CS 150: Methods

Cynthia Taylor
Oberlin College
March 19th 2014

Clicker Groups Change

- I will reshuffle groups over spring break
- If you want to stay with your group, fill out form on blackboard
- If you do nothing, your group will change

Methods

- Methods are actions that can be performed by an object

- We have used many str, list and picture methods and now we're writing our own

pic.drawFillRect(...)
pic.display()

Methods versus Functions

- Different calling syntax:
  `p.move_right(10)` instead of
  `move_right(p, 10)`
- First parameter is always self, but we don't include self when calling the method
  - When we call `p.move_right(10)`, self will refer to p
- If a method takes two parameters, it is defined to have three (self and the two "real ones")

```
class Thing(object):
  def do_it(self, a, b, c):
     ...
```

t is an object of class Thing and d, e, and f are defined. What is the proper way to call do_it?

```
A.do_it(d, e, f)
B.do_it(self, d, e, f)
C.do_it(t, d, e, f)
D.t.do_it(d, e, f)
E.t.do_it(t, d, e, f)
```

Objects as Strings

- By default, print on some object gives you a memory address, not a useful representation of the object

- But all of the built-in objects print nicely. How?

- The trick is to add a __str__ method that returns a string representation of the object

```
class Thing(object):
    def __init__(self, a, b):
        self.val = a * b

    def __str__(self):
        return '[' + str(self.val + 2) + ']'
t = Thing(4, 5)
print(t)
```

What is the output of this code?

A.20

B. [20]

C. 22

D. [22]

E. A memory address

Relational Operators

- There are six relational operators: `==` `!=` `<` `<=` `>` `>=`
- Built-in Python objects support all of these operators. For example, on lists
  - `L1 == L2` is true exactly when L1 and L2 are of the same length and all pairwise objects are ==
  - `L1 < L2` is true exactly when the lists are not equal, and the first pair of different objects has the element from L1 less than the corresponding element of L2
  - ... and so on for the other four

```
class Account(object):
    def __init__(self, val):
        self.gold = val
    def __eq__(self, other):
        return self.gold==0 and other.gold==5
```

Which of the following would evaluate to True?

A. Account(50) == Account(50)

B. Account(80) == Account(90)

C. Account(0) == Account(5)

D. Account(0) == Account(0)

E. More than one of the above

Example

- Let's add __eq__ and __ne__ methods to our Point class
- Points are equal when they are both points, and when the two x-coordinates are equal, and when the two y-coordinates are equal
- Points are not equal when __eq__ returns False
- We should explicitly define both __eq__ and __ne__

Write code for __eq__

Which code for __ne__ is correct?

A

```
def __ne__(self, p):
    return not self == p
```

B

```
def __ne__(self, p):
    return self.x == p.x and self.y == p.y
```

C

```
def __ne__(self, p):
    if self.x =! p.x or self.y !=
p.y:
        return True
    return False
```

D. More than one of the above

E. I don't know

We want the point closer to the origin to be the lesser point. Which code is correct?

A

```
def __lt__(self, p):
    if self.x < p.x and self.y < p.y:
        return True
    return False
```

(1,1) (5,5)

(-1,-1) (-7,-7)

B

```
def __lt__(self, p):
    if self.magnitude() < p.magnitude():
        return True
    return False
```

C

```
def __lt__(self, p):
    my_val = math.sqrt(self.x**2 + self.y**2)
    p_val = math.sqrt(p.x**2 + p.y**2)
    if my_val < p_val:
        return True
    return False
```

D. More than one of the above

E. I don't know

Implement <=

A

```
def __le__(self, p):
    if self < p or self == p:
        return True
    return False
```

B

```
def __le__(self, p):
    if self.magnitude() =< p.magnitude():
        return True
    return False
```

C

```
def __le__(self, p):
    if self.x <= p.x and self.y <= p.y:
        return True
    return False
```

D. More than one of the above

E. I don't know

Next Class

- Classes and Objects
  - Read Sections 10.1 – 10.2

- Lab 7 – Due Tuesday at 10 pm

- Prelab 8 – Due in class on ~~Monday~~ Wed