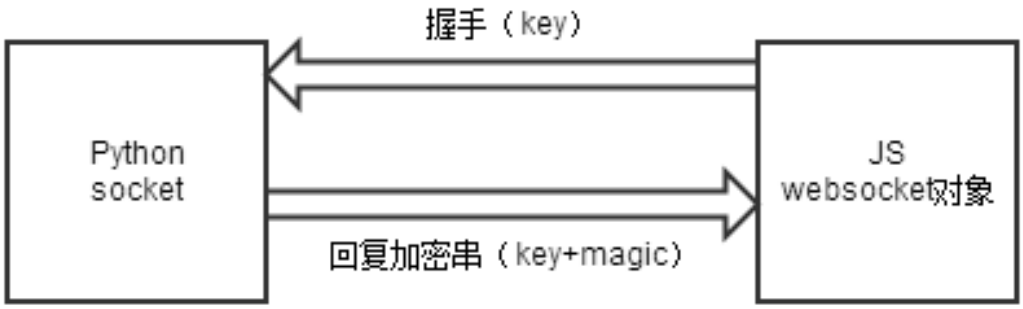
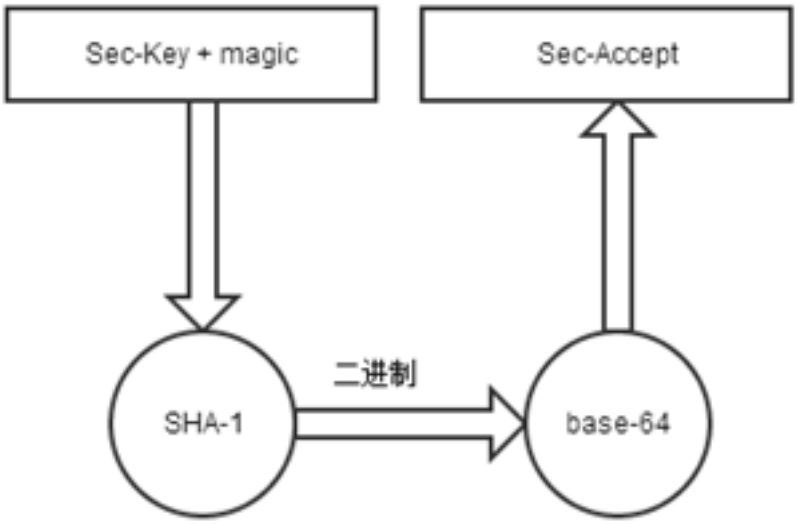
websocket 握手



js 客户端先向服务器端 python 发送握手包，格式如下：
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13

服务器回应包格式：
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat

其中， Sec-WebSocket-Key 是随机的，服务器用这些数据构造一个 SHA-1 信息摘要。
方法为： key+magic ， SHA-1 加密， base-64 加密，如下：



Python 中的处理代码
MAGIC_STRING = '258EAFA5-E914-47DA-95CA-C5AB0DC85B11'
res_key = base64.b64encode(hashlib.sha1(sec_key + MAGIC_STRING).digest())

握手 JS 代码

js 中有处理 websocket 的类，初始化后自动发送握手包，如下：
var socket = new WebSocket('ws://localhost:port ');

握手 Python Server 代码

```
HOST = 'localhost'
PORT = 3368
MAGIC_STRING = '258EAFA5-E914-47DA-95CA-C5AB0DC85B11'
HANDSHAKE_STRING = "HTTP/1.1 101 Switching Protocols\r\n" \
        "Upgrade:websocket\r\n" \
        "Connection: Upgrade\r\n" \
        "Sec-WebSocket-Accept: {1}\r\n" \
        "WebSocket-Location: ws://{2}/chat\r\n" \
        "WebSocket-Protocol:chat\r\n\r\n"

def handshake(con):
#con 为用 socket，accept()得到的 socket
#这里省略监听，accept 的代码,具体可见 blog：
http://blog.csdn.net/ice110956/article/details/29830627
  headers = {}
  shake = con.recv(1024)

  if not len(shake):
```

```
    return False

  header, data = shake.split('\r\n\r\n', 1)
  for line in header.split('\r\n')[1:]:
    key, val = line.split(': ', 1)
    headers[key] = val

  if 'Sec-WebSocket-Key' not in headers:
    print ('This socket is not websocket, client close.')
    con.close()
    return False

  sec_key = headers['Sec-WebSocket-Key']
  res_key = base64.b64encode(hashlib.sha1(sec_key + MAGIC_STRING).digest())

  str_handshake = HANDSHAKE_STRING.replace('{1}', res_key).replace('{2}',
HOST + ':' + str(PORT))
  print str_handshake
  con.send(str_handshake)
return True
```

接受数据

请求包格式

固定字节：
（ 1000 0001 或是 1000 0002 ）这里没用，忽略
包长度字节：
第一位肯定是 1 ，忽略。剩下 7 个位可以得到一个整数 (0 ~ 127) ，其中（ 1-125 ）表此字节为长度字节，大小即为长度；
（126）表接下来的两个字节才是长度；
（127）表接下来的八个字节才是长度；
用这种变长的方式表示数据长度，节省数据位。
mark 掩码：
mark 掩码为包长之后的 4 个字节，之后的兄弟数据要与 mark 掩码做运算才能得到真实的数据。
兄弟数据：
得到真实数据的方法：将兄弟数据的每一位 x ，和掩码的第 i%4 位做 xor 运

算，其中 i 是 x 在兄弟数据中的索引。

接受数据的代码：

```
def recv_data(self, num):
  try:
    all_data = self.con.recv(num)
    if not len(all_data):
      return False
  except:
    return False
  else:
    code_len = ord(all_data[1]) & 127
    if code_len == 126:
      masks = all_data[4:8]
      data = all_data[8:]
    elif code_len == 127:
      masks = all_data[10:14]
      data = all_data[14:]
    else:
      masks = all_data[2:6]
      data = all_data[6:]
    raw_str = ""
    i = 0
    for d in data:
      raw_str += chr(ord(d) ^ ord(masks[i % 4]))
      i += 1
    return raw_str
```

返回包格式

固定字节：固定的 1000 0001('\x81')
包长：根据发送数据长度是否超过 125 ， 0xFFFF(65535) 来生成 1 个或 3 个或 9 个字节，来代表数据长度。

```
def send_data(self, data):
  if data:
    data = str(data)
  else:
```

```
  return False
 token = "\x81"
 length = len(data)
 if length < 126:
  token += struct.pack("B", length)
 elif length <= 0xFFFF:
  token += struct.pack("!BH", 126, length)
 else:
  token += struct.pack("!BQ", 127, length)
 #struct 为 Python 中处理二进制数的模块，二进制流为 C，或网络流的形式。
 data = '%s%s' % (token, data)
 self.con.send(data)
 return True
```

参考文献资料：
Python 通过 websocket 与 js 客户端通信示例分析
http://www.jb51.net/article/51516.htm