作业

1、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
```

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。

先实现工具类

```python
# 温度转换工具类
class Temperature:
    # 温度转换
    @classmethod
    def c2f(cls, c):
        return 9 * c / 5 + 32

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

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

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

    # 华氏温度和开氏温度如何转换？

print(Temperature.c2f(40))
print(Temperature.f2c(104))
print(Temperature.c2k(40))
print(Temperature.k2c(313.15))
```

```python
# 温度转换工具类
class Temperature:
    # 温度转换
    @classmethod
    def c2f(cls, c):
        return 9 * c / 5 + 32

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

    @classmethod
```

```
    def c2k(cls, c):
        return c + 273.15

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

    # 华氏温度和开氏温度转换
    @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.f2c(104))
print(Temperature.c2k(40))
print(Temperature.k2c(313.15))
print(Temperature.f2k(104))
print(Temperature.k2f(313.15))
```

给定一个温度值，先存着，用的时候再转

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。

```
# 温度类，包含转换方法
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 == 'f':
            self._f = t
            self._c = self.f2c(t)
        elif unit == 'k':
            self._k = t
            self._c = self.k2c(t)
        else:
            self._c = t

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

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

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

```
30
31      @classmethod
32      def k2c(cls, k):
33          return k - 273.15
34
35      # 华氏温度和开氏温度如何转换？
36      @classmethod
37      def f2k(cls, f):
38          return cls.c2k(cls.f2c(f))
39
40      @classmethod
41      def k2f(cls, k):
42          return cls.c2f(cls.k2c(k))
43
44  print(Temperature.c2f(40))
45  print(Temperature.f2c(104))
46  print(Temperature.c2k(40))
47  print(Temperature.k2c(313.15))
48  print(Temperature.f2k(104))
49  print(Temperature.k2f(313.15))
50  print('-' * 30)
51
52  t = Temperature(104, 'f')
53  print(t.__dict__)
```

但是上面代码使用温度不方便，使用property装饰器构建属性

```
1   # 温度类，包含转换方法
2   class Temperature:
3       def __init__(self, t, unit='c'):
4           self._c = None
5           self._f = None
6           self._k = None
7
8           # 都要先转换到摄氏度，以后访问再计算其它单位的温度值
9           if unit == 'f':
10              self._f = t
11              self._c = self.f2c(t)
12          elif unit == 'k':
13              self._k = t
14              self._c = self.k2c(t)
15          else:
16              self._c = t
17
18      @property
19      def c(self):
20          return self._c
21
22      @property
23      def f(self): # 华氏温度
24          if self._f is None:
25              self._f = self.c2f(self._c)
26          return self._f
27
28      @property
29      def k(self): # 开氏温度
30          if self._k is None:
```

```
            self._k = self.c2k(self._c)
        return self._k

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

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

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

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

    # 华氏温度和开氏温度如何转换？
    @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.f2c(104))
print(Temperature.c2k(40))
print(Temperature.k2c(313.15))
print(Temperature.f2k(104))
print(Temperature.k2f(313.15))
print('-' * 30)

t = Temperature(104, 'f')
print(t.__dict__)
print(t.c, t.k, t.f)
print(t.__dict__)
```


2、图形

1、有Shape基类，要求所有子类都必须提供面积的计算，子类有三角形、矩形、圆。
2、上题圆类的数据可序列化

三角形面积——海伦公式：

$$p = (a+b+c)/2$$
$$S = \sqrt{p(p-a)(p-b)(p-c)}$$

Shape基类，要求所有子类都必须提供面积的计算，子类有三角形、矩形、圆。

```
import math
```

```
class Shape:
    @property
    def area(self):
        raise NotImplementedError('基类未实现')

class Triangle(Shape):
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    @property
    def area(self):
        p = (self.a + self.b + self.c) / 2
        return math.sqrt(p * (p-self.a) * (p-self.b) * (p-self.c))

class Rectangle(Shape):
    def __init__(self, width, height):
        self.width = width
        self.height = height

    @property
    def area(self):
        return self.width * self.height

class Circle(Shape):
    def __init__(self, radius):
        self.d = radius * 2

    @property
    def area(self):
        return math.pi * self.d * self.d * 0.25


shapes = [Triangle(3,4,5), Rectangle(3,4), Circle(4)]
for s in shapes:
    print('The area of {} = {}'.format(s.__class__.__name__,s.area))
```

上例中，每取一次面积就要计算一次。在上例基础上，可以做以下改动

```
import math

class Shape:
    def __init__(self):
        self._area = None

    @property
    def area(self):
        raise NotImplementedError('基类未实现')

class Triangle(Shape):
    def __init__(self, a, b, c):
        super().__init__()
        self.a = a
        self.b = b
        self.c = c
```

```
        self._p = (self.a + self.b + self.c) / 2

    @property
    def area(self):
        if self._area is None:
            p = self._p
            self._area = math.sqrt(p * (p-self.a) * (p-self.b) * (p-self.c))
        return self._area

class Rectangle(Shape):
    def __init__(self, width, height):
        super().__init__()
        self.width = width
        self.height = height

    @property
    def area(self):
        if self._area is None:
            self._area = self.width * self.height
        return self._area

class Circle(Shape):
    def __init__(self, radius):
        super().__init__()
        self.d = radius * 2

    @property
    def area(self):
        if self._area is None:
            self._area = 3.14 * self.d * self.d * 0.25
        return self._area


shapes = [Triangle(3,4,5), Rectangle(3,4), Circle(4)]
for s in shapes:
    print('The area of {} = {}'.format(s.__class__.__name__,s.area))
```

圆类的数据可序列化

```
import json
import msgpack

class SerializableMixin:
    def dumps(self, t='json'):
        if t == 'json':
            return json.dumps(self.__dict__)
        elif t == 'msgpack':
            return msgpack.packb(self.__dict__)
        else:
            raise NotImplementedError('没有实现的序列化')

class SerializableCircle(SerializableMixin, Circle):
    pass

scm = SerializableCircle(4)
```

```
print(scm.area)
s = scm.dumps('msgpack')
print(s)
```