接口请求断言是指在发起请求之后，对返回的响应内容去做判断，用来查看是否响应内容是否与规定的返回值相符。

接口请求断言

响应内容

在发起请求后，我们使用一个变量 r 存储响应的内容，也就是 Response 对象。

```
>>> import requests>>> r = requests.get('
http://httpbin.org/get
')>>> print(r)<Response [200]>
```

Response 对象有很多功能强大的方法可以调用，比如直接获取响应头，获取 Unicode 编码后的响应内容，获取二进制的响应内容，获取原始的响应内容等等。

获得响应头

```
>>> r.headers{'Date': 'Sun, 05 Apr 2020 16:38:09 GMT', \'Content-
Type': 'application/json', \'Content-Length': '308', 'Connection': 'keep-
alive',\ 'Server': 'gunicorn/19.9.0', \ 'Access-Control-Allow-Origin': '*', \ 'Access-
Control-Allow-Credentials': 'true'}
```

获得编码后的响应值：

```
>>> print(r.text)
{  "args": {},   "data": "",   "files": {},   "form": {    "hogwarts": [      "a",
 "b",       "c"    ]  },   "headers": {    "Accept": "*/*",     "Accept-
Encoding": "gzip, deflate",     "Content-Length": "32",     "Content-
Type": "application/x-www-form-urlencoded",     "Host": "httpbin.org",     "User-
Agent": "python-requests/2.22.0",     "X-Amzn-Trace-Id": "Root=1-5e8a01e3-
0913fca09ac605ccc911ccde"  },   "json": null,   "origin": "113.118.101.232",   "url": "
http://httpbin.org/post
"}
```

还可以使用 r.raw 获得原始响应内容，r.content 获得二进制的响应内容，另外还有编码为 JSON 格式的响应内容，会在后面的章节进行详述。

环境安装

安装 JSON 库：

```
pip install json
```

状态码断言

响应状态码断言：

```
import requestsr = requests.get('
http://httpbin.org/get
')assert r.status_code==200
```

assert 是 Python 的内置函数，用来判断表达式，当表达式条件为 False 的时候就会触发异常。

r.status_code 是 Response 对象内的一个方法，用于获得返回值的状态码。

assert r.status_code==200 就是在判断状态码是否等于200，如果不等于200则会抛出异常。

反例：响应状态码断言，判断响应状态码是否为400

```
>>> import requests>>> r = requests.get('
http://httpbin.org/get
')>>> assert r.status_code==400Traceback (most recent call last):  Fil
e "<stdin>", line 1, in <module>AssertionError
```

从上个例子可以知道，这个响应状态码应该是 200，因为与 400 不相等，所以抛出了异常。

JSON 响应断言

在测试过程中，大部分接口的返回值都为 JSON 格式。所以，掌握如何对 JSON 响应值进行断言这一技能，可以更轻松的完善接口自动化测试用例。

对响应内容进行 JSON 编码

r.json() 对于响应值 **r** 先进行 JSON 编码：

```
>>> import requests>>> r = requests.post('
http://httpbin.org/post
', data = {'hogwarts':["a","b","c"]})>>> r.json()
{'args': {}, 'data': '', 'files': {}, 'form': {'hogwarts': ['a', 'b',
'c']},\ 'headers': {'Accept': '*/*', 'Accept-
Encoding': 'gzip, deflate',\  'Content-Length': '32', 'Content-
Type': 'application/x-www-form-
urlencoded',\  'Host': 'httpbin.org', 'User-Agent': 'python-
requests/2.22.0',\  'X-Amzn-Trace-Id': 'Root=1-5e8a01e3-
0913fca09ac605ccc911ccde'},\  'json': None, 'origin': '113.118.101.232
', 'url': '
http://httpbin.org/post
'}
```

多种类型响应值断言案例

对于字典格式，可以通过 dict["key"] 的方式拿到 value 值.

对于列表格式，可以通过 list[index] 拿到对应索引的 value 值。

在 JSON 的断言之中，主要应用的就是字典和列表自带的查找方法。如果碰到混合或者嵌套的情况，只需要一层一层拨开，直到找到需要进行断言的字段即可。

```
{'args': {}, 'data': '', 'files': {}, \'form': {'hogwarts': ['a', 'b', 'c']}, \'headers'
: {'Accept': '*/*', 'Accept-Encoding': 'gzip, deflate', \'Content-
Length': '32', 'Content-Type': 'application/x-www-form-
urlencoded',\ 'Host': 'httpbin.org', 'User-Agent': 'python-requests/2.22.0', \ 'X-Amzn-
Trace-Id': 'Root=1-5e8a01e3-
0913fca09ac605ccc911ccde'}, \ 'json': None, 'origin': '113.118.101.232', \ 'url': '
http://httpbin.org/post
'}
```

字典格式断言，判断 headers 中的 Host 为 httpbin.org

```
>>> import requests>>> r = requests.post('http://httpbin.org/post', data = {'hogwarts':["a","b","c"]})>>> assert r.json()['headers']["Host"] == "httpbin.org"
```

1. 第一层是 key 值为 "header" 的 value
2. 第二层是 key 值为 "Host" 的 value
3. 判断 key 值为 "Host" 的 value 值是否与 "httpbin.org" 相等

字典混合列表格式断言，判断 hogwarts 对应的列表的第一位是 'a'

```
>>> import requests>>> r = requests.post('http://httpbin.org/post', data = {'hogwarts':["a","b","c"]})>>> assert r.json()['form']["hogwarts"][0] == "a"
```

1. 第一层是 key 值为 'form' 的 value
2. 第二层是 key 值为 'hogwarts' 的 value
3. 第三层是索引为 0 的 value
4. 判断上一步索引为 0 的 value 是否等于 "a"