過去のスレッドのダイジェスト版

Python のお勉強

part17

■委譲（デリゲート）での __getattribute__ の使い方が分りません。(19-39)

```
19 名前：デフォルトの名無しさん 投稿日：2007/02/18( 日 ) 01:54:54
委譲（デリゲート）での __getattribute__ の使い方が分りません。
    従来の __getattr__() は、解決できない属性名の時だけ呼ばれるが
    __getattribute__() は、属性値の取得が必要になったときに、常に呼ばれるらしいのですが。

まず、__getattr__ でのサンプル。

class wrapper(object):
  def __init__(self, obj):
    self.__wrapped = obj # obj を組み込む。
  def __getattr__(self, attrname): # attrname は、知らない名前の属性が来たら呼び出される
    return getattr(self.__wrapped, attrname) # 組み込まれたオブジェクトに getattr 関数で処理を委託。

l = wrapper([]) # リストを引数に wrapper クラスを呼び出すと、リストにできる操作なら全てできる

（実行すると、正しく動作します）
>>> l.append(333)
>>> l.pop(0)
333


20 名前：デフォルトの名無しさん 投稿日：2007/02/18( 日 ) 01:56:08
ここで、__getattr__ を __getattribute__ に置き換えます。

class wrapper(object):
  def __init__(self, obj):
    self.__wrapped = obj # obj を組み込む。
  def __getattribute__(self, attrname):
    return getattr(self.__wrapped, attrname)

l = wrapper([])

（試しに、実行してみます）
>>> l.append(333)
（ここで暴走）

なぜ暴走するのかよく分りません。
__getattribute__ はどう使うのが正しいのでしょうか？


21 名前：デフォルトの名無しさん 投稿日：2007/02/18( 日 ) 02:48:11
>>20
self.__getattribute__ の中で self._wrapped を取得しようとすると
さらに self.__getattribute__(self, '_wrapped') を呼び出して無限再帰ループになるみたい。

解決方法は・・・たとえば、_wrapped をインスタンス属性としてもつんじゃなくて
クラス属性にインスタンスをキーにした辞書をもつようにするとかでどう？

class wrapper(object):
    _wrapped = {}
    def __init__(self, obj):
        wrapper._wrapped[hash(self)] = obj
    def __del__(self):
        del wrapper._wrapped[hash(self)]
    def __getattribute__(self, attrname):
        return getattr(wrapper._wrapped[hash(self)], attrname)


23 名前：デフォルトの名無しさん 投稿日：2007/02/18( 日 ) 03:00:57
__getattr__ を __getattribute__ に置き換えるって話じゃないのか


24 名前：デフォルトの名無しさん 投稿日：2007/02/18( 日 ) 03:08:57
if attrname not in ['_wrapped', etc, etc]:
  return getattr(self._wrapped, attrname)
```

```
else:
  super(wrapper, self).__getattribute(attrname)
```

25 名前：デフォルトの名無しさん 投稿日：2007/02/18(日) 03:14:55
>>21 でおｋかと思ったけど、len() ができないなぁ。

```
>>> l = wrapper([])
>>> len(l)

Traceback (most recent call last):
  File "<pyshell#84>", line 1, in <module>
    len(l)
TypeError: object of type 'wrapper' has no len()
```

TypeError なあたり、決まった型にしか呼べないのかも。
len(x) って内部で x.__len__ を返してるのかと思ってたけど、ちょっと違うのかな。

```
>>> l.append(3)
>>> l.__len__()
1
```

27 名前：デフォルトの名無しさん 投稿日：2007/02/18(日) 04:03:11
>>25
アクセサでフィールド値を取得してから、len しないとダメじゃない？
__getattribute__ じゃなく、__getattr__ の例で悪いけど、
この場合、同じと思う。

```
class wrapper(object):
  def __init__(self, obj):
    self.__wrapped = obj
  def __getattr__(self, attrname):
    return getattr(self.__wrapped, attrname)
  def get_wrapped(self): # フィールド値取得メソッドを設定
    return self.__wrapped

l = wrapper([])
l.append(333)

（ここで len を実行。正常に動作する）
>>> len(l.get_wrapped())
1
>>>
```

28 名前：デフォルトの名無しさん 投稿日：2007/02/18(日) 04:37:24
>>25
__getattribute__ でフックされるのは、dir(l) で出てくるものだけだと思う。
したがって、len は使えないが __len__ は使えるということになる。

```
>>> dir(l)
['__add__', '__class__', '__contains__', '__delattr__',
'__delitem__', '__delslice__', '__doc__', '__eq__', '__ge__',
'__getattribute__', '__getitem__', '__getslice__', '__gt__',
'__hash__', '__iadd__', '__imul__', '__init__', '__iter__',
'__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__reversed__',
'__rmul__', '__setattr__', '__setitem__', '__setslice__',
'__str__', 'append', 'count', 'extend', 'index', 'insert',
'pop', 'remove', 'reverse', 'sort']
>>>
```

30 名前：デフォルトの名無しさん 投稿日：2007/02/18(日) 04:50:46
>>28
なるほど。__len__ が特別だったのか。確かに __init__ とかフックされたら困るよなぁ。
__len__ を定義して、ラップしているオブジェクトの len() を返すようにしたら、ちゃんと動きますた。
質問主じゃないけど勉強になった。この先 __getattribute__ を使うかどうかはともかくとしてｗ

35 名前：デフォルトの名無しさん 投稿日：2007/02/18(日) 12:43:37
>>19 >>20 で __getattribute__ の使い方について質問した者です。

なぜ、__getattr__ ではなく __getattribute__ を使いたいと思ったかというと、
「スーパークラスの配列で，複数の異なるサブクラスのインスタンスを一元管理する」
というテクニックを使えるようにしておきたいと思ったからです。

ちなみに、従来からあるメソッド __getattr__() は、解決できない属性名の時だけ呼ばれるが
__getattribute__() は、属性値の取得が必要になったときに、常に呼ばれる、とのこと。

「スーパークラスの配列で，複数の異なるサブクラスのインスタンスを一元管理する」に
ついては、java の解説ページですが、下のところにあります。
http://itpro.nikkeibp.co.jp/members/NSW/ITBASIC/20050620/162997/

（略）部長さん，課長さん，担当さんをそれぞれ表す Bucho クラス，Kacho クラス，Tanto クラスを
定義し，それぞれのクラスのメンバーとして給与の金額を返す getKyuyo という関数があるとします。
複数のクラスに同じ名前の関数があるのですから，それらを汎化して社員を表す Shain クラスを定義
しましょう。（略）

サブクラスである Bucho クラス，Kacho クラス，Tanto クラスは，0 を返す getKyuyo を継承することに
なります。こんな関数を継承しても役に立ちませんね。スーパークラスから継承した関数の機能が
サブクラスに合わない場合は，サブクラスで同名のメソッドを記述すれば上書き変更できます。
これを「オーバーライド（override)」と呼びます。部長さんの給与は 70 万円，課長さんの給与は
50 万円，担当さんの給与は 30 万円としましょう

（略）汎化を行わなかったら「役職ごとに給与を求めて集計する」という手順になるでしょう。
汎化を行ったことで「社員の給与を一気に集計する」という手順が実現できるのです。（終）

まあ、それに必要かなと思ったんですが、単に shain = [] に shain.append(shainObject) で集めて、
for で 要素に次々と .getKyuyo して、それを加算していくだけでいいような気もします。

本当に __getattribute__ が使えないと困るのかな？ 頭冷やして考えないと分からないw
とにかく、いろいろ参考になりました。みなさん、どうもありがとう。

37 名前：デフォルトの名無しさん 投稿日：2007/02/18(日) 14:06:03
>>35
そのプログラムは単にオブジェクト指向のポリモーフィズム（多様性）を利用しているだけなので
__getattribute__ は必要ありません。そのプログラムを Python で書くなら、こんな感
じになります。

```
class Shain:
    def getKyuyo(self): return 0
class Bucho(Shain):
    def getKyuyo(self): return 700000
class Kacho(Shain):
    def getKyuyo(self): return 500000
class Tanto(Shain):
    def getKyuyo(self): return 300000

s = [ Bucho(), Kacho(), Kacho(), Tanto(), Tanto(), Tanto() ]

goukei = 0
for p in s: goukei += p.getKyuyo()
print goukei
```

ちなみに、Python では Bucho Kacho Tanto を Shain のサブクラスにしなくても動きます。
これは Java の「変数の型」という概念が Python にはないからです。
Python ではオブジェクトに対してメソッドを呼び出した場合、その名前を持つメソッドが
動的に（実行時に）決定されるので、スーパークラスで抽象化する必要はありません。（ダックタイピング）
（ただし、スーパークラスの振る舞いを引き継ぐ為の継承は有効な手段です）
呼び出されるメソッドが静的に（コンパイル時に）決定される Java や C++ などとの大きな違いですね。

39 名前：35 投稿日：2007/02/18(日) 14:26:56
>>37
サンクス。

■ a=1.2+1.0000000000000000ってやると ... (875-879)

875 名前：デフォルトの名無しさん 投稿日：2007/04/03(火) 01:34:43
a=1.2
a=1.2+1.0000000000000000
ってやると a が
2.2000000000000002
になって困ってます
どうやったら解決しますか？

876 名前：デフォルトの名無しさん 投稿日：2007/04/03(火) 01:39:54
```
>>> from decimal import Decimal
>>> Decimal('1.2') + Decimal('1.0000000000000000')
Decimal("2.2000000000000000")
```

878 名前：デフォルトの名無しさん 投稿日：2007/04/03(火) 02:17:09
ありがとうございます

879 名前：デフォルトの名無しさん 投稿日：2007/04/03(火) 08:02:41
ほんとにぱいそん信者は使えないねw

■チミ達は IDE というかエディターなどというものはなに使ってる (915-922)

915 名前：デフォルトの名無しさん 投稿日：2007/04/05(木) 18:35:18
チミ達は IDE というかエディターなどというものはなに使ってる
まさかメモ帳はありえないだろ普通wwww
おれは scite を使ってます

916 名前：デフォルトの名無しさん 投稿日：2007/04/05(木) 19:05:00
vim-7

917 名前：デフォルトの名無しさん 投稿日：2007/04/05(木) 19:10:29
xyzzy

918 名前：デフォルトの名無しさん 投稿日：2007/04/05(木) 19:21:39
IDLE

919 名前：デフォルトの名無しさん 投稿日：2007/04/05(木) 20:45:08
emacs

five-things-I-hate ってちゃんと使ってないと怖くて書けないよな‥

920 名前：デフォルトの名無しさん 投稿日：2007/04/05(木) 21:06:43
PyScripter + IDLE

921 名前：デフォルトの名無しさん 投稿日：2007/04/05(木) 21:18:10
秀丸

922 名前：デフォルトの名無しさん 投稿日：2007/04/05(木) 23:21:04
秀丸 たまに PyCrust 併用
最終確認が必要な場合は eclipse など

■ int を float にするほうほうがわかりません (931-933)

931 名前：デフォルトの名無しさん 投稿日：2007/04/06(金) 21:57:30
int を float にするほうほうがわかりません

932 名前：デフォルトの名無しさん 投稿日：2007/04/06(金) 22:05:59
>>>float(4)
4.0

933 名前：デフォルトの名無しさん 投稿日：2007/04/06(金) 22:09:14
ありがとうございます
とてもさんこうになります
カッコのなかは文字じゃないとだめとおもってました

part18

■ docutils をライブラリとして使う方法を探しています。(16-19)

16 名前：デフォルトの名無しさん 投稿日：2007/04/10(火) 17:35:11
docutils をライブラリとして使う方法を探しています。
s = """
* foo

```
* bar
* baz
"""
html = rst2html(s)
print html,
```

したら

```
<ul>
<li>foo</li>
<li>bar</li>
<li>baz</li>
</ul>
```

に変換されるようにしたいんです。
http://docutils.sourceforge.net/
をみてもいまいち分からないのですが、アドバイスお願いします。

17 名前：デフォルトの名無しさん 投稿日：2007/04/10(火) 19:17:57

```
import docutils.core
src = """\
* foo
* bar
* baz
"""
settings = {
    "stylesheet": "",
    "stylesheet_path": None,
}
print docutils.core.publish_string(src, writer_name='html',
                                   settings_overrides=settings)
```

18 名前：デフォルトの名無しさん 投稿日：2007/04/10(火) 19:27:10

>>17
ありがとうございます

19 名前：デフォルトの名無しさん 投稿日：2007/04/10(火) 19:37:16

1. docutils をライブラリとして使う、ねえ。まず rst2html.py が何してるか見てみますか。
2. docutils.core.publish_cmdline() とゆー関数を呼んでるなあ。どういう関数かな。(site-packages/docutils/core.py を見る)
3. publish_ なんちゃら という関数がいっぱいあるなあ。お、publish_string とゆー関数もあるなあ。
4. 使い方がよく分からないから docutils.core.publish_string でググってみよう。(用例が見つかる)
5. publish_string にはたくさん引数があるなあ。writer_name は rst2html.py で指定されてるのと同じ値を渡してみよう。
6. スタイルシートファイルが無いって怒られるなあ。デフォルトの値だとマズいんだろうなあ。
7. settings_overrides とゆー引数の値にテキトーな辞書を指定すればデフォルトを上書き指定できるんだろうなあ。
8. rst2html.py のコマンドラインオプションに --stylesheet-path とゆーのがあるから "stylesheet_path" を辞書のキーにしてみよう。(ビンゴ！)
9. とりあえずスタイルシートファイルは要らないなあ。値は None にしてみよう。

というような調子で試行錯誤した結果が >>17 のコード。

要約すると「ソース嫁」「ググれ」。

- スレッドダイジェスト版のようなものを作成テスト - 名無しさん (2007 年 04 月 12 日 07 時 27 分 25 秒)
- 1 ページが大きすぎると編集しづらいので、part N ごとに分けてページを作成して、あとから、include で 1 ページにまとめた方がいいかもしれない - 名無しさん (2007 年 04 月 12 日 07 時 43 分 16 秒)