习题参考

1、随机整数生成类

可以指定一批生成的个数，可以指定数值的范围

常规实现如下

```
import random


# 1 普通类实现
class RandomGen:
    def __init__(self, start=1, stop=100, count=10):
        self.start = start
        self.stop = stop
        self.count = count

    def generate(self, start=1, stop=100, count=10):
        return [random.randint(self.start, self.stop) for x in range(self.count)]


# 2 作为工具类来实现，提供类方法
class RandomGen:
    @classmethod
    def generate(cls, start=1, stop=100, count=10):
        return [random.randint(start, stop) for x in range(count)]
```

随机整数生成类，可以指定一批生成的个数，可以指定数值的范围，可以调整每批生成数字的个数。
使用生成器实现，如下：

```
# 使用生成器实现1

import random

class RandomGenerator:
    def __init__(self, start=1, stop=100, patch=10):
        self.start = start
```

```
        self.stop = stop
        self.patch = patch
        self._gen = self._generate()

    def _generate(self):
        while True:
            yield random.randint(self.start, self.stop)

    def generate(self, count=0):
        if count <= 0:
            return [next(self._gen) for _ in range(self.patch)]
        else:
            return [next(self._gen) for _ in range(count)]

a = RandomGenerator()
print(a.generate())
print(a.generate(5))
```

```
# 生成器另一种实现
import random

class RandomGenerator:
    def __init__(self, start=1, stop=100, patch=10):
        self.start = start
        self.stop = stop
        self.patch = patch
        self._gen = self._generate()

    def _generate(self):
        while True:
            yield [random.randint(self.start, self.stop) for _ in range(self.patch)
]

    def generate(self, count=0):
        if count > 0:
            self.patch = count
        return next(self._gen)

a = RandomGenerator()
print(a.generate())
```

```
print(a.generate(5))
```

```
# 使用property
import random

class RandomGenerator:
    def __init__(self, start=1, stop=100, patch=10):
        self.start = start
        self.stop = stop
        self._patch = patch
        self._gen = self._generate()

    def _generate(self):
        while True:
            yield [random.randint(self.start, self.stop) for _ in range(self.patch)
]

    def generate(self):
        return next(self._gen)

    @property
    def patch(self):
        return self._patch

    @patch.setter
    def patch(self, value):
        self._patch = value

a = RandomGenerator()
print(a.generate())
a.patch = 5
print(a.generate())
```


2、打印坐标

使用上题中的类，随机生成20个数字，两两配对形成二维坐标系的坐标，把这些坐标组织起来，并打印输出

```
class Point:
```

```
    def __init__(self, x, y):
        self.x = x
        self.y = y

points = [Point(x,y) for x,y in zip(RandomGenerator(10).generate(),RandomGenerator(
10).generate())]

for p in points:
    print('{}:{}'.format(p.x, p.y))
```

3、车辆信息

记录车的品牌mark、颜色color、价格price、速度speed等特征，并实现增加车辆信息、显示全部车辆信息的功能

```
class Car: # 记录单一车辆
    def __init__(self, mark, speed, color, price):
        self.mark = mark
        self.speed = speed
        self.color = color
        self.price = price

class CarInfo:
    def __init__(self):
        self.info = []

    def addcar(self, car: Car):
        self.info.append(car)

    def getall(self):
        return self.info

ci = CarInfo()
car = Car('audi', 400, 'red', 100)
ci.addcar(car)

ci.getall() # 返回所有数据，此时在实现格式打印
```

4、实现温度的处理

实现华氏温度和摄氏温度的转换。

℃ = 5 × (℉ - 32) / 9

℉ = 9 × ℃ / 5 + 32

完成以上转换后，增加与开氏温度的转换，K = ℃ + 273.15

思路

假定一般情况下，使用摄氏度为单位，传入温度值。

如果不给定摄氏度，一定会把温度值转换到摄氏度。

温度转换方法可以使用实例的方法，也可以使用类方法，使用类方法的原因是，为了不创建对象，就可以直接进行温度转换计算，这个类设计像个温度工具类。

```
class Temperature:
    def __init__(self, t, unit='c'):
        self._c = None
        self._f = None
        self._k = None

        if unit == 'k':
            pass
        elif unit == 'f':
            pass
        else:
            self._c = t

    @property
    def c(self): # 摄氏度
        return self._c

    @property
    def k(self): # 开氏温度
        pass

    @property
    def f(self): # 华氏温度
        pass

    # 温度转换
```

```
    @classmethod
    def c2f(cls, c):
        return 9*c/5 + 32

    @classmethod
    def f2c(cls, f):
        return 5*(f-32)/9


    @classmethod
    def c2k(cls, c):
        return c + 273.15

    @classmethod
    def k2c(cls, k):
        return k - 273.15

    @classmethod
    def f2k(cls, f):
        return cls.c2k(cls.f2c(f))

    @classmethod
    def k2f(cls, k):
        return cls.c2f(cls.k2c(k))
```

进一步完善未完成代码，如下

```
class Temperature:
    def __init__(self, t, unit='c'):
        self._c = None
        self._f = None
        self._k = None

        # 都要先转换到摄氏度，以后访问再计算其它单位的温度值
        if unit == 'k':
            self._k = t
            self._c = self.k2c(t)
        elif unit == 'f':
            self._f = t
            self._c = self.f2c(t)
```

```
        else:
            self._c = t

    @property
    def c(self): # 摄氏度
        return self._c

    @property
    def k(self): # 开氏温度
        if self._k is None:
            self._k = self.c2k(self._c)
        return self._k

    @property
    def f(self): # 华氏温度
        if self._f is None:
            self._f = self.c2f(self._c)
        return self._f

    # 温度转换
    @classmethod
    def c2f(cls, c):
        return 9*c/5 + 32

    @classmethod
    def f2c(cls, f):
        return 5*(f-32)/9


    @classmethod
    def c2k(cls, c):
        return c + 273.15

    @classmethod
    def k2c(cls, k):
        return k - 273.15

    @classmethod
    def f2k(cls, f):
        return cls.c2k(cls.f2c(f))
```

```
    @classmethod
    def k2f(cls, k):
        return cls.c2f(cls.k2c(k))


print(Temperature.c2f(40))
print(Temperature.c2k(40))
print(Temperature.f2c(104.0))
print(Temperature.k2c(313.15))
print(Temperature.k2f(313.15))
print(Temperature.f2k(104))

t = Temperature(37)
print(t.c, t.k, t.f)

t = Temperature(300, 'k')
print(t.c, t.k, t.f)
```

5、模拟购物车购物

思路
购物车购物，分解得到两个对象 `购物车` 、 `物品` ，一个操作 `购买` 。
购买不是购物车的行为，其实是人的行为，但是对于购物车来说就是 `增加add` 。
商品有很多种类，商品的属性多种多样，怎么解决？
购物车可以加入很多不同的商品，如何实现？

```
class Color:
    RED = 0
    BLUE = 1
    GREEN = 2
    GOLDEN = 3
    BLACK = 4
    OTHER = 1000

class Item:
    def __init__(self, **kwargs):
        self.__spec = kwargs

    def __repr__(self):
```

```
        return str(sorted(self.__spec.items()))

class Cart:
    def __init__(self):
        self.items = []

    def additem(self,item:Item):
        self.items.append(item)

    def getallitems(self):
        return self.items

mycart = Cart()
myphone = Item(mark='Huawei', color=Color.GOLDEN, memory='4G')
mycart.additem(myphone)

mycar = Item(mark='Red Flag', color=Color.BLACK, year=2017)
mycart.additem(mycar)

print(mycart.getallitems())
```

注意，以上代码只是一个非常简单的一个实现，生产环境实现购物车的增删改查，要考虑很多。