NPTEL MOOC

PROGRAMMING, DATA STRUCTURES AND ALGORITHMS IN PYTHON

Week 7, Lecture 2

Madhavan Mukund, Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Classes and objects

- Class
  - Template for a data type
    - How data is stored
    - How public functions manipulate data
- Object
  - Concrete instance of template

Classes and objects

```
class Heap:
  def __init__(self,l):
    # Create heap
    # from list l

  def insert(self,x):
    # insert x into heap

  def delete_max(self):
    # return max element
```

```
# Create object,
# calls __init__()
l = [14,32,15]
h = Heap(l)

# Apply operation
h.insert(17)

h.insert(28)

v = h.delete_max()
```

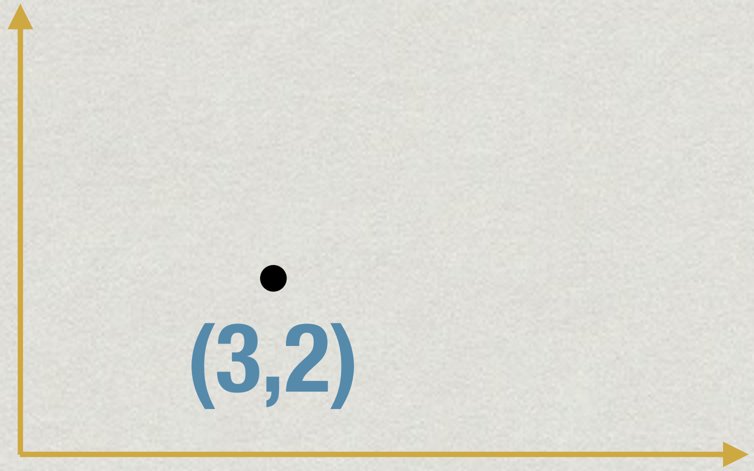
Points on a plane

```
class Point:
  def __init__(self,a,b):
    self.x = a
    self.y = b

  def translate(self,deltax,deltay):
    # shift (x,y) to (x+deltax,y+deltay)
    self.x += deltax # same as selfx =
                     #        self.x + deltax
    self.y += deltay
```

Points on a plane

```
p = Point(3,2)
```

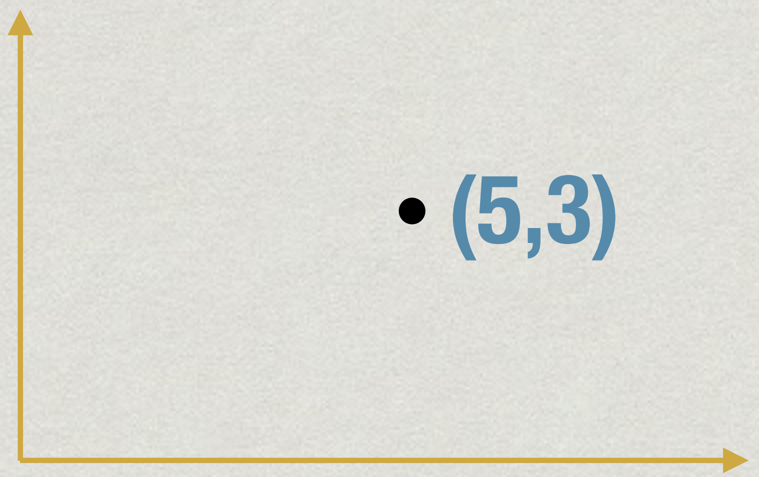


```
class Point:
  def __init__(self,a,b):
    self.x = a
    self.y = b

  def translate(self,deltax,deltay):
    # shift (x,y) to (x+deltax,y+deltay)
    self.x += deltax # same as selfx =
                     #        self.x + deltax
    self.y += deltay
```

Points on a plane

```
p = Point(3,2)
p.translate(2,1)
```



```
class Point:
  def __init__(self,a,b):
    self.x = a
    self.y = b

  def translate(self,deltax,deltay):
    # shift (x,y) to (x+deltax,y+deltay)
    self.x += deltax # same as selfx =
                     #        self.x + deltax
    self.y += deltay
```

Points on a plane

```
class Point:
  . . .
  def odistance(self):
    # Distance from (0,0)
    # from math import *
    return(
      sqrt(
        (self.x*self.x) + (self.y*self.y)
      ))
```

Polar coordinates

- Recall polar coordinates
- Instead of (x,y), use $(r,\boldsymbol{\theta})$
  - $x = r \cos \boldsymbol{\theta}$
  - $y = r \sin \boldsymbol{\theta}$
  - $r = \sqrt{}(x^2 + y^2)$ — same as distance
  - $\boldsymbol{\theta} = \tan^{-1}(y/x)$

Points on a plane

```
class Point:
  def __init__(self,a,b):
    self.r = sqrt(a*a + b*b)
    if a == 0:
      self.theta = 0
    else:
      self.theta = atan(b/a)

  def odistance(self):
    return(self.r)

  def translate(self,deltax,deltay):
    # Convert (r,theta) to (x,y) and back!
```

- Private implementation has changed
- Functionality of public interface remains same

Default arguments

```
class Point:
  def __init(self,a=0,b=0):
    self.x = a
    self.y = b

  . . .
```

```
# Point at (3,4)
p1 = Point(3,4)

# Point at (0,0)
p2 = Point()
```

Special functions

- `__init__()`
  - Constructor, called when object is created
- `__str__()`
  - Return string representation of object
  - `str(o) == o.__str__()`
  - Implicitly invoked by `print()`

```
def __str__(self):  # For Point()
  return('('+str(self.x)+','+str(self.y)+')')
```

Special functions

- `__add__()`
  - Invoked implicitly by +
  - `p1 + p2 == p1.__add__(p2)`

```
def __add__(self,p):  # For Point()
  return(Point(self.x+p.x,self.y+p.y)

p1 = Point(1,2)
p2 = Point(2,5)
p3 = p1 + p2  # p3 is now (3,7)
```

Special functions

- `__mult__()`
  - Called implicitly by `*`
- `__lt__(), __gt__(), __le__(), . . .`
  - Called implicitly by `<`, `>`, `<=`
- Many others, see Python documentation