibly smart people. Lisp seems to attract some really smart people too. I can't speak much about the Python community as I don't have a lot of firsthand experience, but it seems to be "too big" sometimes. (I remember people being quite rude on their IRC channel, and from what I've heard from friends that are really into Python, that seems to be the rule rather than the exception.)

Anyway, some resources that you might find useful are:

1) The Pragmatic Programmers Ruby Metaprogramming series (http://www.pragprog.com/screencasts/v-

dtrubyom/the-ruby-object-model-and-metaprogramming) -- not free, but the later episodes are quite intriguing. (The code is free, if you want to download it and see what you'd be learning about.)

2) On Lisp by Paul Graham (http://www.paulgraham.com/onlisp.html). It's a little old, but it's a classic (and downloadable for free).

Share Improve this answer Follow

answered Jul 29, 2009 at 20:52

community wiki
Benjamin Oakes

6

@Jason I respectively disagree. There are differences that make Ruby superior to Python for metaprogramming - both philosophical and pragmatic. For starters, Ruby gets inheritance right with Single Inheritance and Mixins. And when it comes to metaprogramming you simply need to understand that it's all about the *self*. The canonical difference here is that in Ruby you have access to the *self* object at runtime - in Python you do not!

Unlike Python, in Ruby there is no separate compile or runtime phase. In Ruby, every line of code is executed against a particular *self* object. In Ruby every class inherits from both object and a hidden metaclass. This makes for some interesting dynamics:

```
class Ninja
  def rank
    puts "Orange Clan"
  end

  self.name #=> "Ninja"
end
```

Using *self.name* accesses the Ninja classes' metaclass *name* method to return the class name of Ninja. Does metaprogramming flower so beautiful in Python? I sincerely doubt it!

Share Improve this answer Follow

answered Jan 30, 2011 at 2:13

community wiki
Eric Davidson

5

I am using Python for many projects and I think Python does provide all the features you asked for.

important:

1. Metaprogramming: Python supports metaclasses and runtime class/method generation etc
2. Syntax: Well thats somehow subjective. I like Pythons syntax for its simplicity, but some People complain that Python is whitespace-sensitive.
3. Paradigms: Python supports procedural, object-oriented and basic functional programming.

4. I think Python has a very practical oriented style, it was very inspiring for me.

Somewhat important:

1. Performance: Well its a scripting language. But writing C extensions for Python is a common optimization practice.
2. Documentation: I cannot complain. Its not that detailed as someone may know from Java, but its good enough.

As you are grad student you may want to read this paper claiming that Python is all a scientist needs. Unfortunately I cannot compare Python to Ruby, since I never used that language.

Regards, Dennis

Share Improve this answer Follow

answered Sep 28, 2008 at 8:27

community wiki
xardias

3 Python is not whitespace-sensitive It is indentation-sensitive. – jfs Dec 28, 2008 at 16:16

Well, if you don't like the lisp syntax perhaps assembler is the way to go. :-)

4

It certainly has minimal distinction between code and data, is multi-paradigm (or maybe that is no-paradigm) and it's a mind expanding (if tedious) experience both in terms of the learning and the tricks you can do.

Share Improve this answer Follow

answered Oct 2, 2008 at 16:05

community wiki
rickardg

I think I have done more meta programming in assembly language than in any other language. Code is data and data can try to be code. And data isn't signed or unsigned, it's the opcode that decides. – Nosredna Jul 29, 2009 at 20:13

1 I have assembler code that you could claim to be OO. It has things that look rather like methods - you call the one that depends on the "class". – justintime Oct 21, 2009 at 16:41

4

Io satisfies all of your "Important" points. I don't think there's a better language out there for doing crazy meta hackery.

Share Improve this answer Follow

answered Oct 15, 2008 at 21:09

community wiki
frigolitmonster

Thanks for that, didn't know it. Looks fun at first glance, maybe plenty of (()) but anyway. Hope I'll find the time to have a look, soon. – flq Jul 29, 2009 at 21:19

3

> one that supports the metaprogramming hacks that just **can't be done in a statically compiled language**
>
> I would love to find a language that allows some of the cool stuff that **Lisp** does

Lisp can be compiled.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

community wiki
RHSeeger

3

Did you try Rebol?

Share Improve this answer Follow

answered Sep 28, 2009 at 22:08

community wiki
Anon

3

My answer would be neither. I know both languages, took a class on Ruby and been programming in python for several years. Lisp is good at metaprogramming due to the fact that its sole purpose is to transform lists, its own source code is just a list of tokens so metaprogramming is natural. The three languages I like best for this type of thing is Rebol, Forth and Factor. Rebol is a very strong dialecting language which takes code from its input stream, runs an expression against it and transforms it using rules written in the language. Very expressive and extremely good at dialecting. Factor and Forth are more or less completely divorced from syntax and you program them by defining and calling words. They are generally mostly written in their own language. You don't write applications in traditional sense, you extend the language by writing your own words to define your particular application. Factor can be especially nice as it has many features I have only seen in smalltalk for evaluating and working with source code. A really nice workspace, interactive documents, etc.

Share Improve this answer Follow

answered Feb 2, 2012 at 23:45

community wiki
ToddB

There isn't really a lot to separate Python and Ruby. I'd say the Python community is larger and more mature

2

than the Ruby community, and that's really important for me. Ruby is a more flexible language, which has positive and negative repercussions. However, I'm sure there will be plenty of people to go into detail on both these languages, so I'll throw a third option into the ring. How about JavaScript?

JavaScript was originally designed to be Scheme for the web, and it's prototype-based, which is an advantage over Python and Ruby as far as multi-paradigm and metaprogramming is concerned. The syntax isn't as nice as the other two, but it is probably the most widely deployed language in existence, and performance is getting better every day.

Share Improve this answer Follow

answered Sep 28, 2008 at 3:38

community wiki
Jim

2

If you like the lisp-style code-is-data concept, but don't like the Lispy syntax, maybe Prolog would be a good choice.

Whether that qualifies as a "fun to use, modern language", I'll leave to others to judge. ;-)

Share Improve this answer Follow

answered Sep 28, 2008 at 6:43

community wiki
JesperE

Ruby is my choice after exploring Python, Smalltalk, and Ruby.

2

Share Improve this answer Follow

answered Sep 28, 2008 at 13:29

community wiki
ryw

What about OCaml ?

2

OCaml features: a static type system, type inference, parametric polymorphism, tail recursion, pattern matching, first class lexical closures, functors (parametric modules), exception handling, and incremental generational automatic garbage collection.

I think that it satisfies the following:

> Important:
>
> 2. Nice, clean, sane syntax and consistent, intuitive semantics. Basically a well thought-out, fun to use, modern language.

3. Multiple paradigms. No one paradigm is right for every project, or even every small subproblem within a project.
4. An interesting language that actually affects the way one thinks about programming.

Somewhat important:

1. Performance. It would be nice if performance was decent, but when performance is a real priority, I'll use D instead.
2. Well-documented.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

community wiki
2 revs
Robert Vuković

OCaml allows you to create classes/methods at runtime? How does that work? – Jason C Jul 29, 2009 at 20:30

I just read about OCaml and maybe it can not create stuff at runtime so I have removed it. – Robert Vuković Jul 30, 2009 at 7:39

I've use Python a very bit, but much more Ruby. However I'd argue they both provide what you asked for.

2

If I see all your four points then you may at least check: http://www.iolanguage.com/

And Mozart/Oz may be interesting for you also: http://mozart.github.io/

Regards Friedrich

Share Improve this answer Follow

edited Jan 15, 2017 at 21:20

community wiki
3 revs, 2 users 94%
Friedrich

1

For python-style syntax and lisp-like macros (macros that are real code) and good DSL see converge.

Share Improve this answer Follow

edited Sep 28, 2008 at 13:23

community wiki
2 revs
Kasprzol

1

I'm not sure that Python would fulfill all things you desire (especially the point about the minimal distinction between code and data), but there is one argument in favour of python. There is a project out there which makes it easy for you to program extensions for python in

D, so you can have the best of both worlds.
http://pyd.dsource.org/celerid.html

Share Improve this answer Follow

answered Sep 29, 2008 at 6:10

community wiki
Mauli

---

if you love the rose, you have to learn to live with the thorns :)

1

Share Improve this answer Follow

answered Dec 27, 2008 at 23:07

community wiki
AgentOrange

I would recommend you go with Ruby.

When I first started to learn it, I found it really easy to pick up.

1

Share Improve this answer Follow

edited Dec 22, 2011 at 21:13

community wiki
3 revs, 3 users 67%
Mardix

Do not to mix Ruby Programming Language with Ruby Implementations, thinking that POSIX threads are not possible in ruby.

1

You can simply compile with pthread support, and this was already possible at the time this thread was created, if you pardon the pun.

The answer to this question is simple. If you like lisp, you will probably prefer ruby. Or, whatever you like.

Share Improve this answer Follow

edited Aug 3, 2012 at 0:16

community wiki
2 revs, 2 users 92%
nonanonym

0

I suggest that you try out both languages and pick the one that appeals to you. Both Python and Ruby can do what you want.

Also read this thread.

Share Improve this answer Follow

edited May 23, 2017 at 11:46

community wiki
2 revs
Alexander Kojevnikov

0

Go with JS just check out AJS (Alternative JavaScript Syntax) at my github http://github.com/visionmedia it will give you some cleaner looking closures etc :D

Share Improve this answer Follow

answered May 11, 2009 at 23:42

community wiki
TJ Holowaychuk

0

Concerning your main-point (meta-programming): Version 1.6 of Groovy has AST (Abstract Syntax Tree) programming built-in as a standard and integrated feature. Ruby has RubyParser, but it's an add-on.

Share Improve this answer

edited Jul 29, 2009 at 20:32

Follow

community wiki
2 revs
groovy