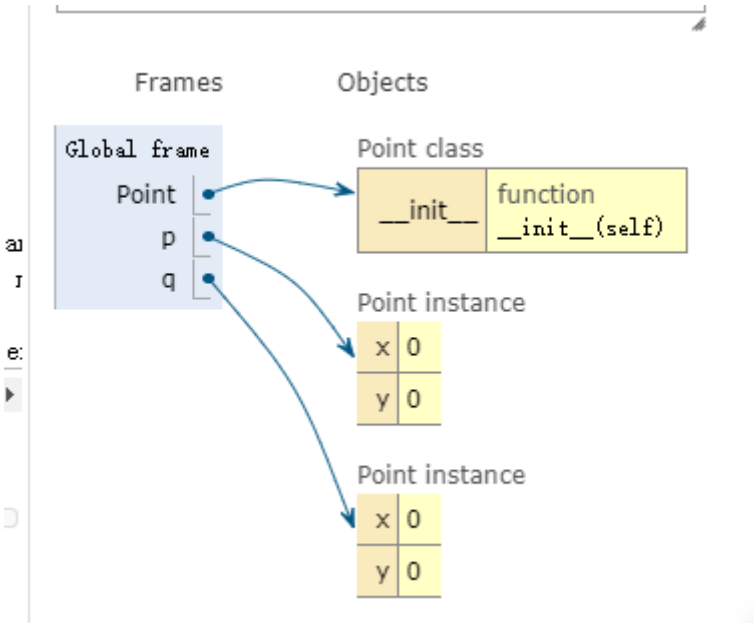
4.1 Classes and Objects

- python: object-oriented programming (OOP)对象
  - 无论是list integer dic都是classes 是本身存在于python里面的 但是有时候想用一些没有被defined的class 就得自己创建了
- user-defined classes
  - x轴和y轴 (10,2) (8,-3)
  - 跟点相关的典型操作：要求x y坐标：getX getY 希望用一种类型的函数防止对点的修改又可以直接查看到这些值 或者其他比如距离 矩形的题目
  - 首先希望有自己x和y的attribute
    - class Point: """ Point class for representing and manipulating x,y coordinates. """
    - def __init__(self): """ Create a new point at the origin """ 构造函数
    - self.x = 0
    - self.y = 0
    - 实际已经创建了两个点了 只不过都是00 所以觉得什么都没发生



    - p和q是两个不同的点

```
class Point:
    """ Point class for representing and manipulating x,y coordinates. """

    def __init__(self):

        self.x = 0
        self.y = 0

p = Point()          # Instantiate an object of type Point
q = Point()          # and make a second point

print(p)
print(q)

print(p is q)
```

```
<__main__.Point object>
<__main__.Point object>
False
```

- 像Point这样创建新的object的function叫做constructor
- 将类看作制造对象的工厂可能会有所帮助。类本身不是点的实例，但它包含创建点实例的机制。每次调用构造函数时，您都是在请求工厂为您创建一个新对象。当对象离开生产线时，将执行其初始化方法，以正确设置对象的工厂默认设置。
- “创建一个新对象”和“将其设置初始化为工厂默认设置”的组合过程称为实例化。instantiation

- 一般class define是放在import之后 也就是整个程序的最前面
- 现在只是创建了(0,0)如果想创建其他位置的点

```
class Point:
    """ Point class for representing and manipulating x,y coordinates. """

    def __init__(self, initX, initY):

        self.x = initX
        self.y = initY

p = Point(7,6)
```

- 练习

Check Your Understanding

1. Create a class called `NumberSet` that accepts 2 integers as input, and defines two instance variables: `num1` and `num2`, which hold each of the input integers. Then, create an instance of `NumberSet` where its num1 is 6 and its num2 is 10. Save this instance to a variable `t`.

Save & Run　2020/3/23 下午10:55:52 - 3 of 3　Show in CodeLens

```
class NumberSet:
    def __init__(self,num1,num2):
        self.num1=num1
        self.num2=num2

t=NumberSet(6,10)
```

- method belongs to a

- 用point而不是simple tuple的好处（7，6）
  - 可以对point用function和operation但是对tuple不行
  - 又加了一个method

```
class Point:
    """ Point class for representing and manipulating x,y coordinates. """

    def __init__(self, initX, initY):

        self.x = initX
        self.y = initY

    def getX(self):
        return self.x

    def getY(self):
        return self.y


p = Point(7,6)
print(p.getX())
print(p.getY())
```

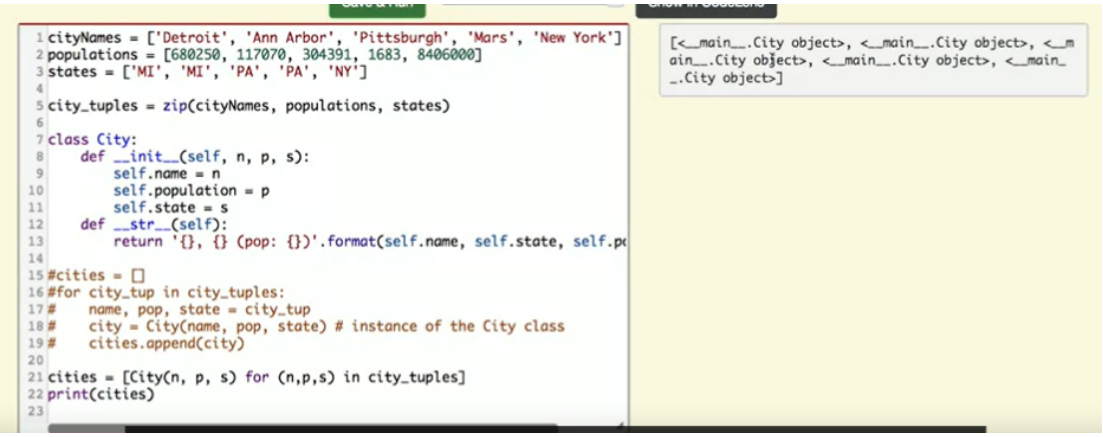
```
7
6
```

  - def distanceFromOrigin(self):
  - return ((self.x ** 2) + (self.y ** 2)) ** 0.5
  - 不会的题

Check Your Understanding

1. Create a class called `Animal` that accepts two numbers as inputs and assigns them respectively to two instance variables: `arms` and `legs`. Create an instance method called `limbs` that, when called, returns the total number of limbs the animal has. To the variable name `spider`, assign an instance of `Animal` that has 4 arms and 4 legs. Call the limbs method on the `spider` instance and save the result to the variable name `spidlimbs`.

    - 要先class Animal define__int__(self, arms, legs) self.arms=arms self.legs=legs再去动作 define里除了self一定要 其他两个可以是任意名字 但是在下面self.xxx的时候要对应好 xxx是variable的名字
    - 案例
      -



- objects as arguments and parameters
  - 算距离 需要import math然后用point method（构造一个坐标系）输入3个量 但是其中两个量是第一个量的组成成分 distancefromorigin就是离远点的距离 根号x方+y方

```
import math
class Point:
    """ Point class for representing and manipulating x,y coordinates. """
    def __init__(self, initX, initY):

        self.x = initX
        self.y = initY
    def getX(self):
        return self.x
    def getY(self):
        return self.y
    def distanceFromOrigin(self):
        return ((self.x ** 2) + (self.y ** 2)) ** 0.5

def distance(point1, point2):
    xdiff = point2.getX()-point1.getX()
    ydiff = point2.getY()-point1.getY()

    dist = math.sqrt(xdiff**2 + ydiff**2)
    return dist

p = Point(4,3)
q = Point(0,0)
```

  - 有一个__str__(self):是输出话的 基本上都是xxx.format的格式

```
class Point:
    """ Point class for representing and manipulating x,y coordinates. """

    def __init__(self, initX, initY):

        self.x = initX
        self.y = initY

    def getX(self):
        return self.x

    def getY(self):
        return self.y

    def distanceFromOrigin(self):
        return ((self.x ** 2) + (self.y ** 2)) ** 0.5

    def __str__(self):
        return "x = {}, y = {}".format(self.x, self.y)

p = Point(7,6)
print(p)
```

```
x = 7, y = 6
```

Save & Run Original - 1 of 1 Show in CodeLens

```
        return ((self.x ** 2) + (self.y ** 2)) ** 0.5

    def __str__(self):
        return "x = {}, y = {}".format(self.x, self.y)

    def halfway(self, target):
        mx = (self.x + target.x)/2
        my = (self.y + target.y)/2
        return Point(mx, my)

p = Point(3,4)
q = Point(5,12)
mid = p.halfway(q)
# note that you would have exactly the same result if you instead wrote
# mid = q.halfway(p)
# because they are both Point objects, and the middle is the same no matter what

print(mid)
print(mid.getX())
print(mid.getY())
```

```
x = 4.0, y = 8.0
4.0
8.0
```

- 如果要中点的话

Save & Run Original - 1 of 1 Show in CodeLens

```
        return ((self.x ** 2) + (self.y ** 2)) ** 0.5

    def __str__(self):
        return "x = {}, y = {}".format(self.x, self.y)

    def halfway(self, target):
        mx = (self.x + target.x)/2
        my = (self.y + target.y)/2
        return Point(mx, my)

p = Point(3,4)
q = Point(5,12)
mid = p.halfway(q)
# note that you would have exactly the same result if you instead wrote
# mid = q.halfway(p)
# because they are both Point objects, and the middle is the same no matter what

print(mid)
print(mid.getX())
print(mid.getY())

```

```
x = 4.0, y = 8.0
4.0
8.0
```

- sorting lists of instances
  - 有一种是 for f in sorted(L,key=lambda x:x.price): for f in sorted(L,key=Fruit.sort_priority)

Save & Run Show in CodeLens

```
class Fruit():
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def sort_priority(self):
        return self.price


L = [
    Fruit('Cherry', 10),
    Fruit('Apple', 5),
    Fruit('Blueberry', 20)
]

for f in sorted(L, key=lambda x: x.sort_priority()):
    print(f.name)

```

```
Apple
Cherry
Blueberry
```

```
class Fruit():
    def __init__(self, name, price):
        self.name = name
        self.price = price

L = [Fruit("Cherry", 10), Fruit("Apple", 5), Fruit("Blueberry", 20)]
for f in sorted(L, key=lambda x: x.price):
    print(f.name)

```

```
Apple
Cherry
Blueberry
```

Save & Run Original - 1 of 1 Show in CodeLens

```
class Fruit():
    def __init__(self, name, price):
        self.name = name
        self.price = price

    def sort_priority(self):
        return self.price

L = [Fruit("Cherry", 10), Fruit("Apple", 5), Fruit("Blueberry", 20)]
print("-----sorted by price, referencing a class method-----")
for f in sorted(L, key=Fruit.sort_priority):
    print(f.name)

print("---- one more way to do the same thing-----")
for f in sorted(L, key=lambda x: x.sort_priority()):
    print(f.name)

```

```
-----sorted by price, referencing a class method-----
Apple
Cherry
Blueberry
---- one more way to do the same thing-----
Apple
Cherry
Blueberry
```