```
'''Auth模块'''
Auth模块是Django自带的用户认证模块

# auth模块常用方法
    from django.contrib import auth
    authenticate()
    # 验证用户名以及密码是否正确，一般需要username 、password两个关键字参数
    # 如果认证成功（用户名和密码正确有效），便会返回一个 User 对象
    #authenticate()会在该 User 对象上设置一个属性来标识后端已经认证了该用户
        #且该信息在后续的登录过程中是需要的。
    user_obj = authenticate(username='usernamer',password='password')

    '''login(HttpRequest, user)'''
    # 该函数接受一个HttpRequest对象，以及一个经过认证的User对象
    # 该函数实现一个用户登录的功能,本质上会在后端为该用户生成相关session数据
    from django.contrib.auth import authenticate, login

    def my_view(request):
      username = request.POST['username']
      password = request.POST['password']
      user = authenticate(username=username, password=password)
      if user is not None:
        login(request, user)
        # Redirect to a success page.
        ...
      else:
        # Return an 'invalid login' error message.
        ...

    '''logout(request)'''
    # 该函数接受一个HttpRequest对象，无返回值
    # 当调用该函数时，当前请求的session信息会全部清除
    # 该用户即使没有登录，使用该函数也不会报错
    from django.contrib.auth import logout

    def logout_view(request):
      logout(request)
      # Redirect to a success page.

    '''is_authenticated()'''
    # 用来判断当前请求是否通过了认证
    def my_view(request):
      if not request.user.is_authenticated():
        return redirect('%s?next=%s' % (settings.LOGIN_URL, request.path))

    '''login_requierd()'''
    # auth 给我们提供的一个装饰器工具，用来快捷的给某个视图添加登录校验
    from django.contrib.auth.decorators import login_required

    @login_required
    def my_view(request):
      ...
    #若用户没有登录，则会跳转到django默认的 登录URL '/accounts/login/ ' 并传递当前访问url的绝对路径
        #(登陆成功后，会重定向到该路径)
    # 如果需要自定义登录的URL，则需要在settings.py文件中通过LOGIN_URL进行修改
    #LOGIN_URL = '/login/'  # 这里配置成你项目登录页面的路由

    '''create_user()'''
```

```
#auth 提供的一个创建新用户的方法，需要提供必要参数（username、password）等
from django.contrib.auth.models import User
user = User.objects.create_user（username='用户名',password='密码',email='邮箱',...）

'''create_superuser()'''
#auth 提供的一个创建新的超级用户的方法，需要提供必要参数（username、password）等
from django.contrib.auth.models import User
user = User.objects.create_superuser（username='用户名',password='密码',email='邮箱',...）

'''check_password(password)'''
# auth 提供的一个检查密码是否正确的方法，需要提供当前请求用户的密码
# 密码正确返回True，否则返回False。
ok = user.check_password('密码')

'''set_password(password)'''
#auth 提供的一个修改密码的方法，接收 要设置的新密码 作为参数
#注意：设置完一定要调用用户对象的save方法！！！
user.set_password(password='')
user.save()

is_staff          # 用户是否拥有网站的管理权限.
is_active         # 是否允许用户登录, 设置为 False，可以在不删除用户的前提下禁止用户登录


#扩展默认的auth_user表

from django.contrib.auth.models import AbstractUser
class UserInfo(AbstractUser):
    """
    用户信息表
    """
    nid = models.AutoField(primary_key=True)
    phone = models.CharField(max_length=11, null=True, unique=True)

    def __str__(self):
        return self.username

#注意：
按上面的方式扩展了内置的auth_user表之后，一定要在settings.py中告诉Django
    # 引用Django自带的User表，继承使用时需要设置
    AUTH_USER_MODEL = "app名.UserInfo"
        # 一旦我们指定了新的认证系统所使用的表，我们就需要重新在数据库中创建该表
        # 而不能继续使用原来默认的auth_user表了
```