Different Return Types in Python

I have a function that returns two different types (`bool` and `int`):

```
def limit_reached():
    rate_limit = api.GetRateLimitStatus()
    if not rate_limit['remaining_hits'] == 0:
        return False
    else:
        return rate_limit['remaining_hits']
```

Basically, I want to be able to call the function to see if the "limit" limit has been reached. If it has not, before continuing, print how many `'remaining_hits'` are left.

So far I have this, but can't figure out how to efficiently show `'remaining_hits'` without calling the `GetRateLimitStatus()` function again:

```
if limit_reached():
    print "Limit reached. Exiting"
    exit()
else:
    ##Print remaining hits
    ##Continue...
```

Also, if it helps, I am using the Twitter API for Python

python twitter return-type

edited Jan 12 '12 at 22:29
Rik Poggi
12.6k ●3 ●32 ●53

asked Jan 12 '12 at 21:32
user1001715
21 ●3

2 You do realize that different return types is epically bad design. Also, your logic `rate_limit['remaining_hits'] != 0: return False` means you'll either return False or `0` , which doesn't seem right. Please return one type and use exceptions for exceptional situations. – S.Lott Jan 12 '12 at 21:34

Plus, the `if` in the second fragment is also the wrong way. – Anony-Mousse Jan 12 '12 at 21:36

4 Answers

In python, the integer '0' is identically equivalent to 'False'. So, for any and every truth test in Python, the integer '0' evaluates to false.

For what I see, you can adapt your code to use this fact.

Hope it helps.

answered Jan 12 '12 at 21:37
heltonbiker
7,867 ●8 ●48 ●107

Store the result in a variable?

```
remaining = limit_reached()
if not remaining:
    print "Limit reached."
else:
    print "Remaining:", remaining
```

P.S. you can also return `0` when the limit was reached... No need to return `False` when it actually means `0`.

edited Jan 12 '12 at 22:13

answered Jan 12 '12 at 21:35

Anony-Mousse

36.9k ● 4 ● 56 ● 99

You should redefine your limit_reached function:

```
def limit_reached():
    return api.GetRateLimitStatus()['remaining_hits']
```

then something like:

```
remaining = limit_reached()
if remaining: # or 'remaining > 0' if you want to be explicit
    ##Print remaining hits
    ##Continue...
else:
    print "Limit reached. Exiting"
    exit()
```

answered Jan 12 '12 at 21:44

Gerrat

14.7k ● 4 ● 32 ● 53

As a commenter pointed out, returning different variable types is bad style. It is fairly easy to always return a boolean value, like so:

```
def limit_reached():
    rate_limit = api.GetRateLimitStatus()
    return rate_limit['remaining_hits'] == 0
```

`rate_limit['remaining_hits'] == 0` is a complete statement that will return a 'true' or 'false' value, the result of which you can return from your function. Note that you do need to use two equals signs.

If you need the integer value itself, you can always return that instead, and test the condition in your other code, like so:

```
def limit_reached():
    rate_limit = api.GetRateLimitStatus()
    return rate_limit['remaining_hits']

if limit_reached() == 0:
    print "Limit reached. Exiting"
    exit()
else:
    ##Print remaining hits
    ##Continue...
```

Or, you could take advantage of the fact that all numbers (integer, float, double, it doesn't matter) with an exact value of `0` are treated as `false` (other constants treated as `false` include `[]`, `()`, `''`, or '{}' - see here).

```
if limit_reached():
    ##Print remaining hits
    ##Continue...
else:
    print "Limit reached. Exiting"
    exit()
```

Note that the clauses have been reversed in this case.

answered Jan 12 '12 at 21:49

Hannele

3,661 ● 2 ● 20 ● 47