Top of Form



Bottom of Form

Top of Form



**Python 3: The Python Environment  
Lesson 9, Project 1**

Handed in: 8 Jun 2015 06:39:14PM Graded: 11 Jun 2015 10:03:03AM

**Here are your instructions:**

Create a Pydev project named **Python3\_Homework09** and assign it to your Python3\_Homework working set. In **Python3\_Homework09/src**, create a program named **centipede.py**, including a class named "Centipede." This class has the following requirements:

1. Once instantiated if called with a value, it appends that argument to an internal list:

>>> from centipede import Centipede

>>> ralph = Centipede()

>>> ralph('pretzel')

>>> ralph.stomach

['pretzel']

2. If you print() the class, it returns a comma-delimited string of the internal list:

>>> ralph('pickles')

>>> print(ralph)

pretzel,pickles

3. Each time an attribute is set on the centipede object, it appends the name of the attribute to another internal list:

>>> ralph.shoes = 100

>>> ralph.hat = 1

>>> ralph.legs

['shoes', 'hat']

4. The representation of the centipede object must be a comma-delimited string of this second internal list.

>>> repr(ralph)

'shoes,hat'

5. The **legs** and **stomach** attributes should be protected against having their values reset from outside. They're "internal use only" when it comes to changing them, and an AttributeError should be raised if attempts are made to set them.

>>> ralph.legs = 3

Traceback (most recent call last):

File "<console>", line 1, in <module>

File "V:\workspace\Python3\_Homework09\src\centipede.py", line 15, in \_\_setattr\_\_

raise AttributeError("{0} is for internal use only".format(key))

AttributeError: legs is for internal use only

Copy and include this **test\_centipede.py** unittest file, which your centipede program should pass:

import unittest

from centipede import Centipede

class TestBug(unittest.TestCase):

def test\_stomach(self):

ralph = Centipede()

ralph('chocolate')

ralph('bbq')

ralph('cookies')

ralph('salad')

self.assertEqual(ralph.\_\_str\_\_(), 'chocolate,bbq,cookies,salad')

def test\_legs(self):

ralph = Centipede()

ralph.friends = ['Steve', 'Daniel', 'Guido']

ralph.favorite\_show = "Monty Python's Flying Circus"

ralph.age = '31'

self.assertEqual(ralph.age, '31', "ralph doesn't know how old he is")

self.assertEqual(ralph.\_\_repr\_\_(), 'friends,favorite\_show,age')

def test\_protected(self):

ralph = Centipede()

self.assertRaises(AttributeError, setattr, ralph, "legs", [])

self.assertRaises(AttributeError, setattr, ralph, "stomach", [])

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

**Your Comment:**

Kirby,

Wowsers! \_\_setattr\_\_ is/was a tough cookie to get the hang of! Thank goodness for print

statements (which I didn't start using until FAR TOO LATE into the debugging process, but better

late than never).

But while I was at it, I added some functionality so that you can add stomach items to our

centipede instances during instantiation (e.g. ralph = Centipede('pretzel', 'pickles')) and also added

a test to the test suite for that feature (which I named test\_EATstantiation...Ha!)

Definitely looking forward to your comments on how I setup the \_\_setattr\_\_ definition. I feel like

is probably a much simpler way than I did it, but it does work and based on your comments I'll go

back anyway and refactor.

**Items Handed In**

* [Open Project Handed In](https://students.oreillyschool.com/student/project/?/.handin/147-6931-1/com.ost.jwoloson.147.6931.1.Python3_Homework09.zip)

**Overall Comments:**

I see what you did to get around \_\_init\_\_ having trouble initializing its own self.legs and self.stomach:

you created a mixin so that \_\_init\_\_ might be outsourced. That's creative. Lets move on with some

comments.

The implementation below shows how, in \_\_init\_\_, you can avoid getting the attention of \_\_setattr\_

by using self.\_\_dict\_\_]'legs'] = [ ] instead, which is invoking \_\_getitem\_\_ (square brackets trigger

that one, while \_\_getattr\_\_ is triggered by dot notation e.g. obj.legs).

'''

centipede class

'''

class Centipede(object):

def \_\_init\_\_(self):

self.\_\_dict\_\_['stomach'] = []

self.\_\_dict\_\_['legs'] = []

def \_\_call\_\_(self, stuff):

self.stomach.append(stuff)

def \_\_str\_\_(self):

return ",".join(self.stomach)

def \_\_repr\_\_(self):

return ",".join(self.legs)

def \_\_setattr\_\_(self, name, value):

if (name == "legs") or (name == "stomach"):

raise AttributeError

if not hasattr(self, name):

self.\_\_dict\_\_['legs'].append(name)

self.\_\_dict\_\_[name] = value

Another approach is:

def \_\_init\_\_(self):

super().\_\_setattr\_\_('stomach', []) # parent version of \_\_setattr\_\_

super().\_\_setattr\_\_('legs', [])

This technique does not require a superclass mixin, merely an appeal

(call) to the "higher" (parent) version of \_\_setattr\_\_, which is not concerned

with Centipede anatomical parts (as in protection them).

Good work. You see how messing with \_\_setattr\_\_ has lots of ripple effects!

-Kirby

**Grade:**

Great

[Take Me Back](ostreturn:)

© 2014, O'Reilly Media, Inc. All rights reserved.

Bottom of Form