day01-爬虫基础

1. 爬虫的概念和作用

1.1 爬虫的概念

爬虫就是模拟客户端（浏览器）发送请求 获取响应，通过一定的规则自动的从互联网上获取数据的方式就是爬虫。

1.2 爬虫的作用

- 抓取数据
- 12306抢票软件
- 点赞 投票
- 自动化测试
- web安全 扫描网址的漏洞。
- 原则上只要是浏览器能做的事情，爬虫都可以去做。

1.3 爬虫的分类

- 通用爬虫
    - 搜索引擎（百度 谷歌 360） 抓取的时候 是没有上限的
- 聚焦爬虫
    - 针对一个网址或者是某一类网址编写的爬虫就是聚集爬虫。 （当前阶段学习的爬虫）

1.4 爬虫的流程-【重要】

- 准备url地址（相当于在浏览器地址栏中输入了一个网址）
- 发送请求 获取数据（输入完网址之后 回车）
- 提取数据（从当前页面中将想要获取的数据拿到）
- 保存数据（将拿到的数据保存电脑）

2. HTTP

2.1 http和https的概念

协议：双方约定好的规定好的规则。

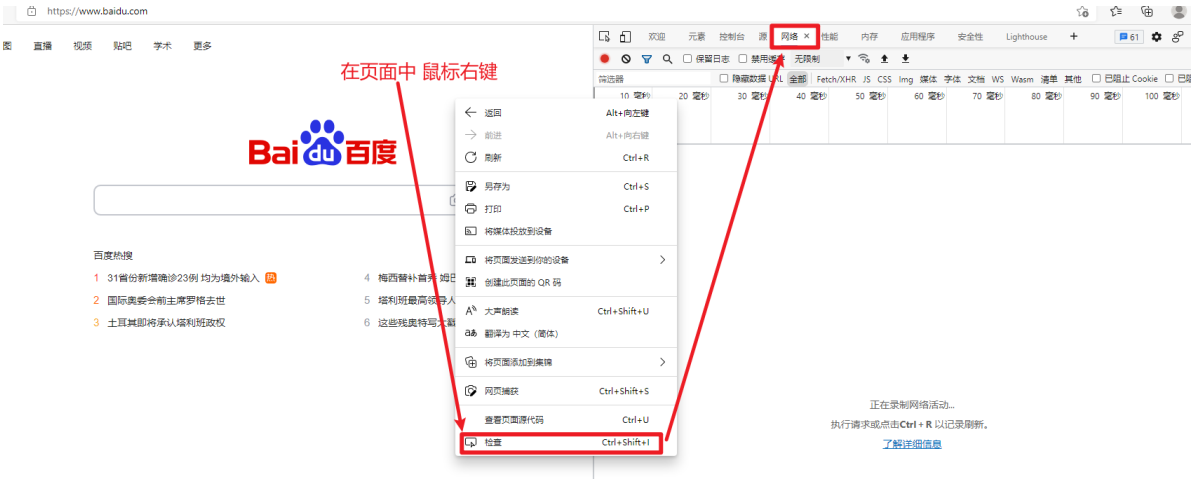
http协议： 超文本传输协议，规定了在互联网中，客户端和服务器端进行通信的规则。 默认端口号 80

- 传输数据的时候 是不会对数据进行加密

https协议： http + ssl（安全套接字层）客户端和服务器端进行通信的规则。默认端口号： 443

- ssl 负责对我们传输的数据进行加密，相对于http来说https安全
- https要比http效率低。

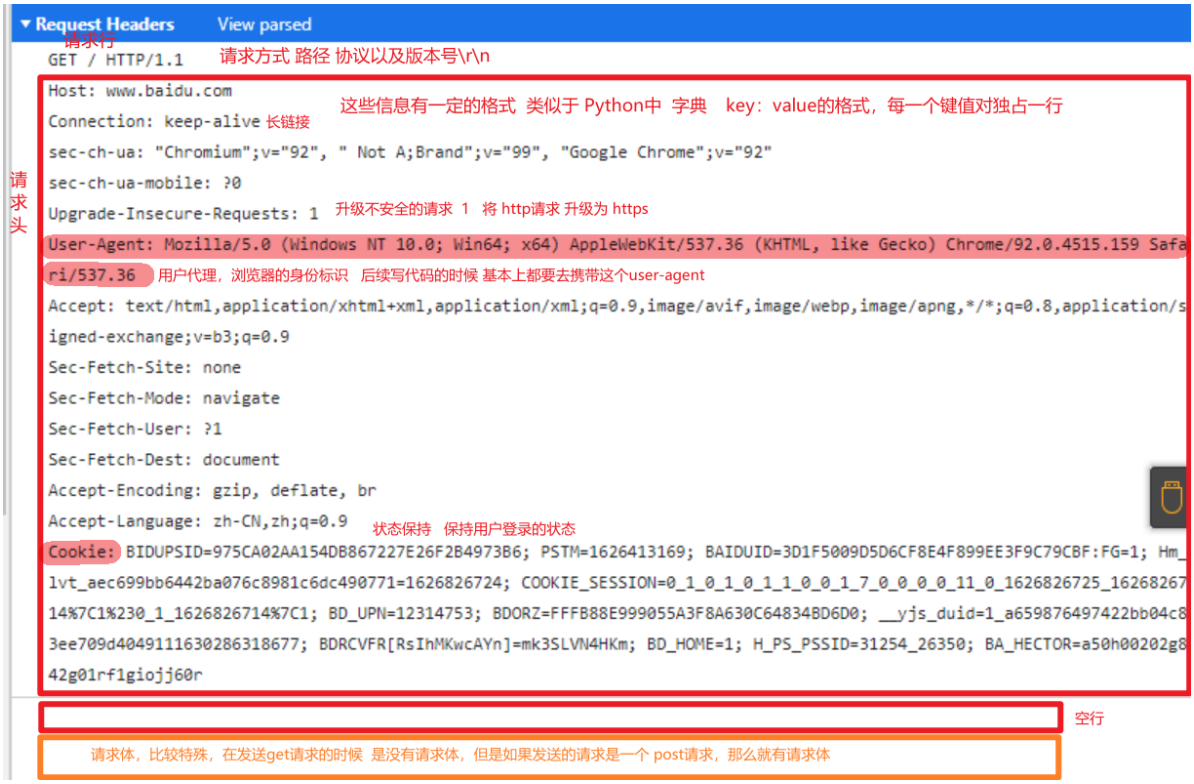
2.2 请求报文和响应报文的格式



请求报文

指的就是我们在浏览器地址栏中输入网址之后，点击回车，向服务器发送请求的时候携带的数据。

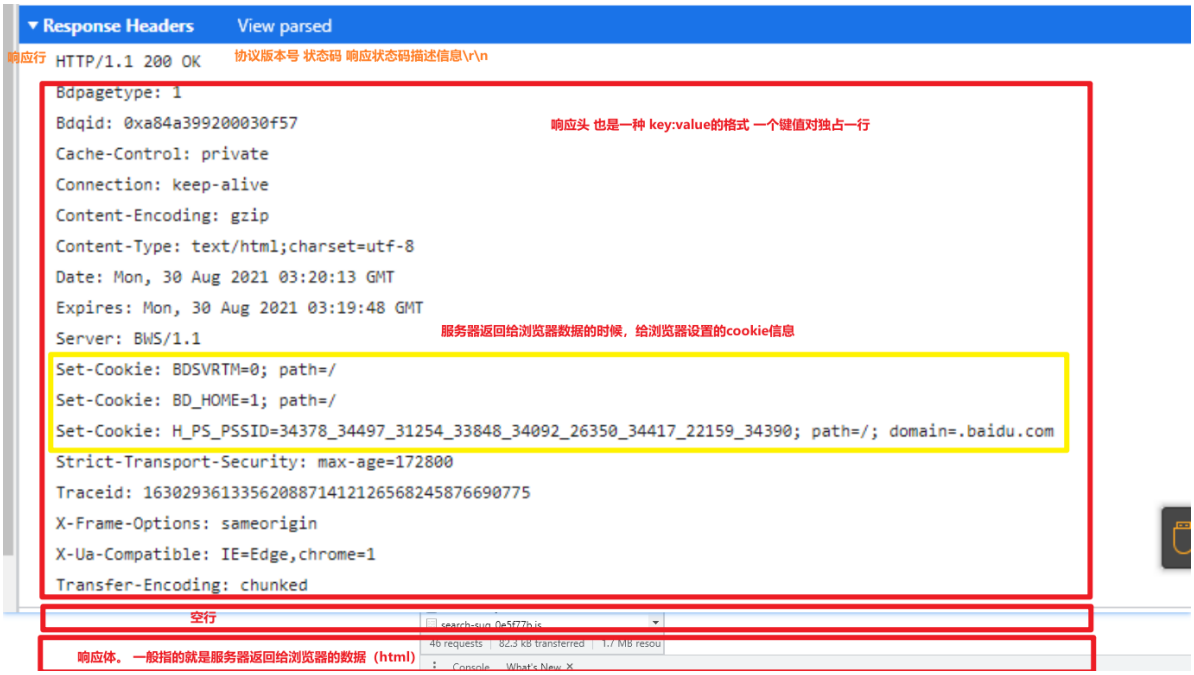
组成： 请求行 请求头 空行 请求体

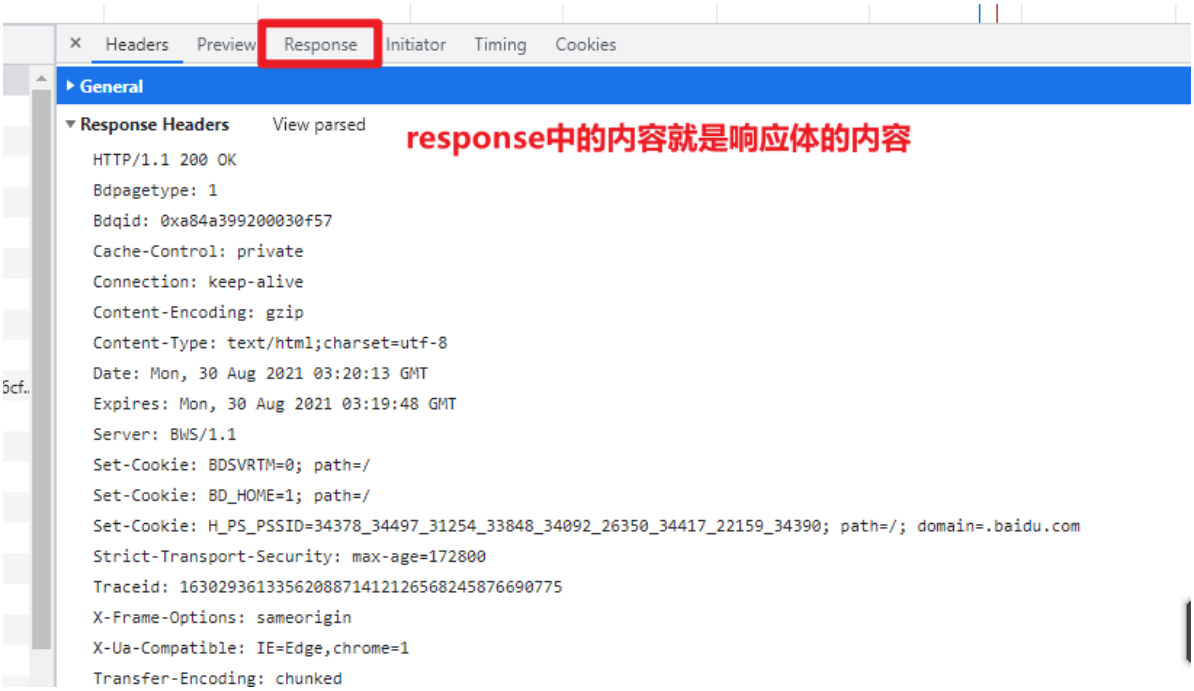


响应报文

指的就是服务器返回给浏览器数据的时候，遵循的数据的格式

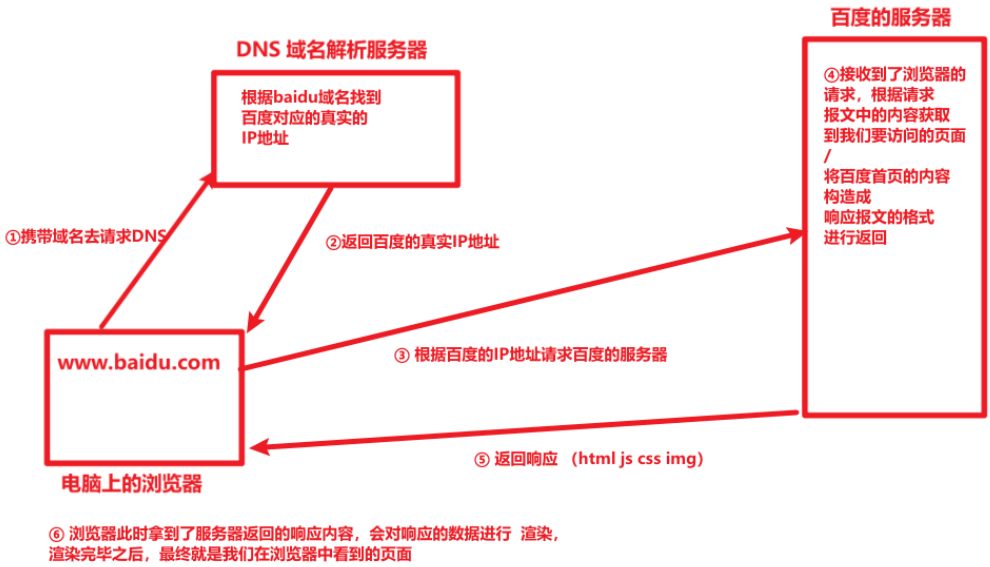
格式： 响应行 响应头 空行 响应体





2.3 浏览器请求的过程

当我们在浏览器地址栏中输入 www.baidu.com 并且回车之后，都发生了什么?



2.4 url地址的完整格式

url地址：统一资源定位符

```
协议://域名:端口号/路径/?key=value#锚点

协议： http https （ftp）
域名： baidu.com  itcast.cn
端口号： 省略不写， 默认端口 http80  https443
路径：  要访问的资源路径  http://www.baidu.com/index.html
?key=value: 请求参数或者叫做查询字符串参数（query_string） 客户端发送请求的时候携带的数据。如果有多条数据， 每个数据之间是有 & 符号隔开，
    例如： ?a=1&b=2&c=3
#锚点： 对前面的地址，或者是访问到的页面资源没有任何影响，在页面内部进行跳转 定位。
```

2.5 响应状态码

- 2 表示成功
  - 200
- 3 表示的是重定向
  - 302
- 4 资源请求不到
  - 404 服务器资源找不到
  - 403 forbidden 访问被拒绝，没有访问的权限。
- 5 服务器错误
  - 500 一般就是服务器错误（数据库问题、服务器宕机）
  - 503 服务器在一定的时间内容，接收到的请求过多，暂时无法处理这些请求的时候 就会返回503

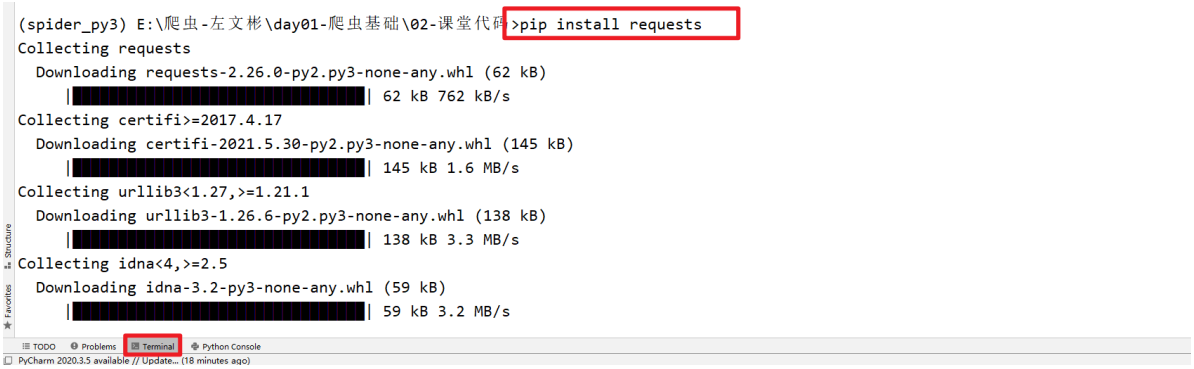
3. requests模块的使用

3.1 介绍与安装

requests在Python中可以去实现 发送请求获取响应。

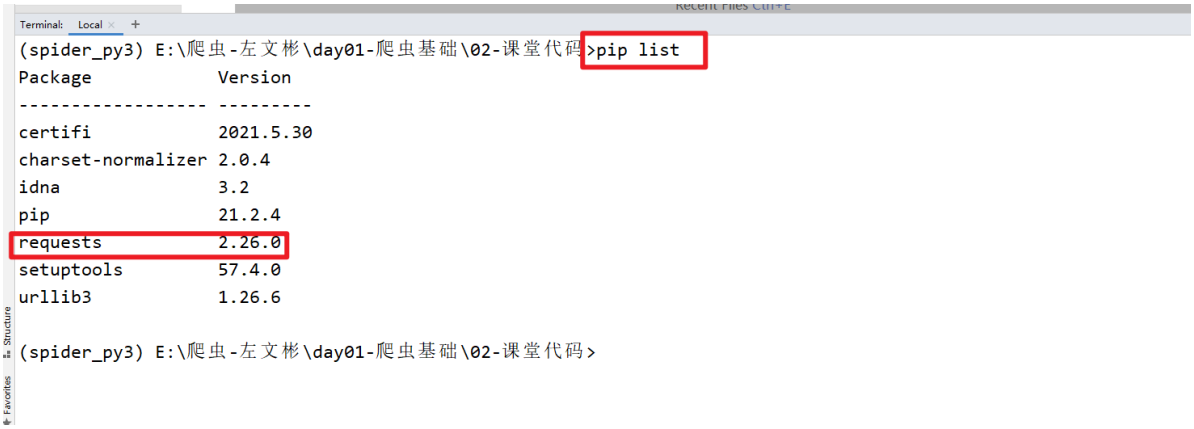
requests模块第三方模块，想要去使用该模块，要去安装下载。

```
pip install requests
```



在当前使用的Python环境中将requests模块安装上了。

可以去查看当前环境中安装的模块信息，以及模块具体的信息

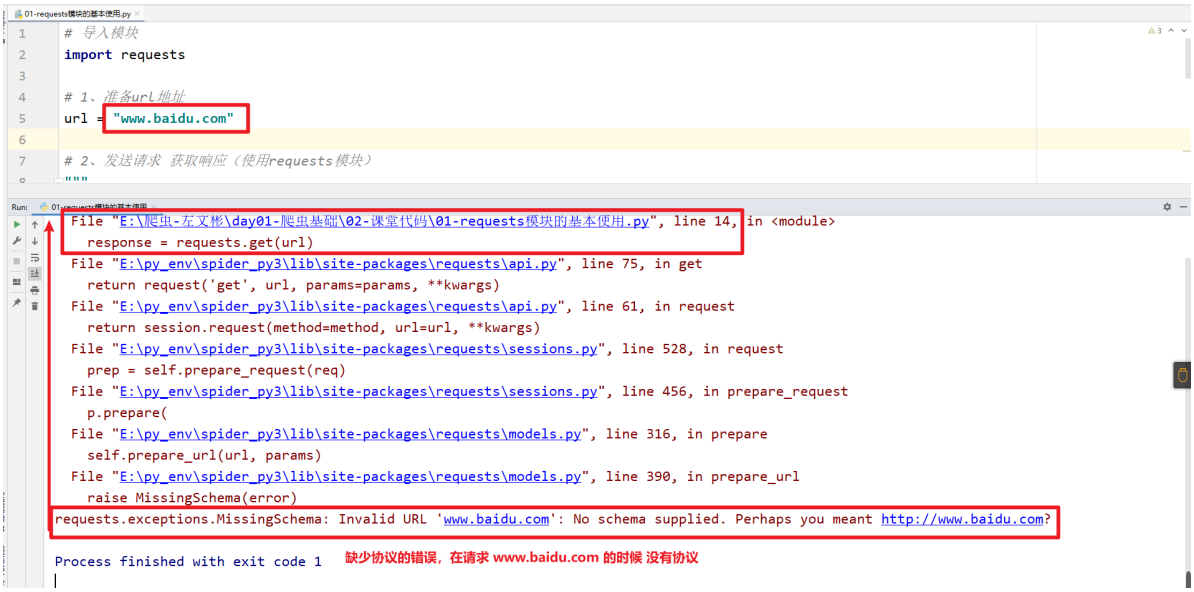


官方文档的连接： https://docs.python-requests.org/zh_CN/latest/

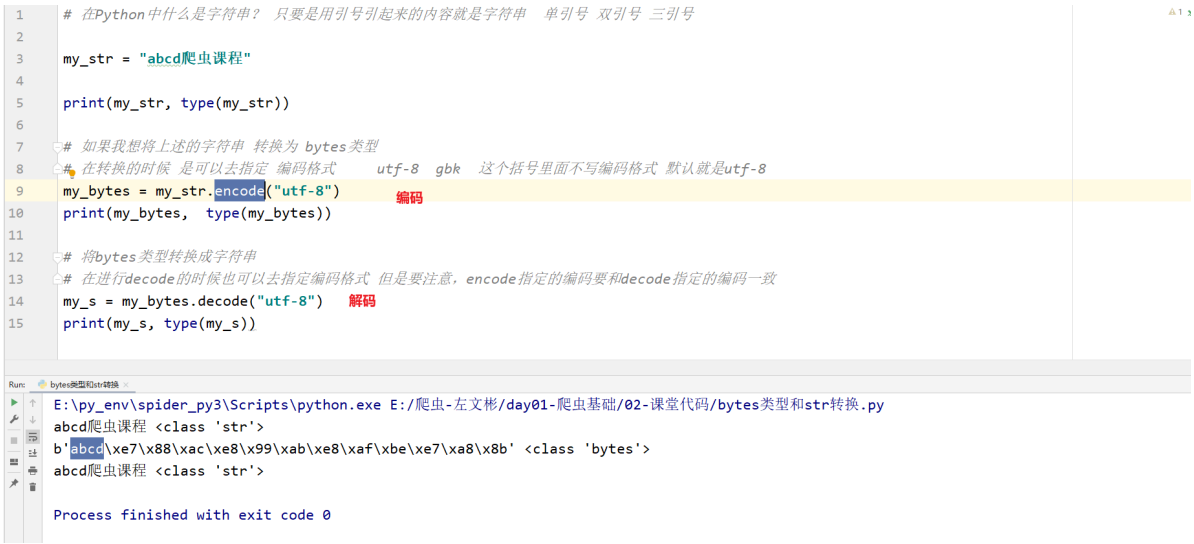
3.2 requests模块基本使用

```python
# 导入模块
import requests

# 1、准备url地址
url = "http://www.baidu.com"

# 2、发送请求 获取响应（使用requests模块）
"""
requests.get(url) 接收一个参数就是要发送请求的url地址
调用完该方法之后，有一个返回值，
这个返回值 是一个 response 响应对象
response = requests.get(url)
"""
response = requests.get(url)
# 可以去打印一下 获取到的response对象
# <Response [200]>   打印一个变量的时候，如果输出的格式  是以 <> 这个内容
# 表示这个变量是一个对象  类型 就是  Response
# 200 指的就是响应状态码
print(response)

# 3、提取数据（省略）
# 4、保存数据（省略）
```

注意点：在准备url地址的时候，是必须书写url地址的完整格式，要包含协议

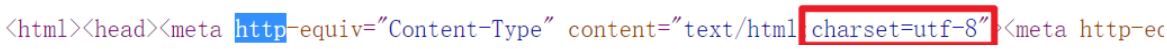


3.3 获取响应内容



如何去查看一个网页的编码格式

查看一个网页的编码格式 在 head标签中找 meta meta中找 charset




```python
import requests

# 1、准备url地址
url = "http://www.baidu.com"

# 2. 使用requests模块去发送请求 获取到响应对象 response
response = requests.get(url)

print(response)

# 查看一下 响应的内容（数据）
# response.text 属性，返回的是响应的字符串内容， html字符串
# response.text 属性 获取到的响应的字符串类型的数据，但是呢数据在互联网上进行传输的时候，使用的是 二进制 bytes 数据类型进行传输的
# 原因就是我们在调用response.text属性的时候，在源码中自动的将 bytes类型 --- str
# 但是在进行转换的时候，decode() 方法中指定的编码格式 是 程序推测出来的 进行解码。
# 互联网中传输的时候 可能是用的 utf-8 编码之后的bytes类型 但是在解码的时候 使用 gbk 或者是其他的，
# 如何解决
# 在response中有一个属性  response.encoding 就是用于获取 在程序中进行解码的编码格式
# 经过查看 网页源码 我们发现 百度使用的是  utf-8编码，相当于 将html字符串转换为byte类型的时候， encode("utf-8")
# 但是在程序中 将 bytes类型转换为str的时候  decode("ISO-8859-1")
print(response.encoding)
# 在使用 response.text 之前 修改 ISO-8859-1 为 utf-8
response.encoding = "utf-8"
print(response.text)
# 打印数据的类型  使用 type(数据)
print(response.encoding)
print(type(response.text))
```

response.content

```python
import requests

# 1、准备url地址
url = "http://www.baidu.com"

# 2. 使用requests模块去发送请求 获取到响应对象 response
response = requests.get(url)

print(response)

# 查看一下 响应的内容（数据）
# response.text 属性，返回的是响应的字符串内容， html字符串
# response.text 属性 获取到的响应的字符串类型的数据，但是呢数据在互联网上进行传输的时候，使用的是 二进制 bytes 数据类型进行传输的
# 原因就是我们在调用response.text属性的时候，在源码中自动的将 bytes类型 --- str
# 但是在进行转换的时候，decode() 方法中指定的编码格式 是 程序推测出来的 进行解码。
# 互联网中传输的时候 可能是用的 utf-8 编码之后的bytes类型 但是在解码的时候 使用 gbk 或者是其他的，
# 如何解决
# 在response中有一个属性  response.encoding 就是用于获取 在程序中进行解码的编码格式
```

```python
# 经过查看 网页源码 我们发现 百度使用的是  utf-8编码，相当于 将html字符串转换为byte类型的时候， encode("utf-8")
# 但是在程序中 将 bytes类型转换为str的时候  decode("ISO-8859-1")
print(response.encoding)
# 在使用 response.text 之前 修改 ISO-8859-1 为 utf-8
response.encoding = "utf-8"
print(response.text)
# 打印数据的类型  使用 type(数据)
print(response.encoding)
print(type(response.text))

print('-' * 66)

# 第二种方式 获取响应的内容
# response.content  获取响应的 bytes类型数据， 最原始的没有被转换为字符串的数据
print(response.content)
print(type(response.content))

# 如果我想获取响应的字符串   decode中什么都不写 默认的就是 utf-8
print(response.content.decode())
```

总结

- response.text: 获取的就是响应的html字符串类型的数据，但是获取到的内容乱码，想要解决乱码问题
    - 先去修改 解码的时候 使用的编码格式 response.encoding = "指定的编码格式"
    - 再去使用response.text
- response.content: 获取的就是响应的bytes类型的数据，如果想要获取字符串类型的额数据
    - response.content.decode()
    - decode() 方法什么都不写 表示使用的编码格式默认就是 utf-8、
- 后续再写爬虫的时候，一般都会去使用 `response.content.decode()` 这种方式

3.4 响应对象的属性和方法

- response.text: 获取到响应的html字符串类型的数据
- response.content： 获取到响应的 bytes类型的数据，如果要转换为字符串类型 可以使用 response.content.decode()
- response.url : 获取当前响应的url地址。
- response.headers: 获取当前响应报文中的响应头信息
- response.request.headers: 获取当前响应对应的请求报文中的请求头信息
- response.cookies: 获取服务器设置给客户端的cookie信息。
- response.request._cookies: 获取当前在发送请求的时候请求报文中携带的cookie信息。
- response.status_code: 获取响应状态码
- response.json(): 当响应数据是 json 字符串的时候，才可以使用该方法，该方法可以将json字符串自动的转换为Python中的字典或者是列表。
    - 数据一种格式，JavaScript对象的字符串表现形式。

```python
import requests

```

```python
# 1. 准备要请求的url地址
url = "http://www.baidu.com"

# 2. 发送请求 获取响应
response = requests.get(url)
print(response)

# - response.status_code: 获取响应状态码
print(f"获取当前响应的响应状态码： {response.status_code}")
# - response.url : 获取当前响应的url地址。
print(f"当前响应的url地址： {response.url}")
# - response.headers: 获取当前响应报文中的响应头信息
print(f"当前响应报文中的响应头信息：{response.headers}")
# - response.request.headers: 获取当前响应对应的请求报文中的请求头信息
print(f"获取当前请求的请求头信息： {response.request.headers}")
# - response.cookies: 获取服务器设置给客户端的cookie信息。 就是响应头中的set-cookie字段的值
# <RequestsCookieJar[<Cookie BDORZ=27315 for .baidu.com/>]>  获取到的cookie信息类型是 RequestsCookieJar
print(f"当前服务器设置给客户端的cookie信息： {response.cookies}")
# - response.request._cookies: 获取当前在发送请求的时候请求报文中携带的cookie信息。
# <RequestsCookieJar[]>  获取到的内容是一个 RequestsCookieJar 类型
# 为空 就表示在发送请求的时候 并没有携带任何的cookie信息。
print(f"获取在发送请求的时候请求头中携带的cookie： {response.request._cookies}")
```

3.5 案例

https://www.tupianzj.com/meinv/20210601/228566.html

```python
import requests

# 1. 先去获取到整个页面的html字符串
# 要对页面的URL地址发送请求
html_url = "https://www.qqtn.com/article/article_304072_1.html"

# 发送请求 获取到响应 整个页面的响应
response = requests.get(html_url)
print(response.content.decode())

url_list = [

]
```

decode 方法默认什么都不写，就是使用的 utf-8，
手动的去指定一个编码格式进行解码。

```
Run: 05-save_many_image
E:\py_env\spider_py3\Scripts\python.exe E:/爬虫-左文彬/day01-爬虫基础/02-课堂代码/05-save_many_image.py
Traceback (most recent call last):
  File "E:\爬虫-左文彬\day01-爬虫基础\02-课堂代码\05-save_many_image.py", line 9, in <module>
    print(response.content.decode())
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xc1 in position 328: invalid start byte

Process finished with exit code 1
```

在使用 utf-8编码格式对 bytes类型数据进行解码的时候 有错误。那也就是说明utf-8编码不对的。

```python
import requests


# 1. 获取到要保存的图片的url地址
img_url = "https://p.qqan.com/up/2020-10/16034386315737821.png"

# 2. 对图片的url地址发送请求  拿到图片的响应数据
response = requests.get(img_url)
# 3. 将图片的响应数据保存到文件中
# 3.1 with open 打开文件，保存的是图片文件  lyf.jpg
```

```
with open("lyf.png", "wb") as f:
    # 3.2 打开文件的时候，是有文件操作的模式，因为保存的是图片文件，就必须以 bytes类型进行写入 选择 wb
    # 3.3 将响应数据的 bytes类型进行写入， response.content
    f.write(response.content)
#
# w: 将数据以 str 类型写入到文件
# 图片 音频 视频  多媒体文件，这些文件在计算中进行存储的时候要以 bytes类型进行存储
# 说白了就是要以 bytes类型的方式 进行读写操作。
# wb： 以bytes类型进行对数据进行写入。
# rb： 以bytes类型进行对数据进行读取
# with open("aa.txt", "wb")
```

保存多张图片

```
import requests, re, time

# 1. 先去获取到整个页面的html字符串
# 要对页面的URL地址发送请求
html_url = "https://www.qqtn.com/article/article_303842_1.html"

# 发送请求 获取到响应 整个页面的响应
response = requests.get(html_url)
# print(response.content.decode("gbk"))

# 获取一下 整个页面的html字符串内容
html_str = response.content.decode('gbk')

# 获取当前页面中所有的图片的url地址
url_list = re.findall('<img src="(.*?)"/>', html_str)

# 保存图片
for img_url in url_list:
    # num = 1
    # num = url_list.index(img_url)
    # 字符串 切片
    # img_url[start:end:step]
    # 使用切片来取， 直接获取到 图片的后缀名
    image_name = img_url[-10:]
    # 获取到图片的响应对象
    img_response = requests.get(img_url)
    # 将图片的响应内容 保存到文件中
    # with open("lyf_{}.png".format(num), "wb") as f:
    with open(image_name, "wb") as f:
        f.write(img_response.content)
        # num += 1
        print(f'图片 ： {image_name}  保存成功......')
        time.sleep(1)
```