

Stack Overflow is a community of 4.7 million programmers, just like you, helping each other.

Join them; it only takes a minute:

Sign up

Join the Stack Overflow community to:



Ask programming questions



Answer and help your peers

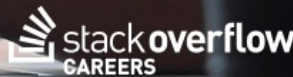


Get recognized for your expertise

Python 3, let json object accept bytes or let urlopen output strings

Work on work you love.

From home.



With Python3 I am requesting from some url a json document.

```
response = urllib.request.urlopen(request)
```

The `response` object is a file like object with `read`, `readline` functions.

Normally a json object can be created with a file (opened in `textmode`)

```
obj = json.load(fp)
```

What I would like to do is:

```
obj = json.load(response)
```

this however does not work as `urlopen` returns a file object in binary mode.

A work around is of course:

```
str_response = response.readall().decode('utf-8')
obj = json.loads(str_response)
```

but this feels bad...

Is there a better way that I can transform a byte file object to a string file object? Or am I missing any parameters for either `urlopen` or `json.load` to give an encoding?

This would look to me as a common use case so I'm confident I'm missing some usefull function.

json encoding python-3.x urlopen

asked Jul 28 '11 at 17:00



Peter Smit

7,879 ● 11 ● 65 ● 124

4 Answers

HTTP sends bytes. If the resource in question is text, the character encoding is normally specified, either by the Content-Type HTTP header or by another mechanism (an RFC, HTML `meta http-equiv ,...`).

`urllib` *should* know how to encode the bytes to a string, but it's too naïve—it's a horribly underpowered and un-Pythonic library.

[Dive Into Python 3](#) provides an overview about the situation.

Your "work-around" is fine—although it feels wrong, it's the correct way to do it.

edited Jun 11 '12 at 16:47

answered Jul 28 '11 at 17:13



Wayne Werner

15.3k ● 6 ● 63 ● 121



Humphrey Bogart

3,032 ● 7 ● 34 ● 54

3 This may be the "correct" way to do it but if there was one thing I could undo about Python 3 it would be this bytes/strings crap. You would think the built-in library functions would at least know how to deal with other built-in library functions. Part of the reason we use python is the simple intuitive syntax. This change breaks that all over the place. – [ThatAintWorking](#) May 2 '14 at 23:22

1 Check out [the "requests" library](#) -- it handles this sort of thing for you automatically. – [offby1](#) Sep 2 '14 at 23:15

- 1 This isn't a case of the built-in library functions needing to "know how" to deal with other functions. JSON is defined as a UTF-8 representation of objects, so it can't magically decode bytes that it doesn't know the encoding of. I do agree that `urlopen` ought to be able to decode the bytes itself since it knows the encoding. Anyway, I've posted the Python standard library solution as an answer — you can do streaming decoding of bytes using the `codecs` module. — [jbg](#) Sep 14 '14 at 1:41

Add  projects to your  **stackoverflow** profile.
CAREERS

Python's wonderful standard library to the rescue...

```
import codecs

reader = codecs.getreader("utf-8")
obj = json.load(reader(response))
```


Works with both py2 and py3.

edited Apr 1 '15 at 19:27

 [sorin](#)
41.6k ● 60 ● 223 ● 375

answered Sep 14 '14 at 1:39

 [jbg](#)
1,401 ● 6 ● 15

- 5 I got this error when trying this answer in `python 3.4.3` not sure why? The error was `TypeError: the JSON object must be str, not 'StreamReader'` — [Aaron Lelevier](#) Aug 5 '15 at 23:52
- 4 @AronYsidoro Did you possibly use `json.loads()` instead of `json.load()` ? — [sleepycal](#) Sep 28 '15 at 13:17
- 1 Fantastic solution, thank you for sharing — [sleepycal](#) Jan 6 at 16:01
- 1 For bonus points, use the encoding specified in the response, instead of assuming utf-8: `response.headers.get_content_charset()` . Returns `None` if there is no encoding, and doesn't exist on python2. — [Phil Frost](#) Mar 21 at 19:26 
- 1 @PhilFrost That's slick. In practice it might pay to be careful with that; JSON is always UTF-8, UTF-16 or UTF-32 by definition (and is overwhelmingly likely to be UTF-8), so if another encoding is returned by the web server, it's possibly a misconfiguration of the web server software rather than genuinely non-standard JSON. — [jbg](#) Mar 22 at 7:02

I have come to opinion that the question is the best answer :)

```
import json
from urllib.request import urlopen

response = urlopen("site.com/api/foo/bar").read().decode('utf8')
obj = json.loads(response)
```

answered Aug 27 '15 at 12:55

 [SergO](#)
469 ● 5 ● 8

Just found this simple method to make `HttpResponse` content as a json

```
import json

request = RequestFactory() # ignore this, this just like your request object
response = MyView.as_view()(request) # got response as HttpResponse object
response.render() # call this so we could call response.content after

json_response = json.loads(response.content.decode('utf-8'))

print(json_response) # {"your_json_key": "your json value"}
```

Hope that helps you

edited Dec 27 '15 at 17:26

answered Dec 27 '15 at 17:20

 [Aditya Kresna Permana](#)
1,595 ● 1 ● 10 ● 14