hash

方法	意义
hash	内建函数 hash() 调用的返回值,返回一个整数。如果定义这个方法该类的实例就可hash。

```
print(hash(1))
print(hash('tom'))
print(hash(('tom',)))
```

```
1 class A:
2
       def __init__(self, name):
3
            self.name = name
4
 5
       def __hash__(self):
6
            return 1
 7
8
       def __repr__(self):
9
           return self.name
10
11
12
   a1 = A('tom')
13 a2 = A('tom')
14 print(a1, a2, hash(a1), hash(a2))
15 print([a1, a2])
16 | print((a1, a2))
    print({a1, a2}) # 去重了吗?
17
18 | print({a1, a1}) # 去重了吗? 为什么
19
20 # 元组
21 t1 = ('tom',)
22 t2 = ('tom',)
23 | print(t1 is t2)
24 | print(t1 == t2)
25 | print({t1, t2})
```

上例中,A的实例放在set中,它们hash值是相同的,为什么不能去重? hash值相同就会去重吗?

```
1 class A:
 2
       def __init__(self, name):
 3
            self.name = name
 4
 5
        def __hash__(self):
 6
            return 1
 7
        def __repr__(self):
 8
9
            return self.name
10
11
        def __eq__(self, other):
            return self.name == other.name
12
```

```
13

14 a1 = A('tom')

15 a2 = A('tom')

16 print({a1, a2}) # 去重了吗?

17 print({A('jerry'), A('jerry')}) # 去重了吗?
```

方法	意义
eq	对应==操作符,判断2个对象内容是否相等,返回bool值 定义了这个方法,如果不提供hash 方法,那么实例将不可hash了

__hash___方法只是返回一个hash值作为set的key,但是 去重 ,还需要 __eq___来判断2个对象是否相 等。

hash值相等,只是hash冲突,不能说明两个对象是相等的。

因此,一般来说实现 __hash__ 方法,要同时实现 __eq__ 方法。 去重 依赖 __eq__ 方法。

不可hash对象isinstance(p1, collections.Hashable)一定为False。

思考:

- 1. list类实例为什么不可hash?
- 2. functools.lru_cache使用到的functools._HashedSeq类继承自list,为什么可hash?

练习

设计二维坐标类Point,使其成为可hash类型,并比较2个坐标的实例是否相等?

- 1. list类实例为什么不可hash
 - 源码中有一句 __hash__ = None, 也就是如果调用 __hash__ ()相当于None(), 一定报错。 所有类都继承object, 而这个类是具有 __hash__ ()方法的, 如果一个类不能被hash, 就把 __hash__ 设置为None。
- 2. _HashedSeq类提供了__hash__方法,这个方法实际上计算的是元组的hash值

练习参考

```
class Point:
 1
 2
        def __init__(self, x, y):
            self.x = x
 3
 4
            self.y = y
 5
        def __hash__(self):
 6
 7
            return hash((self.x, self.y))
 8
 9
        def __eq__(self, other):
            return self.x == other.x and self.y == other.y
10
11
        def __repr__(self):
12
            return "<Point {}:{}>".format(self.x, self.y)
13
14
    p1 = Point(4, 5)
15
| 16 | p2 = Point(4, 5)
17
    print(hash(p1), hash(p2))
    print(p1 is p2)
18
    print(p1 == p2)
19
```