**session：**

**session技术 也叫 会话技术。**

**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，都是为了解决http协议无状态的这个特点。不同的是，cookie信息是存储在客户端，而session信息是存储在服务器端。

需要注意的是，不同的语言，不同的框架，有不同的实现。虽然底层的实现不完全一样，但目的都是让服务器端能方便的存储数据而产生的。

session的出现，是为了解决cookie存储数据不安全的问题的。

**2. session的跟踪机制跟cookie有关：**

Flask框架中，session的跟踪机制跟Cookie有关，这也就意味着脱离了Cookie，session就不好使了。

因为session跟踪机制跟cookie有关，所以，要分服务器端和客户端分别起到什么功能来理解。

**\* session工作过程：**

服务器端可以采用类似于mysql、redis等技术来存储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服务器端的session中（可想象为一个容器），

再通过盐的机制，盐起到混淆原数据的作用（类似于加密），

然后随机生成一个唯一的session\_id，用来标识(用户名和密码)并存储到session中，

之后再把这个session\_id存储到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通过Cookie技术自动的发送给服务器，

服务器端再从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。

存储在服务器的数据会更加的安全，不容易被窃取。

但存储在服务器也有一定的弊端，就是会占用服务器的资源，但现在服务器已经发展至今，一些session信息还是绰绰有余的。

**\*服务器端功能：**

**1.**把用户的相关信息存储到服务器端的session中

**2.**通过盐的机制加密随机生成一个唯一的session\_id,用来标识用户相关信息并将session\_id也存入session中

**3.**把session作为cookie的key，session\_id作为cookie的value创建cookie信息并返回给客户端

**4.**客户端发送第二次以后的请求时，获取cookie信息，与服务器端session容器中session\_id对比，得出指定 用户信息。

**\* 客户端功能：**

**1.**通过cookie存储session加密后的session\_id信息

**2.**以后浏览器再请求服务器的时候，就会自动的把cookie信息（包含session\_id）发送给服务器

**3.扩展面试题：**

若客户端禁用了浏览器的Cookie功能，session功能想继续保留，该咋整？给出你的实现思路(能代码实现最好)