**深信服面试题**

1. python的数据类型,它们之间的区别和应用场景
2. 整型包括：int 这种数据类型对应的开辟的内存大小为4B

long 对应开辟的内存为8B

注：python中只有int, 具有long的性质

1. 浮点型 float
2. 复数型 complex

复数具有实部和虚部，如：3 + 4j

1. 布尔类型 bool

描述状态的类型值只有True和False，也可以使用1和0,非0即为True

1. 字符串类型str

字符串类型数据的标识：使用单引号/双引号包含的单个或者多个字符

字符：用引号包含的一个符号

1. 列表类型list

存放的是多个变量的容器，标识为：[]

1. 元祖类型tuple

存放的是多个变量的容器，标识：()

和list的区别：list中的元素的值是可变的，元祖中元素的值不可变

共同点：添加在容器中的元素都有自己的编号，这个编号是从0开始的，这个编号也称之为索引、下标或者脚标

注意：如果元祖中只有一个元素，该元素后面需要添加逗号分隔符

原因：()还具有另一层意思：提高优先级的问题，如果不添加逗号，python解释器会将其视为提高优先级的小括号，如果加上逗号只是告诉解释器这个是一个元祖类型

tuple1 = (10,)

print(tuple1)

print(type(tuple1))

1. 字典类型dict

存放的数据是键值对格式的，键和值是一一对应的，根据键来查找值的

要求：键是不允许重复的，是唯一的

键的类型对应的数据必须是不可变的

字典的标识：{} 如：{key:value,key1:value2}

1. 集合set

可理解为存放字典的容器，集合中的元素不允许重复，可用于排重

如：list = [99,35,54,21,43,56]

1. 空值类型

只有一个值None，代表的状态是空

如：a = None，表示该变量没有指向任何地址

什么时候使用空值？

变量在使用之前要先进行声明和赋值初始化，在赋予初始化时不知道该给变量赋予什么值就可以赋予None

1. tcp/ip的理解,三次握手,四次挥手

Transmission Control Protocol/Internet Protocol的简写，即**传输控制协议/因特网互联协议**，又称**网络通讯协议**，是Internet最基本的协议、Internet国际互联网络的基础，由网络层的IP协议和传输层的TCP协议组成。TCP/IP 定义了电子设备如何连入[因特网](https://baike.baidu.com/item/%E5%9B%A0%E7%89%B9%E7%BD%91/114119" \t "https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE/_blank)，以及数据如何在它们之间传输的标准。协议采用了4层的层级结构（网络访问层、互联网层、传输层、应用层），每一层都呼叫它的下一层所提供的协议来完成自己的需求。[通俗](https://baike.baidu.com/item/%E9%80%9A%E4%BF%97/10621346" \t "https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE/_blank)而言：TCP负责发现[传输](https://baike.baidu.com/item/%E4%BC%A0%E8%BE%93/7078195" \t "https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE/_blank)的问题，一有问题就发出信号，要求重新传输，直到所有[数据安全](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%AE%89%E5%85%A8/3204964" \t "https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE/_blank)正确地传输到目的地。而IP是给[因特网](https://baike.baidu.com/item/%E5%9B%A0%E7%89%B9%E7%BD%91/114119" \t "https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE/_blank)的每一台联网设备规定一个地址。

**层级结构：**

（1）网络访问层(Network Access Layer)在TCP/IP参考模型中并没有详细描述，只是指出主机必须使用某种协议与网络相连；

（2）互联网层(Internet Layer)是整个体系结构的关键部分，其功能是使主机可以把分组发往任何网络，并使分组独立地传向目标；

（3）传输层(Tramsport Layer)使源端和目的端机器上的对等实体可以进行会话（TCP和UDP）；

（4）应用层(Application Layer)包含所有的高层协议，包括：虚拟终端协议(TELNET，TELecommunications NETwork)、文件传输协议(FTP，File Transfer Protocol)、电子邮件传输协议(SMTP，Simple Mail Transfer Protocol)、域名服务(DNS，Domain Name Service)、网上新闻传输协议(NNTP，Net News Transfer Protocol)和超文本传送协议(HTTP，HyperText Transfer Protocol)等。

**注意**：TCP/IP协议不是[TCP](https://baike.baidu.com/item/TCP" \t "https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE/_blank)和[IP](https://baike.baidu.com/item/IP" \t "https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE/_blank)这两个协议的合称，而是指因特网整个TCP/IP协议族

**特点：**

（1）TCP/IP协议不依赖于任何特定的计算机硬件或操作系统，提供开放的协议标准，即使不考虑Internet，TCP/IP协议也获得了广泛的支持。所以TCP/IP协议成为一种联合各种硬件和软件的实用系统。

（2）TCP/IP协议并不依赖于特定的网络传输硬件，所以TCP/IP协议能够集成各种各样的网络。用户能够使用以太网（Ethernet）、[令牌环网](https://baike.baidu.com/item/%E4%BB%A4%E7%89%8C%E7%8E%AF%E7%BD%91" \t "https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE/_blank)（Token Ring Network）、拨号线路（Dial-up line）、[X.25](https://baike.baidu.com/item/X.25" \t "https://baike.baidu.com/item/TCP%2FIP%E5%8D%8F%E8%AE%AE/_blank)网以及所有的网络传输硬件。

（3）统一的网络地址分配方案，使得整个TCP/IP设备在网中都具有惟一的地址

（4）标准化的高层协议，可以提供多种可靠的用户服务。

**TCP建立连接时会经历“三次握手”：**

第一次：发送端——> 向接受端发送一个连接请求，给接收端一个标记，要建立连接

第二次：接收端接收到连接请求——> 给发送端回应一个接受连接的状态，并告知允许连接

第三次：发送端接收到接收端回馈的连接信息——> 向接收端发送状态确定连接

比如：打电话 拨号 ——> 发送连接请求

对方看到来电，接听电话“hello”

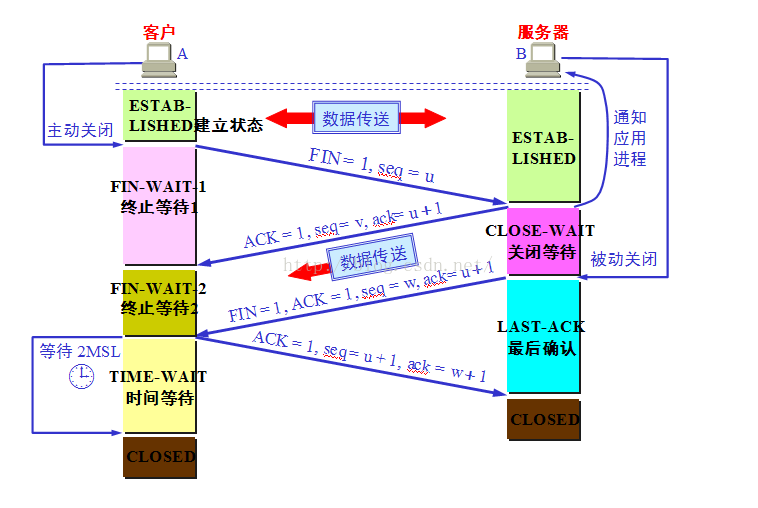
发起者，回应一声“hello”

“三次握手”的**意义**：通过“三次握手”保证了客户端与服务器建立的是双工连接，而可靠性更多是通过重发机制实现的。

**“四次挥手”：**关闭TCP连接的过程，就是指断开一个TCP连接时，需要客户端和服务端总共发送4个包以确认双方连接的断开。

意义：保证TCP连接的全双工连接，这一过程由客户端或者服务端任一方执行close来触发。

流程如下图：



**扩展：进行网络通信的三要素：IP地址、端口号、通信协议**

**IP地址**

IP地址是一个32位的二进制数，把32份分成四份，每一份8位，由于IP地址没有负数，所以每份的取值范围是0-255，IP地址分为ipv4和ipv6(128b),ipv6的网段范围会更广

IP地址的作用：1、用于标识网络通信上的计算机；

1. IP地址有一个特殊网段127.0.0.0--127.255.255.255,用于标识自己的电脑，一般使用127.0.0.1来表示本机IP地址，称为回环地址

**端口号**

端口号是16b的二进制数，取值范围为：0-65535

作用：1、用于定位通信的应用程序；

2、相同协议下，不同应用程序的端口号不能重复，会造成数据在不同的应用程序之间进行传递；

3、尽量不要使用1024以下的端口号，这些端口号要么被一些知名服务占用，要么被预留

**通信协议**

制定的通信的规则：**TCP** 和 **UDP ——> 传输层**

**TCP ：传输控制协议（Transmissiom Control Protocol）**

特点：1、面向连接，数据通信之前必须建立连接；

2、数据是以数据流的形式传递的；

3、数据传递时可以双方指定数据大小的规则，传递大数据；

4、是一个可靠的协议；

5、传递速率慢

**UDP ：用户数据协议 （User Datagram Protocol）**

特点：1、面向无连接的，数据传输端和接收端可以不建立连接；

2、发送数据大小的限制: 64K之内；

3、不可靠的协议；

4、传递速率快；

5、数据传递时是通过数据包的形式传递的，这个数据包的大小是在64k之内的

运用如：视频聊天

1. 请用python代,码实现生产者消费者模型（仅限使用python标准库提供的组件，勿使用第三方库）

思路：

产者消费之模型就是,比如一个早餐包子铺,顾客吃包子和厨师做包子,不可能是将包子一次全做出来,在给顾客吃,但是单线程只能这麽做,所以用多线程来执行,厨师一边做包子,顾客一边吃包子,当顾客少时,厨师做的包子就放在一个容器中,等着顾客来吃,当顾客多的时候,就从容器中先取出来给顾客吃,厨师继续做包子用队列来模拟这个容器

*# 当做完一屉包子后就要给顾客发送一个信号,表示已经做完,让他们吃包子***import** threading, time, queue  
  
q = queue.Queue()  
  
**def** Produce(name):  
 count = 1 *# conut表示做的包子总屉数* **while** count < 10:  
 print(**'厨师%s在做包子中...'** % name)  
 time.sleep(2)  
 q.put(count) *# 容器中添加包子  
 # 当做完一屉包子后就要给顾客发送一个信号,表示已经做完,让他们吃包子* print(**'produce%s已经做好了第%s屉包子'** % (name, count))  
 count += 1  
 print(**'oking...'**)  
  
**def** Consumer(name):  
 count = 0 *# count表示包子被吃的总屉数* **while** count < 10:  
 time.sleep(2) *# 排队去取包子,* **if not** q.empty(): *# 如果存在* data = q.get() *# 取包子, 吃包子* print(**'Consumer %s已经把第%s屉包子吃了...'** % (name, data))  
 count += 1  
 **else**:  
 print(**'包子被吃完了...'**)  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 p1 = threading.Thread(target=Produce, args=(**'面点师傅'**,))  
 c1 = threading.Thread(target=Consumer, args=(**'张三'**,))  
 c2 = threading.Thread(target=Consumer, args=(**'李四'**,))  
 p1.start()  
 c1.start()  
 c2.start()

*# 2这里就是当顾客吃完了然后给生产者发送一个信号当生产者就接收到信号时, 继续做包子***import** threading, time, queue  
  
q = queue.Queue()  
**def** Produce(name):  
 count = 1 *# conut表示做的包子总个数* **while** count < 10:  
 print(**'厨师%s在做包子中...'** % name)  
 time.sleep(2)  
 q.put(count) *# 容器中添加包子* print(**'produce%s 已经做好了第%s屉包子'** %(name,count))  
 count += 1  
  
 *# q.task\_done() # 当做完一个包子后就要给顾客发送一个信号,表示已经做完,让他们吃包子* q.join() *# 等待接收信号,* print(**'ok...'**)  
  
**def** Consumer(name):  
 count = 0 *# count表示包子被吃的总个数* **while** count < 10:  
 time.sleep(2)  
 *# print('waiting...')  
 # q.join()* data = q.get() *# 取包子, 吃包子* print(**'%seating...'** % name)  
 time.sleep(4) *# 吃包子用了4s然后给厨师发送一个信号* q.task\_done()  
  
 print(**'\033[32;1mConsumer %s已经把第%s屉包子吃了...\033[0m'** % (name, data))  
 print(**'包子被吃完了...'**)  
 count += 1  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 p1 = threading.Thread(target=Produce, args=(**'面点师傅'**,))  
 c1 = threading.Thread(target=Consumer, args=(**'张三'**,))  
 c2 = threading.Thread(target=Consumer, args=(**'李四'**,))  
   
 p1.start()  
 c1.start()  
 c2.start()