Chap15_PointRectangle_90

May 8, 2020

1 Clase si Obiecte. Capitolul 15

1.1 Ex. 5.1. Distanta intre doua puncte

```
In [6]: import math
        class Point(object):
            """Represents a point in 2D space."""

        def distanta(p1, p2):
             return math.sqrt((p2.x-p1.x)**2+(p2.y-p1.y)**2)

        one = Point()
        two = Point()
        one.x = input("abscisa primul punct = ")
        one.y = input("ordonata primului punct = ")
        two.x = input("abscisa punctului al doilea = ")
        two.y = input("ordonata punctului al doilea = ")

        print "Distanta este %.3f" % distanta(one,two)
```

```
abscisa primul punct = 10
ordonata primului punct = 20
abscisa punctului al doilea = 30
ordonata punctului al doilea = 40
Distanta este 28.284
```

1.2 Ex.15.2. Mutare dreptunghi

```
In [10]: class Point(object):
             """ Punctul """

         class Rectangle:
             """ defineste dreptunghiul prin colt, lungime si latime"""

         def muta_dreptunghi(bbox, ddx, ddy):
             """Mutare dreptunghi """
```

```
        bbox.corner.x += ddx
        bbox.corner.y += ddy

    box = Rectangle()
    box.corner = Point()
    box.corner.x = int(input("abscisa coltului: "))
    box.corner.y = int(input("ordonata coltului: "))
    box.width = int (input("lungimea (Ox): "))
    box.height = int (input("latimea (Oy): "))

    dx = int(input("muta pe Ox: "))
    dy = int(input("muta pe Oy: "))

    print "Initial: ", box.corner.x, box.corner.y, box.width, box. height
    muta_dreptunghi(box, dx, dy)
    print "Final: ", box.corner.x, box.corner.y, box.width, box. height
```

```
abscisa coltului: 10
ordonata coltului: 20
lungimea (Ox): 100
latimea (Oy): 200
muta pe Ox: 5
muta pe Oy: 3
Initial:  10 20 100 200
Final:  15 23 100 200
```

1.3 Ex.15.3. Muta dreptunghi in dreptunghi NOU

1.3.1 (!) functie care returneaza dreptunghiul nou

In [12]:
```
import copy

class Point(object):
    """ Punctul """

class Rectangle:
    """ defineste dreptunghiul prin colt, lungime si latime"""

def muta_dreptunghi(bbox, ddx, ddy):
    """Mutare dreptunghi """
    new_bbox = copy.copy(bbox)
    bbox.corner.x += ddx
    bbox.corner.y += ddy
    return new_bbox

box = Rectangle()
box.corner = Point()
box.corner.x = int(input("abscisa coltului: "))
```

```
        box.corner.y = int(input("ordonata coltului: "))
        box.width = int (input("lungimea (Ox): "))
        box.height = int (input("latimea (Oy): "))

        dx = int(input("muta pe Ox: "))
        dy = int(input("muta pe Oy: "))

        print "Initial: ", box.corner.x, box.corner.y, box.width, box. height
        new_box = muta_dreptunghi(box, dx, dy)
        print "Final: ", new_box.corner.x, new_box.corner.y, new_box.width, new_box. height
```

```
abscisa coltului: 10
ordonata coltului: 20
lungimea (Ox): 30
latimea (Oy): 40
muta pe Ox: 5
muta pe Oy: 3
 Initial:  10 20 30 40
Final:  15 23 30 40
```

1.4 Alte probleme din capitolul 15

1.4.1 15.4. –> Determina centrul de greutate al unui dreptunghi

```
In [28]: class Point(object):
             """Represents a point in 2D space."""

         def print_Point(point):
             print point.x, ", ", point.y

         class Rectangle(object):
             """Represents a rectangle.
             atributes:width,height,corner"""

         def find_center(r):
             p=Point()
             p.x=r.corner.x +r.width/2.0
             p.y=r.corner.y +r.height/2.0
             return p

         box=Rectangle()
         box.width=100.0
         box.height=200.0
         box.corner=Point()
         box.corner.x=0.0
         box.corner.y=0.0

         print "Dreptunghi: ", (box.corner.x, box.corner.y), box.width, box.height
```

```
        print "Centrul de greutate al dreptunghiului: ",
        print_Point(find_center(box))
        print "-------------------------"

        #Ce returneaza urmatoarea secventa de cod? Justificati raspunsul.
        print "Dreptunghi: ", (box.corner.x, box.corner.y), box.width, box.height
        print "Centrul de greutate al dreptunghiului: ", print_Point(find_center(box))
```

```
Dreptunghi:  (0.0, 0.0) 100.0 200.0
Centrul de greutate al dreptunghiului:  50.0 ,  100.0
-------------------------
Dreptunghi:  (0.0, 0.0) 100.0 200.0
Centrul de greutate al dreptunghiului:  50.0 ,  100.0
None
```

1.4.2 15.5 –> Scaleaza dreptunghi - obiectul este modificabil - prin atributele lui

```
In [29]: class Point(object):
             """Represents a point in 2D space."""

         def print_Point(point):
             print point.x, ", ", point.y

         class Rectangle(object):
             """Represents a rectangle.
             atributes:width,height,corner"""

         def grow_rectangle(dr,dwidth,dheight):
             dr.width = dr.width + dwidth
             dr.height = dr.height + dheight

         box=Rectangle()
         box.width=100.0
         box.height=200.0
         box.corner=Point()
         box.corner.x=0.0
         box.corner.y=0.0

         print "Dreptunghi dat:(", box.corner.x, ",", box.corner.y, ")", box.width, box.height
         grow_rectangle(box,50,100)
         print "Dreptunghi scalat:(", box.corner.x, ",", box.corner.y, ")", box.width, box.heigh
```

```
Dreptunghi dat:( 0.0 , 0.0 ) 100.0 200.0
Dreptunghi scalat:( 0.0 , 0.0 ) 150.0 300.0
```

1.4.3 (Q) Este OBLIGATORIU comentariul multi-linie dupa definirea clasei?

1.4.4 (A) Este nevoie de cel putin o linie de cod in corpul clasei; in lipsa de linii proprii, se adauga o linie "moarta", adica un comentariu

```
In [7]: # fara comentariu in corpul clasei
        class Point(object):
            def print_point(p):
                print(p.x,',',p.y)

        punct = Point()
        punct.x=10
        punct.y=20

        punct.print_point()
        print '1abracadabra'

        # ----------------------
        # doar comentariu in corpul clasei
        class Point(object):
            """def print_point(p):
                print(p.x,',',p.y)"""

        punct = Point()
        punct.x=10
        punct.y=20

        #punct.print_point()
        print '2abracadabra'

        # ----------------------
        # corpul clasei fara linii de cod --> eroare "de indentare" pentru linia urmatoare

        #class Point(object):

        punct = Point()
        punct.x=10
        punct.y=20

        #punct.print_point()
        print '3abracadabra'
```

```
(10, ',', 20)
1abracadabra
2abracadabra
3abracadabra
```

1.4.5 La COPIEREA un obiect cu copy.copy(obiect), obiectele incapsulate vor avea aceeasi referinta; se copiaza / se creeaza obiecte diferite, dar ambele refera ACELASI obiect INCAPSULAT

In [5]:
```
import copy

class Rectangle(object):
    """Represents a rectangle.

    attributes: width, height, corner.
    """

class Point():
    """ Represents a point."""

box = Rectangle()
box.width = 100.0
box.height = 200.0
box.corner = Point() # corner este incapsulat in box
box.corner.x = 0.0
box.corner.y = 0.0

boxx = copy.copy(box)

boxx.corner.x = 10 # modific valoarea atributului lui boxx.corner
print box.corner.x # se  modifica si valoarea atributului lui box.corner
```

```
10
```