In python, how do I make a class a generator? [duplicate]

Asked 7 years, 5 months ago Modified 5 years, 7 months ago Viewed 410 times

1

This question already has answers here:

How to build a basic iterator? (10 answers)

Closed 7 years ago.

I'm working with the varnish API in varnish 4. The way it produces log entries is by Dispatch'ing against it, and passing in a callback. Simple version:

```
class vlog(object):
    def __init__(self):
        self.body = []
    def callback(self, vap, cbd, priv):
        self.body.append(cbd)
    def main(self):
        self.vlog = varnishapi.VarnishLog(sopath="libvarnishapi.so")
        while True:
            ret = self.vlog.Dispatch(self.callback)
            if ret == 0:
                time.sleep(0.1)
```

I need to do several more things in self.callback to aggregate the data properly, but whatever.

My REAL question is this: How can I turn the above class into a generator? My ideal usage would be something like this:

```
vlog_inst = vlog()
for log_aggregate in vlog_inst:
    pass
```

Simply putting a 'yield' statement in the callback function never triggers iteration. (surprising to me, adding the yield statement also causes all of my print statements to produce no output as well... I'm obviously missing something.)

python generator

3 Define `__iter__` and `next` (`__next__` in Python 3). There has to be a dup for this.
– Ashwini Chaudhary Aug 5, 2015 at 14:28

Since this is a datasource leveraging a callback, I can't 'restart' iteration with a next() function... the callback has nothing I can give a 'return' data too... and next would force me to 'reconnect' to the log-stream, which would mean I loose data. Also, since the datasource is effectively infinite, generators are a better approach.
– Jason Aug 5, 2015 at 14:51

1 Answer

Sorted by: Highest score (default)

2

You want your class to be *iterable*. A generator is a way to implement an iterable, but that's not required to make a class iterable.

You could implement an `__iter__` method; if you make it a generator function you don't need anything else:

```
def __iter__(self):
    while True:
        yield self.vlog.Dispatch(self.callback)
```

Share Improve this answer Follow

answered Aug 5, 2015 at 14:30

Martijn Pieters ♦
1.0m 282 3947 3283