ES 112
Introduction to Objected Oriented Programming
Nov 11, 2015

Computing

IIT Gandhinagar, India

Nov, 2015

Recap

- An `object` is a collection of data and methods grouped together.

```
#Point class
class Point():
    """ To represent a point in 2D"""

blank=Point()
# Point Class with initilizer
class Point():
    def __init__(self,x,y):
        self.x=x
        self.y=y

a=Point(1,2)
print a.x, a.y
#Gives 1 2
```

Recap

- An `object` is a collection of data and methods grouped together.
- A `class` is a collection of objects of similar type, user defined data type.

```
1  #Point class
2  class Point():
3      """ To represent a point in 2D"""
4
5  blank=Point()
6  # Point Class with initilizer
7  class Point():
8      def __init__(self,x,y):
9          self.x=x
10         self.y=y
11
12 a=Point(1,2)
13 print a.x, a.y
14 #Gives 1 2
```

Box Class

```
1  # Point Class with initilizer
2  class Point():
3      def __init__(self,x,y):
4          self.x=x
5          self.y=y
6
7  class Box():
8      def __init__(self,x,y,h,w):
9          self.corner=Point(x,y)
10         self.h=h
11         self.w=w
12
13 box=Box(1,2,10,20)
14
15 print box.corner.x,box.h
16 # Gives ? ?
```

Attributes

- Attributes can be vaiables and methods (functions).
- Class attributes and instance attributes.

```
1  #---
2  class A():
3      foo = []
4
5  >>> a, b = A(), A()
6  >>> a.foo.append(5)
7  >>> b.foo
8  [5]
9  #-------
10 class A():
11     def __init__(self):
12         self.foo = []
13
14 >>> a, b = A(), A()
15 >>> a.foo.append(5)
16 >>> b.foo
17 []
```

Method

- We will create an object called circle.

Method

- We will create an object called circle.
- Identify the data attributes: origin of the circle (x,y), and radius.

Method

- We will create an object called circle.
- Identify the data attributes: origin of the circle (x,y), and radius.
- Identify the functions associated with a circle: Find area, find circumference.

Method

- We will create an object called circle.
- Identify the data attributes: origin of the circle (x,y), and radius.
- Identify the functions associated with a circle: Find area, find circumference.

```
1  class Circle():
2      def __init__(self,x1,y1,r1):
3          self.x=x1
4          self.y=y1
5          self.r=r1
6      #a method to compute area
7      def area(self):
8          return 3.1415*self.r*self.r
9      #a method to compute circumference
10     def circm(self):
11         return 2*3.1415.self.r
12
13 #instantiation
14 c=Circle(1,1,5)
15
16 print c.r
17 print c.area() # Argument? c is the argument
18 print Circle.area(c)
```

Methods are Class attributes

- The frist argument to a method is the object itself.
- The method belongs to the class and when we write a.area() it gets converted to Circle.area(a).

```
1  class Circle():
2      def __init__(self,x1,y1,r1):
3          self.x=x1
4          self.y=y1
5          self.r=r1
6      #a method to compute area
7      def area(self):
8          return 3.1415*self.r*self.r
9      #a method to compute circumference
10     def circm(self):
11         return 2*3.1415.self.r
12
13 #instantiation
14 c=Circle(1,1,5)
15
16 print c.r
17 print c.area() # Argument? c is the argument
18 print Circle.area(c)
```

Copying and Comparing Objects

- Nested list: Guess the output.

```
>>> a=5
>>> b=a
>>> b=2
>>> print a,b
5 2
>>> lst=[1,2,3]
>>> lst1=lst
>>> lst1[0]=100
>>> print lst, lst1
#What is the output?
```

Copying and Comparing Objects

- Similar things happen with objects.

```
1  class Point():
2      def __init__(self,x,y):
3          self.x=x
4          self.y=y
5
6  class Box():
7      def __init__(self,x,y,h,w):
8          self.corner=Point(x,y)
9          self.h=h
10         self.w=w
11
12 box1=Box(1,2,10,20)
13 box2=box1
14
15 box1.h=2
16 print box2.h
17 # Gives 2
```

Copying and Comparing Objects

- Other side of the issue.

```
1 class Point():
2     def __init__(self,x,y):
3         self.x=x
4         self.y=y
5
6 a=Point(1,2)
7 b=Point(1,2)
8 print a==b
9 #Gives False
```

Shallow Copying

```
>>> import copy
>>> box2 = copy.copy(box)
>>>box2 is box
False
>>>box2.corner is box.corner
True
```

Does not separate the new insatance completely.

Deep Copying

In order to make the objects completely distinct in memory, need to deep-copy.

```
>>> import copy
>>> box2 = copy.deepcopy(box)
>>>box2 is box
False
>>>box2.corner is box.corner
False
```

Linked List