参考：http://blog.jobbole.com/21351/

1. 什么是元类？
2. 使用type方法动态创建一个类?
3. 类的__class__属性？
4. 类的__metaclass__属性？
   当类对象定义了__metaclass__属性时，Python会在内存中通过__metaclass__创建类对象。如果Python没有找到__metaclass__，它会继续在父类中寻找__metaclass__属性，并尝试做和前面同样的操作。如果Python在任何父类中都找不到__metaclass__，它就会在模块层次中去寻找__metaclass__，并尝试做同样的操作。如果还是找不到__metaclass__,Python就会用内置的type来创建这个类对象。
5. 如何自定义元类？
   只要指定了类的__mataclas__属性就能够让元类对定义的类产生作用。

方式一:通过指定__metaclass__为一个可执行方法使用元类
下面的upper_attr方法能够将类属性名称全部改成大写。

```
def upper_attr(future_class_name, future_class_parents, future_class_attr):
    '''返回一个类对象，将属性都转为大写形式'''
    attrs = ((name, value) for name, value in future_class_attr.items() if not
    uppercase_attr = dict((name.upper(), value) for name, value in attrs)
    return type(future_class_name, future_class_parents, uppercase_attr)
```

```
class Foo(object):
    # 我们也可以只在这里定义__metaclass__，这样就只会作用于这个类中
    __metaclass__ = upper_attr
    bar = 'bip'

# 此时Foo.bar属性不存在，取而代之的是Foo.BAR属性
print hasattr(Foo, 'bar')
print hasattr(Foo, 'BAR')

```

方式二：通过继承type类型，并且定义__new__方法

```
class UpperAttrMetaclass(type):
    def __new__(cls, name, bases, dct):
        attrs = ((name, value) for name, value in dct.items() if not name.star
        uppercase_attr = dict((name.upper(), value) for name, value in attrs)
        return super(UpperAttrMetaclass, cls).__new__(cls, name, bases, upperc

class Foo(object):
    # 我们也可以只在这里定义__metaclass__，这样就只会作用于这个类中
    __metaclass__ = UpperAttrMetaclass
    bar = 'bip'
```

1. 使用six.add_metaclass为一个类添加元类（主要处理兼容性问题）