

20.12. Testing classes

To test a user-defined class, you will create test cases that check whether instances are created properly, and you will create test cases for each of the methods as functions, by invoking them on particular instances and seeing whether they produce the correct return values and side effects, especially side effects that change data stored in the instance variables. To illustrate, we will use the `Point` class that was used in the introduction to classes.

To test whether the class constructor (the `__init__`) method is working correctly, create an instance and then make tests to see whether its instance variables are set correctly. Note that this is a side effect test: the constructor method's job is to set instance variables, which is a side effect. Its return value doesn't matter.

A method like `distanceFromOrigin` in the `Point` class you saw does its work by computing a return value, so it needs to be tested with a return value test. A method like `move` in the `Turtle` class does its work by changing the contents of a mutable object (the point instance has its instance variable changed) so it needs to be tested with a side effect test.

Try adding some more tests in the code below, once you understand what's there.

Save & Run

Download

Load History

Show CodeLens

```

1 class Point:
2     """ Point class for representing and manipulating x,y coordinates. """
3
4     def __init__(self, initX, initY):
5
6         self.x = initX
7         self.y = initY
8
9     def distanceFromOrigin(self):
10         return ((self.x ** 2) + (self.y ** 2)) ** 0.5
11
12     def move(self, dx, dy):
13         self.x = self.x + dx
14         self.y = self.y + dy
15
16 import test
17
18 #testing instance variables x and y
19 p = Point(3, 4)
20 test.testEqual(p.y, 4)
21 test.testEqual(p.x, 3)
22
23 #testing the distance method
24 p = Point(3, 4)
25 test.testEqual(p.distanceFromOrigin(), 5.0)

```



ThinkingAboutClasses.html

(Tamagotchi.html) Next Section

```
26 |
27 #testing the move method
28 p = Point(3, 4)
29 p.move(-2, 3)
30 test.testEqual(p.x, 1)
31 test.testEqual(p.y, 7)
32
```

ActiveCode: 1 (ac19_3_1)

Check your understanding

test-3-1: For each function, you should create exactly one test case.

- ☐ A. True
- ☐ B. False

[Check Me](#)[Compare me](#)

test-3-2: To test a method that changes the value of an instance variable, which kind of test case should you write?

- ☐ A. return value test
- ☐ B. side effect test

[Check Me](#)[Compare me](#)[ThinkingAboutClasses.html](#)
(ThinkingAboutClasses.html)[Tamagotchi.html](#) Next Section

test-3-3: To test the function `maxabs`, which kind of test case should you write?

```
def maxabs(L):  
    """L should be a list of numbers (ints or floats). The return value should be  
    the maximum absolute value of the numbers in L."""  
    return max(L, key=abs)
```

- ☐ A. return value test
- ☐ B. side effect test

Check Me

Compare me

test-3-4: We have usually used the `sorted` function, which takes a list as input and returns a new list containing the same items, possibly in a different order. There is also a method called `sort` for lists (e.g. `[1, 6, 2, 4].sort()`). It changes the order of the items in the list itself, and it returns the value `None`. Which kind of test case would you use on the `sort` method?

- ☐ A. return value test
- ☐ B. side effect test

Check Me

Compare me

Mark as completed

