什么是cookie：

在网站中，http请求是无状态的。也就是说即使第一次和服务器连接后并且登录成功后，第二次请求服务器依然不能知道当前请求是哪个用户。cookie的出现就是为了解决这个问题，第一次登录后服务器返回一些数据（cookie）给浏览器，然后浏览器保存在本地，当该用户发送第二次请求的时候，就会自动的把上次请求存储的cookie数据自动的携带给服务器，服务器通过浏览器携带的数据就能判断当前用户是哪个了。cookie存储的数据量有限，不同的浏览器有不同的存储大小，但一般不超过4KB。因此使用cookie只能存储一些小量的数据。

1. cookie有有效期：服务器可以设置cookie的有效期，以后浏览器会自动的清除过期的cookie。
2. cookie有域名的概念：只有访问同一个域名，才会把之前相同域名返回的cookie携带给服务器。也就是说，访问谷歌的时候，不会把百度的cookie发送给谷歌。

flask操作cookie：

1. 设置cookie：设置cookie是应该在Response的对象上设置。flask.Response对象有一个set_cookie方法，可以通过这个方法来设置cookie信息。

在Chrome浏览器中查看cookie的方式：

* 右键->检查->Network->重新加载页面->找到请求，然后查看Response Headers中的cookie
* 点击url输入框左边的信息icon，然后找到相应的域名，再展开查看cookie。
* 在Chrome的设置界面->高级设置->内容设置->所有cookie->找到当前域名下的cookie。

2. 删除cookie：通过Response.delete_cookie，指定cookie的key，就可以删除cookie了。
3. 设置cookie的有效期：
    * max_age：以秒为单位，距离现在多少秒后cookie会过期。
    * expires：为datetime类型。这个时间需要设置为格林尼治时间，也就是要距离北京少8个小时的时间。
    * 如果max_age和expires都设置了，那么这时候以max_age为标准。
    * max_age在IE8以下的浏览器是不支持的。expires虽然在新版的HTTP协议中是被废弃了，但是到目前为止，所有的浏览器都还是能够支持，所以如果想要兼容IE8以下的浏览器，那么应该使用expires，否则可以使用max_age。
    * 如果没有显示的指定过期时间，那么这个cookie将会在浏览器关闭(浏览会话结束)后过期。

示例代码：

```
@app.route('/')
def hello_world():
    resp = Response("知了课堂")
    expires = datetime(year=2017,month=12,day=11,hour=16,minute=0,second=0)
    # 使用expires参数，就必须使用格林尼治时间
    # 要相对北京时间少8个小时
    expires = datetime.now() + timedelta(days=30,hours=16)
    # 在新版本的http协议中，expires参数视为被废弃
    # max_age，在IE8一下的浏览器中是不支持的
    # resp.set_cookie('username','zhiliao',expires=expires,max_age=60)
    resp.set_cookie('username','zhiliao')
    return resp
```

4. 设置cookie的有效域名：cookie默认是只能在主域名下使用。如果想要在子域名下使用，那么应该给set_cookie传递一个domain='.reidwang.com'，这样其他子域名才能访问到这个cookie信息。

session：

1. session的基本概念：session和cookie的作用有点类似，都是为了存储用户相关的信息。不同的是，cookie是存储在本地浏览器，session是一个思路、一个概念、一个服务器存储授权信息的解决方案，不同的服务器，不同的框架，不同的语言有不同的实现。虽然实现不一样，但是他们的目的都是服务器为了方便存储数据的。session的出现，是为了解决cookie存储数据不安全的问题的。
2. session与cookie的结合使用：
    * session存储在服务器端：服务器端可以采用mysql、redis、memcached等来存储session信息。原理是，客户端发送验证信息过来（比如用户名和密码），服务器验证成功后，把用户的相关信息存储到session中，然后随机生成一个唯一的session_id，再把这个session_id存储cookie中返回给浏览器。浏览器以后再请求我们服务器的时候，就会把这个session_id自动的发送给服务器，服务器再从cookie中提取session_id，然后从服务器的session容器中找到这个用户的相关信息。这样就可以达到安全识别用户的需求了。
    * session存储到客户端：原理是，客户端发送验证信息过来（比如用户名和密码）。服务器把相关的验证信息进行一个非常严格和安全的加密方式进行加密，然后再把这个加密后的信息存储到cookie，返回给浏览器。以后浏览器再请求服务器的时候，就会自动的把

cookie发送给服务器，服务器拿到cookie后，就从cookie找到加密的那个session信息，然后也可以实现安全识别用户的需求了。

flask操作session：

1. 设置session：通过flask.session就可以操作session了。操作session就跟操作字典是一样的。session['username']='zhiliao'。
2. 获取session：也是类似字典，session.get(key)。
3. 删除session中的值：也是类似字典。可以有三种方式删除session中的值。
    * session.pop(key)。
    * del session[key]。
    * session.clear()：删除session中所有的值。
4. 设置session的有效期：如果没有设置session的有效期。那么默认就是浏览器关闭后过期。如果设置session.permanent=True，那么就会默认在31天后过期。如果不想在31天后过期，那么可以设置app.config['PERMANENT_SESSION_LIFETIME'] = timedelta(hour=2)在两个小时后过期。