



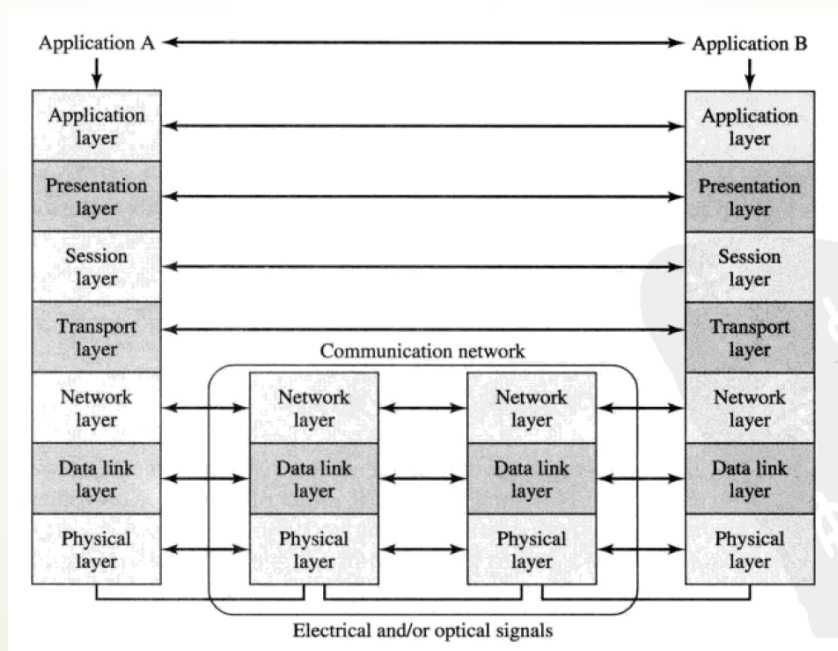
第12课 网络程序设计

socket

requests

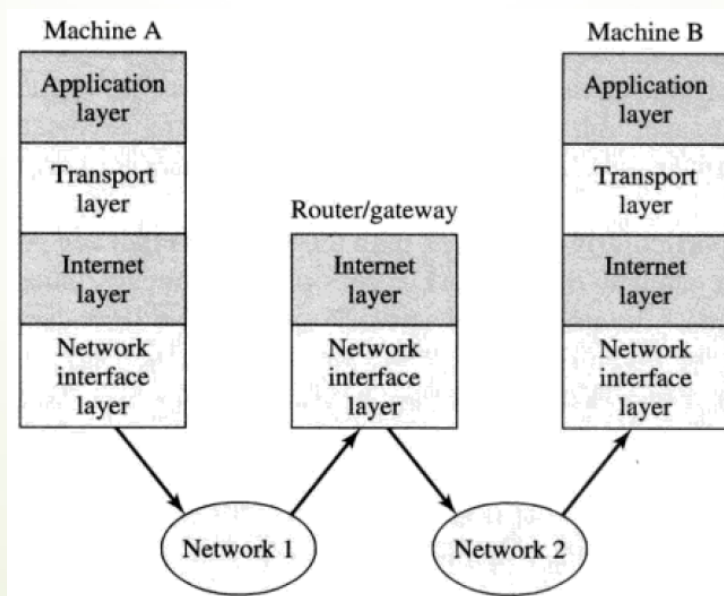
计算机网络基础知识

- 网络体系结构：目前主流的网络体系结构是ISO/OSI参考模型
- 这两种体系都采用了分层设计与实现的方式
- ISO/OSI参考模型从上而下分为应用层、表示层、会话层、传输层、网络层、数据链路层、物理层



计算机网络基础知识

- TCP/IP将网络划分为应用层、传输层、网络层、链路层
- TCP: Transmission Control Protocol ; IP: Internet Protocol
- 分层设计的好处: 各层可以独立设计和实现, 只要保证相邻层之间的调用规范和调用接口一样, 就可以方便、灵活地改变某一层的内部实现。
- 例如, 不管是有线网还是无线网 (Network), 网络层都使用IP



网络协议

- **网络协议**是计算机网络中进行数据交换而建立的规则、标准或者约定的集合。网络协议的三要素：
- 语法：规定了用户数据与控制信息的结构与格式

IP数据包

0	4	8	16	19	24	31
Version	IHL	Type of service	Total length			
Identification			Flags	Fragment offset		
Time to live		Protocol	Header checksum			
Source IP address						
Destination IP address						
Options					Padding	

- 语义：解释控制信息每个部分的意义，需要完成什么动作和响应。
 - 是什么协议（Protocol）？应该怎么做？
 - 是发给我的信息吗（IP Address）？不是我的要怎么做？
- 时序：对事件发生顺序的详细说明。

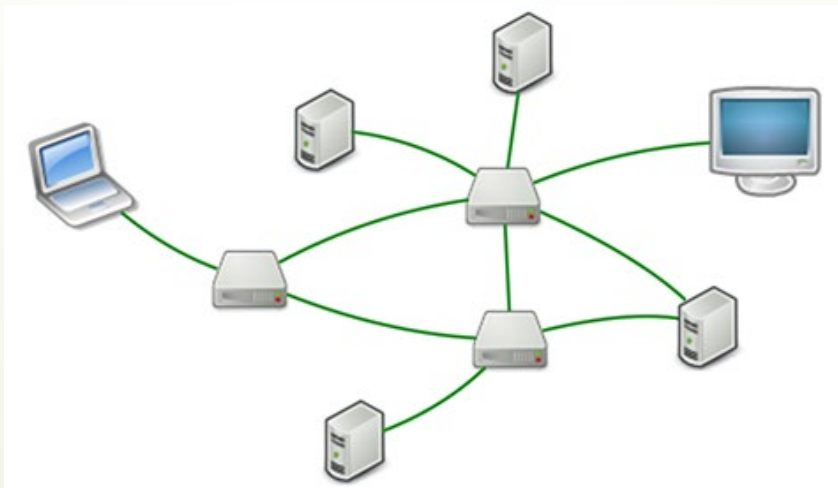
常见的网络协议

- 应用层协议：
 - DNS：域名服务，实现域名与IP地址的转换
 - FTP：文件传输协议
 - HTTP：超文本传输协议
 - SMTP：简单邮件传输协议
 - 等等等等
- 传输层协议：
 - TCP（Transmission Control Protocol）：面向连接的，具有质量保证的，可靠传输协议，开销较大
 - UDP：尽最大能力传输的无连接协议，开销小
 - TCP和UDP没有优劣之分，仅仅是使用的场合不一样

IP

TCP/IP的IP:

- IP运行于网络体系结构的网络层，是网络互联的重要基础。
- IP协议负责把数据从一台计算机通过网络发送到另一台计算机。
- 数据被分割成一小块一小块，然后通过IP包发送出去。
- 由于互联网链路复杂，两台计算机之间经常有多条线路，因此，路由器就负责决定如何把一个IP包转发出去。
- IP包的特点是按块发送，途径多个路由，但不保证能到达，也不保证顺序到达。



IP地址与MAC地址

- IP地址：用来标识网络上的主机：在一个公开网络上或者同一个局域网内部，每台主机都必须使用不同的IP地址（不同内网之间IP地址可以相同，详见网络地址转换 NAT）
- IP地址与端口号共同来表示网络上特定主机上的特定应用进程，俗称socket
- MAC地址：也称网卡地址或物理地址。

端口作用：在两台计算机通信时，只发IP地址是不够的，因为同一台计算机上跑着多个网络程序。一个数据包来了之后，到底是交给微信还是QQ，就需要端口号来区分。

```
(base) C:\Users\Administrator>ipconfig /all

Windows IP 配置

主机名 . . . . . : win7-PC
主 DNS 后缀 . . . . . : 
节点类型 . . . . . : 混合
IP 路由已启用 . . . . . : 否
WINS 代理已启用 . . . . . : 否

以太网适配器 本地连接 3:

    连接特定的 DNS 后缀 . . . . . : 
    描述 . . . . . : Realtek PCIe GBE Family Controller #2
    物理地址. . . . . : 94-C6-91-09-C1-32
    DHCP 已启用 . . . . . : 是
    自动配置已启用 . . . . . : 是
    IPv6 地址 . . . . . : 2001:da8:2d00:2074:d548:785a:d07e:d264(首选)
    临时 IPv6 地址 . . . . . : 2001:da8:2d00:2074:485:dbb1:522:48eb(受到抨击)
    临时 IPv6 地址 . . . . . : 2001:da8:2d00:2074:6577:d931:27a4:5b3e(受到抨击)
    临时 IPv6 地址 . . . . . : 2001:da8:2d00:2074:741a:fe23:d28:c72(受到抨击)
    临时 IPv6 地址 . . . . . : 2001:da8:2d00:2074:757c:a5d8:67e2:e471(受到抨击)
    临时 IPv6 地址 . . . . . : 2001:da8:2d00:2074:88d1:925f:7f05:e8b8(首选)
    临时 IPv6 地址 . . . . . : 2001:da8:2d00:2074:a010:a13a:54a8:8cc7(受到抨击)
    本地连接 IPv6 地址 . . . . . : fe80::d548:785a:d07e:d264%8(首选)
    IPv4 地址 . . . . . : 172.31.74.70(首选)
    子网掩码 . . . . . : 255.255.255.0
    获得租约的时间 . . . . . : 2020年5月27日 14:23:02
    租约过期的时间 . . . . . : 2020年6月2日 12:08:13
    默认网关 . . . . . : fe80::366b:5bff:fef0:122f%8
    . . . . . : 172.31.74.33
    DHCP 服务器 . . . . . : 172.31.74.33
    DHCPv6 IAD . . . . . : 292309904
    DHCPv6 客户端 DUID . . . . . : 00-01-00-01-26-4E-88-7F-94-C6-91-09-C1-32
    DNS 服务器 . . . . . : 192.168.247.6
    . . . . . : 192.168.247.26
    TCP/IP 上的 NetBIOS . . . . . : 已启用
```

socket

- Socket是计算机之间进行网络通信的一套程序接口
- 网络编程的标准：可以实现跨平台的数据传输
- Socket是网络编程的一个抽象概念：Socket相当于在发送端和接收端之间建立一个管道来实现数据和命令的相互传递
- Python提供了socket模块，支持Socket接口的访问

```
import socket
```

- 使用socket模块进行UDP和TCP编程

UDP协议编程

- UDP适用于对效率要求相对较高而对准确性要求相对较低的场合，例如视频在线点播、网络语音通话等等。
- UDP属于无连接协议，在UDP编程时不需要首先建立连接，而是直接向接收方发送信息。
- socket模块中经常用于UDP编程的方法主要有：
 - ✓ `socket(family,type)`: 创建一个socket对象，
 - ✓ `family`为`socket.AF_INET`表示IPV4，`socket.AF_INET6`表示IPV6；
 - ✓ `type`为`SOCK_STREAM`表示TCP协议，`SOCK_DGRAM`表示UDP协议。
 - ✓ `sendto(string,address)`: 把`string`指定的字符串内容发送给`address`指定的地址，`address`是一个元组，格式为(IP地址,端口号)
 - ✓ `recvfrom(bufsize[,flags])`: 接收数据

UDP协议编程

- 例子：编写UDP通信程序，发送端发送一个字符串“Hello world!”。接收端在计算机的5000端口进行接收，并显示接收内容，如果收到字符串bye，则结束监听。

❖ 发送端代码sender.py:

```
import socket
```

```
import sys
```

ipv4

UDP

```
s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

#假设 172.31.74.70 是接收端机器的IP地址

```
s.sendto(sys.argv[1].encode() , (" 172.31.74.70 " , 5000))
```

```
s.close()
```

IP地址

端口

UDP协议编程

❖ 接收端代码receiver.py:

```
import socket
s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
#绑定端口和端口号, 空字符串表示本机任何可用IP地址
s.bind(('', 5000))
while True:
    data, addr = s.recvfrom(1024)
    #显示接收到的内容
    data = data.decode()
    print('received message:{0} from PORT {1[1]} on {1[0]}'.format(data, addr))
    if data.lower() == 'bye':
        break
s.close( )
```

UDP协议编程

The image shows a side-by-side comparison of two Python scripts and their execution. On the left, the 'sender.py' script in Sublime Text defines a UDP socket, binds it to the IP '172.31.74.70' and port '5000', and sends the message 'Hello World!'. Below it, the Anaconda Prompt shows the command 'python sender.py "Hello World!"' being executed. On the right, the 'receiver.py' script in Sublime Text defines a UDP socket, binds it to port '5000' on any IP, and enters a loop to receive data. It prints the received message and breaks the loop if the message is 'bye'. Below it, the Anaconda Prompt shows the command 'python receiver.py' being executed, which results in the output 'received message:Hello World! from PORT 64602 on 172.31.74.70'.

```
C:\Users\Administrator\Desktop\sender.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

sender.py
1 import socket
2 import sys
3
4 s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
5 s.sendto(sys.argv[1].encode(), ("172.31.74.70", 5000))
6 s.close()
7

Anaconda Prompt (anaconda)
(base) C:\Users\Administrator\Desktop>python sender.py "Hello World!"
(base) C:\Users\Administrator\Desktop>
```

```
C:\Users\Administrator\Desktop\receiver.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

receiver.py
1 import socket
2
3 s=socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
4 #绑定端口和端口号, 空字符串表示本机任何可用IP地址
5 s.bind('', 5000)
6 while True:
7     data, addr=s.recvfrom(1024)
8     #显示接收到的内容
9     print('received message:{0} from PORT {1} on {2}'.format(data, addr[1], addr[0]))
10
11     if data.decode().lower() == 'bye':
12         break
13 s.close()
14

选择Anaconda Prompt (anaconda) - python receiver.py
(base) C:\Users\Administrator\Desktop>python receiver.py
received message:Hello World! from PORT 64602 on 172.31.74.70
```

- 先启动接收端的程序进入监听状态;
- 再启动发送端程序;
- 接收端显示收到的内容, 以及发送端程序所在计算机的IP和占用的端口号。

UDP协议编程



```
Anaconda Prompt (anaconda)
(base) C:\Users\Administrator\Desktop>python sender.py "Hello World!"
(base) C:\Users\Administrator\Desktop>python sender.py "bye"
(base) C:\Users\Administrator\Desktop>python sender.py "Hello World!"
(base) C:\Users\Administrator\Desktop>_

Anaconda Prompt (anaconda)
(base) C:\Users\Administrator\Desktop>python receiver.py
received message:Hello World! from PORT 52783 on 172.31.74.70
received message:bye from PORT 52784 on 172.31.74.70
(base) C:\Users\Administrator\Desktop>
```

- 当发送端发送字符串bye后，接收端程序结束；
- 再次运行发送端程序时，接收端没有反应，**发送端并不报错！**
- UDP协议的特点：尽最大努力传输（best-effort），并不保证好的服务质量（能不能到达就不知道了）。
- 虽然用UDP传输数据不可靠，但它的优点是和TCP比，速度快，对于不要求可靠到达的数据，就可以使用UDP协议。

UDP协议编程

- 课后练习：在你们的计算机上运行发送端和接收端
 - 把IP地址改成你们计算机的配置（最好有两台电脑）
- `socket`模块也可以获得本机IP地址 `gethostbyname`

```
ip = socket.gethostbyname(socket.gethostname())  
print('IP:', ip)  
IP: 172.31.74.70
```

```
ip = socket.gethostbyname("www.baidu.com")  
print('IP:', ip)  
IP: 182.61.200.6  
ip = socket.gethostbyname("www.szu.edu.cn")  
print('IP:', ip)  
IP: 210.39.12.247
```

TCP协议编程

- TCP协议适用于对效率要求相对较低而准确性要求很高的场合，例如文件传输、电子邮件等等，需要建立连接、数据传输、断开连接三个步骤。
- socket模块常用于TCP编程的方法有：
 - ✓ `connect(address)`: 连接远程计算机
 - ✓ `send(bytes)`: 发送数据
 - ✓ `recv(bufsize)`: 接收数据
 - ✓ `bind(address)`: 绑定地址
 - ✓ `listen(backlog)`: 开始监听，等待客户端连接
 - ✓ `accept()`: 响应客户端的请求

TCP协议编程

■ TCP类似于打电话

- 1) 客户端（要打电话的人），要知道服务器的ip地址（电话号码），端口（电话分机）。
- 2) 服务器须早于客户端启动，并在指定ip地址和端口上执行监听；如端口被占用，服务器则无法正常启动（监听类似等待被拨打的状态、端口占用类似忙线）。
- 3) 客户端在申请发送数据时，服务器端必须有足够的时间响应才能进行正常通信（类似电话响，却无人接听）。通常情况下，服务器都需要具备同时处理多个客户端请求的能力。
- 4) 使用Socket协议进行通信的双方必须使用相同的通信协议。通信过程中，双方还必须采用相同的字符编码，按照约定的方式进行通信（类似打电话时双方必须语言相同，才能进行信息交流）
- 5) 通信时，物理网络必须保持通畅，否则通信将会中断（类似连接正常）。
- 6) 通信结束时，客户端和服务端都可以中断连接（类似任何一方都可以挂电话）。

TCP协议编程

- 例子：TCP通信程序。模拟机器人聊天软件原理，在服务端提前建立好字典，然后根据接收到的内容自动回复。

```
#服务端代码server.py
```

```
import socket
```

```
#提前建立好字典:问答对
```

```
words = {'how are you?': 'Fine, thank you.',  
         'how old are you?': '18',  
         'what is your name?': 'Handsome',  
         'where do you work?': 'Shenzhen',  
         'bye': 'Bye'}
```

TCP协议编程

#服务端代码server.py

```
HOST = ''
PORT = 50007
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#绑定socket
s.bind((HOST, PORT))
#开始监听
s.listen(1)
print('Listening at port:', PORT)

conn, addr = s.accept() #响应客户端的请求
print('Connected by', addr)
while True:
    data = conn.recv(1024)
    data = data.decode()
    if not data:
        break
    print('Received message:', data)
    conn.sendall(words.get(data, 'Nothing').encode())
conn.close()
s.close()
```

TCP协议编程

客户端代码client.py:

```
import socket
HOST = '127.0.0.1'           #服务端主机IP地址
PORT = 50007                 #服务端主机端口号
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
                                ipv4 TCP
try:
    s.connect((HOST, PORT))    #尝试连接
except Exception as e:
    print('Server not found or not open')
    sys.exit()
```

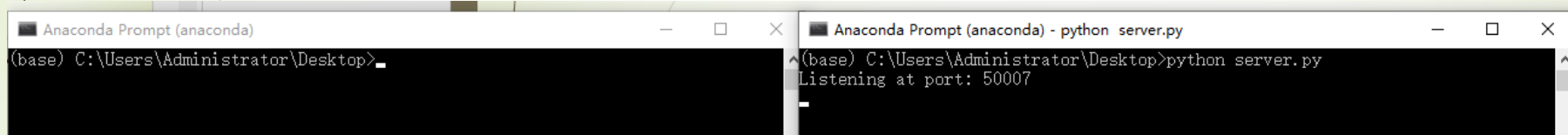
TCP协议编程

```
while True:
    c = input('Input the content you want to send:')
    s.sendall(c.encode())           #发送数据
    data = s.recv(1024)             #从服务端接收数据
    data = data.decode()
    print('Received:', data)

    if c.lower() == 'bye':
        break

s.close()                           #关闭连接
```

TCP协议编程

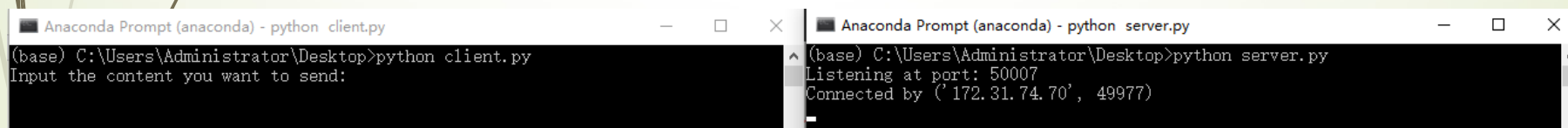


The image shows two side-by-side Anaconda Prompt windows. The left window is titled 'Anaconda Prompt (anaconda)' and shows the command prompt at '(base) C:\Users\Administrator\Desktop>'. The right window is titled 'Anaconda Prompt (anaconda) - python server.py' and shows the command 'python server.py' being executed, with the output 'Listening at port: 50007'.

```
Anaconda Prompt (anaconda)
(base) C:\Users\Administrator\Desktop>

Anaconda Prompt (anaconda) - python server.py
(base) C:\Users\Administrator\Desktop>python server.py
Listening at port: 50007
```

启动运行服务端程序，服务端开始监听



The image shows two side-by-side Anaconda Prompt windows. The left window is titled 'Anaconda Prompt (anaconda) - python client.py' and shows the command 'python client.py' being executed, with the output 'Input the content you want to send:'. The right window is titled 'Anaconda Prompt (anaconda) - python server.py' and shows the command 'python server.py' being executed, with the output 'Listening at port: 50007' and 'Connected by ('172.31.74.70', 49977)'.

```
Anaconda Prompt (anaconda) - python client.py
(base) C:\Users\Administrator\Desktop>python client.py
Input the content you want to send:

Anaconda Prompt (anaconda) - python server.py
(base) C:\Users\Administrator\Desktop>python server.py
Listening at port: 50007
Connected by ('172.31.74.70', 49977)
```

启动运行客户端程序，服务端提示连接已建立

TCP协议编程

```
Anaconda Prompt (anaconda)
(base) C:\Users\Administrator\Desktop>python client.py
Input the content you want to send:what's your name?
Received: Nothing
Input the content you want to send:what is your name?
Received: Handsome
Input the content you want to send:how are you?
Received: Fine, thank you.
Input the content you want to send:where do you work?
Received: Shenzhen
Input the content you want to send:how old are you?
Received: 18
Input the content you want to send:bye
Received: Bye

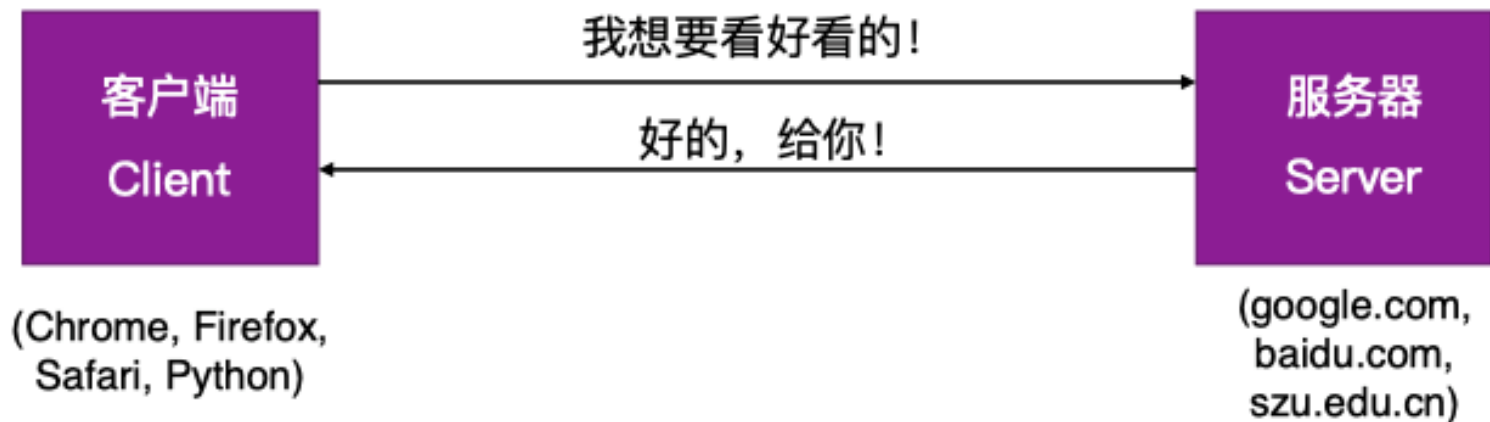
Anaconda Prompt (anaconda)
(base) C:\Users\Administrator\Desktop>python server.py
Listening at port: 50007
Connected by ('172.31.74.70', 49977)
Received message: what's your name?
Received message: what is your name?
Received message: how are you?
Received message: where do you work?
Received message: how old are you?
Received message: bye
(base) C:\Users\Administrator\Desktop>
```

- 在客户端输入要发送的信息后，服务端会根据提前建立的字典来自动回复。
- 服务端每次都在**固定的端口**进行监听。
- 如果服务端程序没有运行，那么客户端就无法建立连接，当然也无法发送任何信息，这正是TCP协议区别于UDP协议的地方。

Python与Web

- Python用于访问网页内容的库
 - ✓ 内置的有urllib模块，但缺少很多实用的高级功能
 - ✓ 更好的方案是使用requests第三方库，处理URL资源特别方便
- requests是用Python语言编写，基于urllib
- 实用的HTTP客户端库，编写爬虫和测试服务器响应数据时经常会用到
- 可以说，requests满足如今网络的需求
- 在命令行下通过pip安装：**pip install requests**

网站工作原理（非常简略版）



请求协议

我想要看好看的!

GET

请求指定资源 <https://www.szu.edu.cn>

POST

将要处理的数据（例如，从表单）提交给服务器。

<https://xxxxx/login>

数据包内容: {"user": "admin", "password": "123456"}

网站工作原理（非常简略）

确保你知道网站将返回什么响应类型！

响应内容类型：

← 好的，给你！

HTML

大多数网页，专为网页浏览器设计

JSON

使用可读文本传输数据对象的数据交换格式。
具有属性-值对和数组的对象。

值可以是：string、number、bool、null。

```
{  
  "latitude": 42.434719,  
  "longitude": -83.985001,  
  "weather": {  
    "temperature": 46,  
    "pressure": 12,  
    "description": "frozen"  
  }  
}
```

JSON字符串用双引号分隔

JSON对象键是字符串

JSON in Python

- JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式，易于人阅读和编写。**独立于编程语言的文本格式来存储和表示数据。**
- JSON数据都可以解释为Python对象
- 使用标准库中的json包。

JSON	Python
Object	Dictionary
Array	List
String	String
Number	Number
Bool	Bool
Null	None

##编码为 JSON 格式数据

```
>>> import json
```

```
>>> data = [ { 'a' : 1, 'b' : 2, 'c' : 3, 'd' : 4, 'e' : 5 } ]
```

```
>>> a = json.dumps(data)
```

```
>>> print(a)
```

```
[{"a": 1, "b": 2, "c": 3, "d": 4, "e": 5}]
```

JSON in Python

#使用参数让 JSON 数据格式化输出

```
>>> data = [ { 'a' : 1, 'b' : 2, 'c' : 3, 'd' : 4, 'e' : 5 } ]  
>>> print(json.dumps(data,indent=4,separators=(',',':')) )
```

```
[  
    {  
        "a":1,  
        "b":2,  
        "c":3,  
        "d":4,  
        "e":5  
    }  
]
```

JSON in Python

将Python数据转换为JSON, 返回字符串

```
json.dumps(obj)
```

将Python数据转换为JSON, 存储在文件中

```
json.dump(obj, f)
```

从文件加载JSON

```
file_data = json.load(f)
```

从字符串加载JSON

```
>>> s_data = json.loads('{"Michael": "Favourite Canadian"}')
```

```
>>> s_data
```

```
{'Michael': 'Favourite Canadian'}
```

```
>>> jsonData = '{"a":1,"b":2,"c":3,"d":4,"e":5}';
```

```
>>> text = json.loads(jsonData)
```

```
>>> text
```

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5}
```

GET请求

```
import requests
```

```
r = requests.get('http://www.baidu.com')
```

```
print(r.status_code)
```

```
print(r.text) #网页源码
```

200（成功）服务器已成功处理了请求。
通常，这表示服务器提供了请求的网页。

```
200
<!DOCTYPE html>
<!--STATUS OK--><html> <head><meta http-equiv=content-type content=text/html;charset=utf-8><me
ta http-equiv=X-UA-Compatible content=IE=Edge><meta content=always name=referrer><link rel=sty
lesheet type=text/css href=http://s1.bdstatic.com/r/www/cache/bdorz/baidu.min.css><title>百度
</title></head> <body link=#0000cc> <div id=wrapper> <div id=head> <div
class=head_wrapper> <div class=s_form> <div class=s_form_wrapper> <div id=lg> <img hidefocus=t
rue src=//www.baidu.com/img/bd_logo1.png width=270 height=129> </div> <form id=form name=f act
ion=//www.baidu.com/s class=fm> <input type=hidden name=bdorz_come value=1> <input type=hidden
name=ie value=utf-8> <input type=hidden name=f value=8> <input type=hidden name=rsv_bp value=
1> <input type=hidden name=rsv_idx value=1> <input type=hidden name=tn value=baidu><span class
="bg s_ipt_wr"><input id=kw name=wd class=s_ipt value maxlength=255 autocomplete=off autofocus
></span><span class="bg s_btn_wr"><input type=submit id=su value=百度</span></div> </div> <div id=u1> <a href=http://news.baidu.com name=tj_trnews class=
mnav>新闻</a> <a href=http://www.hao123.com name=tj_trhao123 class=mnav>hao123</a> <a href=h
ttp://map.baidu.com name=tj_trmap class=mnav>地图</a> <a href=http://v.baidu.com name=tj_trv
ideo class=mnav>视频</a> <a href=http://tieba.baidu.com name=tj_trtieba class=mnav>贴吧</a>
<noscript> <a href=http://www.baidu.com/bdorz/login.gif?login&tpl=mn&u=http%3A%2F%2F
www.baidu.com%2F%3Fbdorz_come%3D1 name=tj_login class=lb>登录</a> </noscript> <script>docume
nt.write(<a href="http://www.baidu.com/bdorz/login.gif?login&tpl=mn&u="+ encodeURIComponent(w
indow.location.href+ (window.location.search === "" ? "?" : "&")+ "bdorz_come=1")+ "" name="tj
_login" class="lb">登录</a>);</script> <a href=//www.baidu.com/more/ name=tj_briicon class=
bri style="display: block;">百度</a> </div> </div> <div id=ftCon> <div id=ftCon
w> <p id=lh> <a href=http://home.baidu.com>百度</a> <a href=http://ir.baidu.com>About
Baidu</a> </p> <p id=cp>&copy;2017 Baidu<a href=http://www.baidu.com/duty/>百度</a>
<a href=http://jianyi.baidu.com/ class=cp-feedback>意见反馈</a> &nb
sp;<a href=//www.baidu.com/img/gu.gif> </p> </div> </div> </div> </b
ody> </html>
```

属性

```
r.status_code  
r.text  
r.url  
r.headers  
r.encoding
```

#返回连接状态
#网页源码的字符串
#打印输出该 URL
#以字典对象存储服务器响应头
#获取当前的编码

```
>>> print(r.url)  
http://www.baidu.com/
```

```
>>> print(r.headers)  
{'Cache-Control': 'private, no-cache, no-store, proxy-revalidate,  
no-transform', 'Connection': 'keep-alive', 'Content-Encoding':  
'gzip', 'Content-Type': 'text/html', 'Date': 'Wed, 20 May 2020  
06:30:36 GMT', 'Last-Modified': 'Mon, 23 Jan 2017 13:23:55 GMT',  
'Pragma': 'no-cache', 'Server': 'bfe/1.0.8.18', 'Set-Cookie':  
'BDORZ=27315; max-age=86400; domain=.baidu.com; path=/',  
'Transfer-Encoding': 'chunked'}
```

```
>>> print(r.encoding)  
ISO-8859-1
```

属性

```
>>> print(r.headers)
{'Cache-Control': 'private, no-cache, no-store, proxy-revalidate,
no-transform', 'Connection': 'keep-alive', 'Content-Encoding':
'gzip', 'Content-Type': 'text/html', 'Date': 'Wed, 20 May 2020
06:30:36 GMT', 'Last-Modified': 'Mon, 23 Jan 2017 13:23:55 GMT',
'Pragma': 'no-cache', 'Server': 'bfe/1.0.8.18', 'Set-Cookie':
'BDORZ=27315; max-age=86400; domain=.baidu.com; path=/',
'Transfer-Encoding': 'chunked'}
```

```
>>> print(r.headers.get('content-type'))
text/html
```

```
r.json()
#把网页中的json数据转成字典并将其返回。
```

GET请求

另一个例子：简单便于理解

```
import requests
```

```
response = requests.get('http://httpbin.org/get')  
print(response.text)
```

```
(base) C:\Users\Administrator\Desktop>python client_web.py  
{  
  "args": {},  
  "headers": {  
    "Accept": "*/*",  
    "Accept-Encoding": "gzip, deflate",  
    "Host": "httpbin.org",  
    "User-Agent": "python-requests/2.22.0",  
    "X-Amzn-Trace-Id": "Root=1-5ed604f2-0423971088b7ae706f7ba90c"  
  },  
  "origin": "116.7.245.182",  
  "url": "http://httpbin.org/get"  
}
```


GET请求：带参数

- 为 URL 的查询字符串 (query string) 传递某种数据。
- 手工构建 URL，数据会以键/值对的形式置于 URL 中，跟在一个问号的后面，如， www.baidu.com/?key=val。
- requests 允许使用 params 关键字参数，以一个字符串字典来提供这些参数。
- 例子，想传递 key1=value1 和 key2=value2 到 <http://httpbin.org/get>

```
import requests
```

```
payload = {'key1': 'value1', 'key2': 'value2'}  
r = requests.get("http://httpbin.org/get", params=payload)  
print(r.text)
```

GET请求：带参数

```
import requests
```

```
payload = {'key1': 'value1', 'key2': 'value2'}
```

```
r = requests.get("http://httpbin.org/get", params=payload)
```

```
print(r.text)
```

```
(base) C:\Users\Administrator\Desktop>python client_web.py
{
  "args": {
    "key1": "value1",
    "key2": "value2"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.22.0",
    "X-Amzn-Trace-Id": "Root=1-5ed6062c-3f2f1468cdf431372d8bf813"
  },
  "origin": "116.7.245.186",
  "url": "http://httpbin.org/get?key1=value1&key2=value2"
}
```

GET请求：带参数

- 练习

```
payload = {'key1': 'value1', 'key2': ['value2', 'value3']}  
r = requests.get("https://www.baidu.com/", params=payload)  
print(r.url)
```

```
r = requests.get('https://www.douban.com/search',  
params={'city': 'shenzhen', 'area': 'nanshan'})  
print(r.url)
```

GET请求：带参数

- 练习

```
payload = {'key1': 'value1', 'key2': ['value2', 'value3']}  
r = requests.get("https://www.baidu.com/", params=payload)  
print(r.url)
```

`https://www.baidu.com/?key1=value1&key2=value2&key2=value3`

```
r = requests.get('https://www.douban.com/search',  
params={'city': 'shenzhen', 'area': 'nanshan'})  
print(r.url)
```

`https://www.douban.com/search?city=shenzhen&area=nanshan`

解析json

```
import requests
```

```
response = requests.get('http://httpbin.org/get')
```

```
print(response.text)
```

```
print(response.json()) #把网页中的json数据转成字典并将其返回
```

```
print(type(response.json())) #dict
```

```
(base) C:\Users\Administrator\Desktop>python client_web.py
```

```
{  
  "args": {},  
  "headers": {  
    "Accept": "*/*",  
    "Accept-Encoding": "gzip, deflate",  
    "Host": "httpbin.org",  
    "User-Agent": "python-requests/2.22.0",  
    "X-Amzn-Trace-Id": "Root=1-5ed606d9-caa351eac1a6efbceb3b8962"  
  },  
  "origin": "116.7.245.186",  
  "url": "http://httpbin.org/get"  
}
```

```
{'args': {}, 'headers': {'Accept': '*/*', 'Accept-Encoding': 'gzip, deflate', 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.22.0', 'X-Amzn-Trace-Id': 'Root=1-5ed606d9-caa351eac1a6efbceb3b8962'}, 'origin': '116.7.245.186', 'url': 'http://httpbin.org/get'}  
<class 'dict'>
```

使用get获取图片

```
import requests
```

```
response =
```

```
requests.get('https://ss0.bdstatic.com/70cFuHSh_Q1YnxGkpoWK1HF6hhy/it/u=213623723,2520951995&fm=26&gp=0.jpg')
```

```
b = response.content
```

```
with open('view.jpg','wb') as f:
```

```
    f.write(b)
```

保存一个二进制文件



Post请求

要发送POST请求，只需要把get()方法变成post()，然后传入data参数作为POST请求的数据：

```
import requests
data = {'name': 'tom', 'age': '22' }
response = requests.post('http://httpbin.org/post', data=data)
print(response.text)
```

```
(base) C:\Users\Administrator\Desktop>python client_web.py
{
  "args": {},
  "data": "",
  "files": {},
  "form": {
    "age": "22",
    "name": "tom"
  },
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Content-Length": "15",
    "Content-Type": "application/x-www-form-urlencoded",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.22.0",
    "X-Amzn-Trace-Id": "Root=1-5ed60917-827ac70a14a23db3bbe7f450"
  },
  "json": null,
  "origin": "116.7.245.182",
  "url": "http://httpbin.org/post"
}
```

可以看到参数传成功了，然后服务器返回了我们传的数据。

Post请求

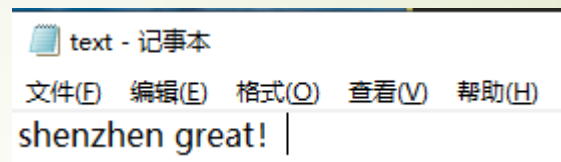
有时候我们需要传送的信息不是表单形式的，**需要我们传JSON格式的数据**过去，所以我们可以用 `json.dumps()` 方法把表单数据序列化。

```
url = 'http://httpbin.org/post'
payload = {'some': 'data'}
r = requests.post(url, data=json.dumps(payload))
print(r.text)
```

```
(base) C:\Users\Administrator\Desktop>python client_web.py
{
  "args": {},
  "data": "{\"some\": \"data\"}",
  "files": {},
  "form": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Content-Length": "16",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.22.0",
    "X-Amzn-Trace-Id": "Root=1-5ed60a18-38706d323cab98de4331f880"
  },
  "json": {
    "some": "data"
  },
  "origin": "116.7.245.186",
  "url": "http://httpbin.org/post"
}
```

可以POST JSON格式的数据

Post请求



如果想要上传文件，那么直接用 **file** 参数即可。

```
import requests
url = 'http://httpbin.org/post'
files = {'file': open('text.txt', 'rb')}
r = requests.post(url, files=files)
print(r.text)
```

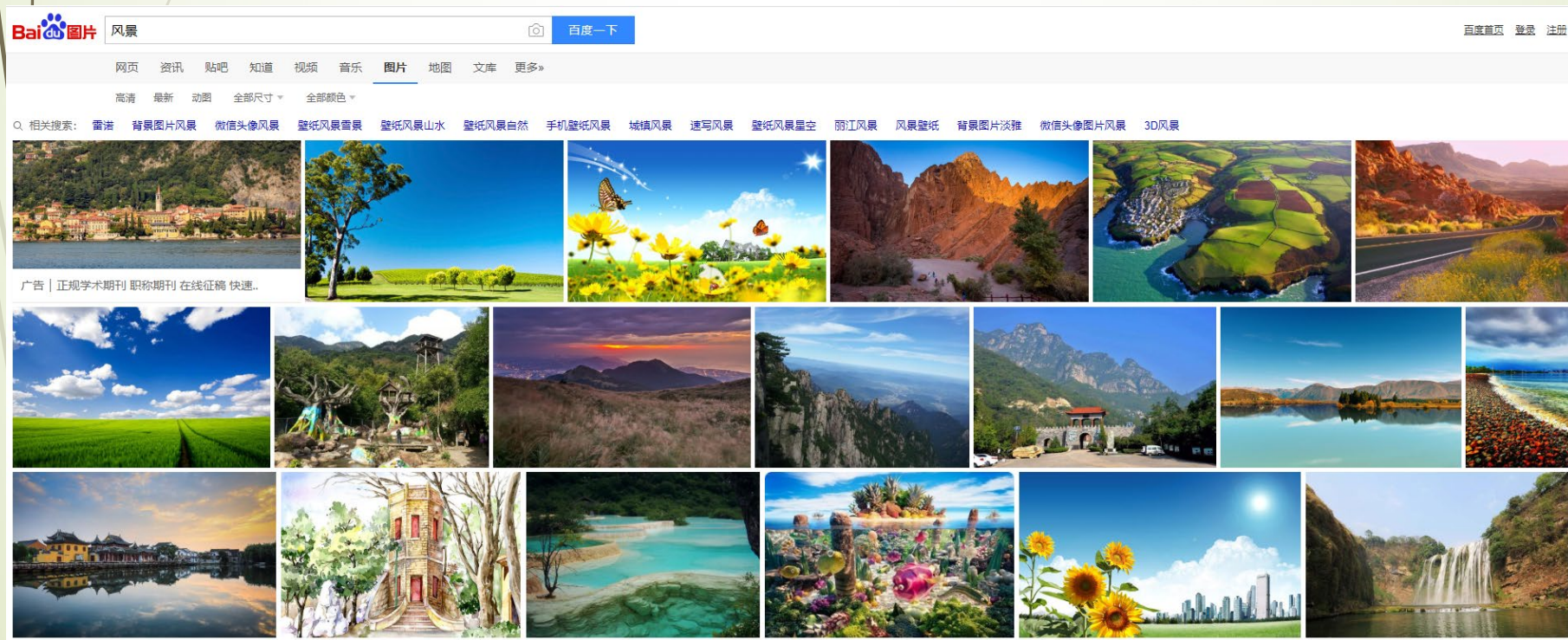
```
(base) C:\Users\Administrator\Desktop>python client_web.py
{
  "args": {},
  "data": "",
  "files": {
    "file": "shenzhen great\u0001"
  },
  "form": {},
  "headers": {
    "Accept": "*/*",
    "Accept-Encoding": "gzip, deflate",
    "Content-Length": "161",
    "Content-Type": "multipart/form-data; boundary=936206b9247cad8da356801bb8f5887a",
    "Host": "httpbin.org",
    "User-Agent": "python-requests/2.22.0",
    "X-Amzn-Trace-Id": "Root=1-5ed60b01-5618c2cbd390aa3bc644f355"
  },
  "json": null,
  "origin": "116.7.245.186",
  "url": "http://httpbin.org/post"
}
```

这样我们便成功
完成了文件的
上传。

简单的图片爬取

比如说，我们想下载一系列的风景图片

百度搜索得到了很多图片，一张一张图片下载？右键→保存？



简单的图片爬取

【有点傻】

比如说，我们想下载一系列的风景图片

百度搜索得到了很多图片，一张一张图片下载？右键→保存？

【聪明人】

第一步：获取图片的url链接：右键检查网页源代码，直接（ctrl+F）搜索objURL

```

    app.setData('jsConfs',{ "normal":{ "margin":5,"padding":0,"pageNumHeight":33,"lineHeight":200,"maxBaseLineHeight":220,"minBaseLineHeight":200,"pageLineNum":3,"pageImgLimit":20,"maxImgWidth":400,"pageMoreNum":10,"leftMenu":0 }, "comic":{ "margin":5,"padding":0,"pageNumHeight":33,"lineHeight":190,"maxBaseLineHeight":210,"minBaseLineHeight":190,"pageImgLimit":20,"imgDigestHeight":400,"maxImgWidth":400,"pageMoreNum":10,"leftMenu":0 }, "ad":{ "margin":5,"padding":0,"pageNumHeight":33,"lineHeight":190,"maxBaseLineHeight":210,"minBaseLineHeight":190,"pageLineNum":3,"pageImgLimit":20,"imgDigestHeight":0,"maxImgWidth":400,"pageMoreNum":10,"leftMenu":0 }, "star":{ "margin":5,"padding":0,"pageNumHeight":33,"lineHeight":260,"maxBaseLineHeight":280,"minBaseLineHeight":260,"pageLineNum":3,"pageImgLimit":20,"imgDigestHeight":0,"maxImgWidth":400,"pageMoreNum":10,"leftMenu":0 }, "avatar_brand":{ "margin":5,"padding":0,"pageNumHeight":33,"lineHeight":300,"maxBaseLineHeight":320,"minBaseLineHeight":280,"pageLineNum":3,"pageImgLimit":20,"imgDigestHeight":500,"pageMoreNum":10,"leftMenu":0 }, "avatar_star":{ "margin":5,"padding":0,"pageNumHeight":33,"lineHeight":260,"maxBaseLineHeight":280,"minBaseLineHeight":260,"pageLineNum":3,"pageImgLimit":20,"imgDigestHeight":400,"pageMoreNum":10,"leftMenu":0 }, "star_head":{ "margin":5,"padding":0,"pageNumHeight":33,"lineHeight":260,"maxBaseLineHeight":280,"minBaseLineHeight":260,"pageLineNum":3,"pageImgLimit":20,"imgDigestHeight":0,"maxImgWidth":400,"pageMoreNum":10,"leftMenu":190 }, "avatar_wallpaper":{ "margin":0,"padding":0,"pageNumHeight":33,"lineHeight":180,"maxBaseLineHeight":200,"minBaseLineHeight":160,"pageLineNum":3,"pageImgLimit":20,"imgDigestHeight":0,"maxImgWidth":400,"leftMenu":190 }, "single":{ "margin":5,"padding":0,"pageNumHeight":33,"lineHeight":200,"maxBaseLineHeight":220,"minBaseLineHeight":200,"pageLineNum":1,"pageImgLimit":20,"imgDigestHeight":400,"pageMoreNum":10,"leftMenu":0 }, "personalized":{ "margin":5,"padding":0,"pageNumHeight":33,"lineHeight":200,"maxBaseLineHeight":220,"minBaseLineHeight":200,"pageLineNum":2,"pageImgLimit":20,"imgDigestHeight":0,"maxImgWidth":400,"pageMoreNum":10,"leftMenu":0 } }}
    app.setData('imgData',{ "queryEnc":"%E9KA3X%E6%99%A7","displayName":1118658,"bdIsClustered":1,"listNum":1939,"bfFatDispNum":1118658,"bdSearchTime":"","isNeedAsyncRequest":0,"data":{"thumbUrl":"http://img1.imgtn.bdimg.com/it/u=1752460159,226752426&fm=26&gp=0.jpg","replaceUrl":["ObJURU":"http://vq4.imgtn.bdimg.com/vt/u=1752460159,226752426&fm=214&gp=0.jpg","ObJURU":"http://vq4.imgtn.bdimg.com/vt/u=1752460159,226752426&fm=214&gp=0.jpg","FromURL":"http://www.gardenscn.cn/u5a62u51b2u3047u9352u2540u7c39u690/u5e2u6ad9.html"},"ObJURU":"http://vvp.enderdesk.com/vedpic_source/vd8/55/5fV/d8555f8a30fa609f1673f3395baebff0.jpg","ObJURU":"http://vvp.enderdesk.com/vedpic_source/vd8/55/5fV/d8555f8a30fa609f1673f3395baebff0.jpg","FromURL":"http://www.enderdesk.com/download/54132V","FromURL":"http://www.enderdesk.com/download/54132V","adType":"","middleURU":"","http://img1.imgtn.bdimg.com/it/u=1752460159,226752426&fm=26&gp=0.jpg","largeImageURL":"","hasLarge":0,"hoverURL":"http://img1.imgtn.bdimg.com/it/u=1752460159,226752426&fm=26&gp=0.jpg","pageNum":0,"ObJURU":"http://tupian.enderdesk.com/2012/1222/ldh/1/adlyK20X2810X29,"fromURL":"ippr_zcQs4zdH3FAzdh3Fooo_ze3Bjcpj61fh_ze3Bv54AzhD3FetJaZdH3FP9AZdH3Fcbln_l_ze3Bip4s","strategyAssessment":"3352832306_34_0","filesize":"","bdSrcType":"","di":"88440","is":"0.0","partnerId":"","bdSetImgNum":0,"spn":0,"bdIngnewsDate":"","fromPageTitle":"<strong>风景</strong>迷人的澳大利亚","bdSourceName":"","bdFromPageTitlePrefix":"","isAspDianjing":0,"token":"","imgType":"","adid":"","pi":""}

```

简单的图片爬取

【聪明人】

第二步：把图片链接保存到本地： 如何寻找objURL??

字符串的处理

第三步：使用requests下载图片

（如果第二步的返回结果是个列表，那么这里只是一个简单的循环）

需要使用的库：

- ✓os（保存图片，如路径设置等）
- ✓requests（处理网页信息，下载图片，如使用get请求）