python使用@property与__gertattr__创建动态类属性

最近在读mmdetection的源码，发现其中有很多的property与__gertattr__的使用，这里研究记录一下这两个方法的使用。

问题示例

一个例子的问题描述：对于一张图像：

- 属性： 宽， 高
- 方法：翻转，旋转

```
class Image:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def flip(self):
        pass
    def rotate(self):
        pass
```

如果需要添加一个属性长宽比怎么办：

方法一、添加self.aspect_ratio属性

最直接的方法，添加一个self.aspect_ratio属性：

```
class Image:
    def __init__(self, width, height):
        self.width = width
        self.height = height
        self.aspect_ratio = self.height / self.width

    def flip(self):
        pass
    def rotate(self):
        pass
```

但是对于这个代码执行如下的操作就会出现问题，当修改图像的宽或者高以后，图像的宽高比就会出错：

```
i = Image(1, 3)
print(i.width, i.height, i.aspect_ratio)  # 1  3  3.0
i.width = 4
print(i.width, i.height, i.aspect_ratio)  # 4  3  3.0
```

方法二、将宽高比定义为一个方法

另外一种正确的方法，将宽高比定义为一个方法：

```
class Image:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def flip(self):
        pass
    def rotate(self):
```

```
        pass
    def aspect_ratio(self):
        return self.height / self.width


i = Image(1, 3)
print(i.width, i.height, i.aspect_ratio())  # 1  3  3.0
i.width = 4
print(i.width, i.height, i.aspect_ratio()) # 4  3  0.75
```

这种方法虽然得到的结果正确，但是不是一个真正的属性方法。

方法三、 property装饰器

使用@property装饰器，神奇的将一个方法看起来像是一个属性：

```
class Image:
    def __init__(self, width, height):
        self.width = width
        self.height = height
        # self.aspect_ratio = self.height / self.width

    def flip(self):
        pass
    def rotate(self):
        pass
    @property
    def aspect_ratio(self):
        return self.height / self.width


i = Image(1, 3)
print(i.width, i.height, i.aspect_ratio)  # 1  3  3.0
```

```
i.width = 4
print(i.width, i.height, i.aspect_ratio ) # 4    3    0.75
```

方法四、使用__gertattr__方法获取属性

```
class Image:
    def __init__(self, width, height):
        self.width = width
        self.height = height
        # self.aspect_ratio = self.height / self.width

    def flip(self):
        pass
    def rotate(self):
        pass


    def __setattr__(self, name, value):
        if name == 'aspect_ratio':
            raise Exception('aspect_ratio can\'t be set. ')
        else:
            serlf.__dict__[name] = value

    def __getattr__(self, name):
        if name == 'aspect_ratio':
            return self.height / self.width
        else:
            raise AttributeError
```

其他，gertattr函数的使用方法

getattr获取属性的值

```
class Image:
    def __init__(self, width, height):
        self.width = width
        self.height = height
        # self.aspect_ratio = self.height / self.width

    def flip(self):
        pass
    def rotate(self):
        pass
    @property
    def height2(self):
        return getattr(self, 'height')


i = Image(1, 3)
print(i.width, i.height, i.height2)    # 1  3  3
```

如果getattr的name在勒种不存在这样的属性，那么将会报错，可以设置为一个默认值。

```
class Image:
    def __init__(self, width, height):
        self.width = width
        self.height = height
        # self.aspect_ratio = self.height / self.width

    def flip(self):
        pass
    def rotate(self):
        pass
    @property
```

```
    def height2(self):
        return getattr(self, 'height1',  0)


i = Image(1, 3)
print(i.width, i.height, i.height2)   # 1  3  0
```