GDP - Support #103

Python3 Support

06/11/2017 05:56 PM - Nitesh Mor

Status:	New	
Priority:	Normal	
Assignee:	Nitesh Mor	
Category:	Python API	
Target version:		

Description

(Originally reported by Pat, but deleted accidentally. Restoring it as much as I can)

We were prepping stuff for the signpost workshop and were surprised that the python deb didn't install bindings for Python 3. Py2k's been deprecated for a while, but the bigger issue is it's handling of low-level byte operations. It's tricky to get that correct in py2k, so we try hard to use python3 wherever/whenever possible.

Running the 2to3 tool on the current gdp sources it looks like it's maybe only ~20 lines that need to change to support py3k, and they're all related to imports.

Maintaining both can bit a bit of a burden, but fortunately you can basically turn py2k into py3k. A common header for everything I do that requires interop is to add this to the top of files:

```
# Coerce Py2k to act more like Py3k
from __future__ import (absolute_import, division, print_function, unicode_literals)
from builtins import (
        ascii, bytes, chr, dict, filter, hex, input, int, isinstance, list, map,
        next, object, oct, open, pow, range, round, str, super, zip,
        )
```

The "absolute_import" from that will solve the import differences gdp currently has. I'd recommend just grabbing all of it. Then I find that you can just write py3k code and things just work for both.

History

#1 - 06/11/2017 05:57 PM - Nitesh Mor

It would be nice to have Python bindings work with Python 3. So why don't we have it already? Here's why:

- Shortage of man-power.
- Large number of other Python 2 based sub-projects that depend on such bindings.
- My lack of familiarity with Python 3.

The suggestions for the added header is greatly appreciated. However, I still must ask: how urgent is this, and is there a timeline you have in mind?

#2 - 06/11/2017 05:58 PM - Nitesh Mor

- Description updated

#3 - 06/11/2017 06:36 PM - Nitesh Mor

> (Pat wrote this, which was lying around in the inbox of gdp@berkeley.edu for months)
>
> Shortage of man-power.

I feel that :)

> Large number of other Python 2 based sub-projects that depend on such bindings.

Like other libraries? Or other things that are part of the gdp?

It used to be the case that some of the big libraries hadn't moved over to py3k yet, but that's really not true any more. In general, when I'm writing Python, if I find a library nowadays without Py3k support I avoid it like the plague, it's a huge red flag that the library is out of date and poorly supported. If you have any 3rd party dependencies that are sticking you at py2k I'd strongly recommend replacing them.

> My lack of familiarity with Python 3.

It's not terribly different. The things you immediately notice:

- print is a function not a keyword => effect: you have to call print("foo") instead of just print "foo"
- bytes are a proper type => effect: In py2k bytes() actually returns a string. This is a recipe for tons of subtle bugs and as low-level code authors really strongly motivates py3k over py2k. I'm not sure exactly what your internal gdp<>c interface looks like, but if it handles raw byte arrays you many see some changes here
- unicode string literals => effect: If you want to convert a string to an array of bytes, you have to explicitly encode it (i.e. "foo".encode('utf-8') -> bytes() object). This is actually a really good thing to enforce o/w it's really easy to introduce unicode bugs unintentionally

Ultimately, I think you'll find that moving to Py3k takes at most an afternoon.

> how urgent is this, and is there a timeline you have in mind?

We worked around it for the signpost workshop, but the guy who wrote the interface script had two bugs around handling raw byte arrays that wouldn't have happened in Py3k. It's definitely a big reason that we avoid using the gdp*

- One other being how difficult it is to install on non-debian systems. Would it be feasible to implement a gdp client (and just the client) in a cross-platform language like Python?

- The other big one is that everything (code, documentation) is locked behind login walls. It makes it really hard for us to write any documentation for anything that would use the gdp because we can't link to anything. We can't write software that relies on it because people can't install the gdp.

- Related to this point, we're discussing putting together a Signpost workshop and tutorial for the broader embedded systems community during SenSys this year (November). That tutorial obviously can't include the gdp in its current form, which means we have to plan on non-gdp solutions.

So I guess the answer to the urgency boils down to desired adoption. The limited platform support and login-walls and perceived secrecy tend to make us nervous about really using the GDP, except for demos. So from that perspective it isn't really urgent. If it's really something the TerraSwarm center wants every member to use and make part of their infrastructure, then I'd say making it easily usable and installable (i.e. on PyPI as a pip-installable package) would be the top priority.

#4 - 06/11/2017 06:37 PM - Nitesh Mor

(Again, sent by Pat as an email reply, but got stuck in the gdp@berkeley inbox)

As a follow-up to this, supporting a mixture of Python2 and Python3 broke things on several people's systems during the Signpost workshop.

I rewrote our scripts to be Py2k/3k compatible and workshop attendees just gave up on the gdp.

If you want people to use your python package in 2017, it needs to be pip-installable and it needs to support python3.

#5 - 06/11/2017 09:49 PM - Anonymous

Thanks for resurrecting this issue. The pip deployment ties in with Bug 102, Easier GDP Deployment: shared library issues. It is basically the same issue where libgdp has shared library dependencies that might not be installed.

The alternative is to use the RESTful interface for everything but subscription and a WebSockets or some other interface for subscription.