参数化和封装

- 讲师：潘sir

函数

- 函数的定义：def + 函数名称 + 小括号 + 变量(可没有) + 冒号，例如：`def add(a, b):`
- 函数三要素：入参，逻辑，返回：
    1. 类比接口文档三要素：接口功能，接口入参，接口返回
    2. 一个函数就是一个api接口，只不过这是内部接口，而http是对外开放的接口
    3. 说明：可以没有入参，入参可为空，入参也可有默认；代码逻辑中可以包含注释；返回语句可不写，不写则返回None

```
# 复用性

# 流水账
c = 1 + 2
# 1. 测试需要改动的数据
a = 1
b = 2
# 2. 测试的执行过程
c = a + b # 1 + 2
d = a + c # 1 + c
e = d + b # d + 2
# 3. 测试结果的收集
print(c)

# 抽象出来
# 参数化：测试需要改动的数据 -> 函数入参
# 执行过程 -> 函数体
# 实际结果 -> return
def add(a, b):
    '''
    加法运算
    :param a: 数字
    :param b: 数字
    :return: 两个数的和
    '''
    c = a + b # 1 + 2
    d = a + c # 1 + c
    e = d + b # d + 2
    print(c)
    return c
```

```
if __name__ == '__main__':
    res = add(2, 3)
    print(res)
```

简单封装

```
def aa():
    # 流水账
    # 1. 设置静态数据
    s = requests.session()
    url = 'http://127.0.0.1:5001/login'  # 登录
    url2 = 'http://127.0.0.1:5001/user/dcs'  # 登录后的页面，用作判断

    # 2. 获取动态数据
    r = s.get(url)
    csrf_token = re.findall(r'csrf_token.*?value="(.*?)">-', r.text)

    # 3. 参数关联要发送的数据
    data = {
        "username": "dcs",
        "password": "123",
        "csrf_token": csrf_token
    }

    # 4. 发送请求
    r2 = s.post(url, data=data)

    # 5. 判断是否登录成功
    r3 = s.get(url2)
    assert "资料" in r3.text, "登录失败"
    print("ok")
```

```
ok
```

- 简单封装，往往达不到复用的要求

```
import requests, re


def login_c(user, passwd):
    # 1. 设置静态数据
    s = requests.session()
    url = 'http://127.0.0.1:5001/login'  # 登录
    url2 = 'http://127.0.0.1:5001/user/dcs'  # 登录后的页面，用作判断
    url3 = 'http://127.0.0.1:5001/user/' + user
```

```
    url4 = 'http://127.0.0.1:5001/user/%s'  % user
    # 2. 获取动态数据
    r = s.get(url)
    csrf_token = re.findall(r'csrf_token.*?value="(.*?)">-', r.text)

    # 3. 参数关联要发送的数据
    data = {
        "username": "dcs",
        "password": "123",
        "submit": "Login",
        "csrf_token": csrf_token
    }

    # 4. 发送请求
    r2 = s.post(url, data=data)

    # 5. 判断是否登录成功
    r3 = s.get(url2)
    assert "资料" in r3.text, "登录失败"
    print("ok")
    return r3.text


if __name__ == '__main__':
    login_c()
```

```
ok
```

优化封装step1

- 剥离变量，可以正常复用

```
import requests, re

# 1. 抽离全局变量
s = requests.session()

# 2. 考虑测试时user和passwd经常变化，将它们参数化
def login_c(user, passwd):

    url = 'http://127.0.0.1:5001/login'  # 登录
    # 3. 注意到这个url跟随user变化，参数化的时候需要做一致化处理
#     url2 = 'http://127.0.0.1:5001/user/dcs'  # 固定
#     url2 = 'http://127.0.0.1:5001/user/%s' % user  # 通用

    # 2. 获取动态数据
    r = s.get(url)
    csrf_token = re.findall(r'csrf_token.*?value="(.*?)">-', r.text)
```

```
    # 3. 参数化要发送的数据
    data = {
        "username": user,
        "password": passwd,
        "csrf_token": csrf_token
    }

    # 4. 发送请求，获取登录token
    r2 = s.post(url, data=data)

    # 5. 判断是否登录成功
    r3 = s.get(url2)
    assert "资料" in r3.text, "登录失败"
    print("登录成功")
    # 返回html页面
    return r3.text

def is_login(user):
    url2 = 'http://127.0.0.1:5001/user/%s' % user  # 通用
    # 实际访问的过程
    r3 = s.get(url2)
    if "资料" in r3.text:
        return True
    else:
        return False
#     assert "资料" in r3.text, "登录失败"
#     print("登录成功")
#     # 返回html页面
#     return r3.text

a = is_login("dcs")
assert a == True
assert a == False
if __name__ == '__main__':
    a = login_c("dcs", "123")
    a = login_c("root", "123")
```

```
登录成功
登录成功
```

优化封装step2

- 添加日志和注释，在报告中方便定位问题
- 考虑到，后续需要测试各种异常登录场景，如果把判断逻辑写在一起，可能会导致用例编写困难，所以，进行逻辑拆分，将粒度变细

```python
import requests, re

# 1. 全局变量
s = requests.session()

# 2. 专门负责登录
def login_c(user, passwd):
    url = 'http://127.0.0.1:5001/login'  # 登录

    # 3. 获取token，进行异常捕获和日志记录
    try:
        r = s.get(url)
        csrf_token = re.findall(r'csrf_token.*?value="(.*?)">-', r.text)[0]
        print("获取页面token:", csrf_token)
    except:
        print("获取页面token失败")

    data = {
        "username": user,
        "password": passwd,
        "submit": "Login",
        "csrf_token": csrf_token
    }

    r2 = s.post(url, data=data)
    return r2.text

# 4. 专门负责校验
def is_login(user):
    url2 = 'http://127.0.0.1:5001/user/%s' % user  # 登录后的页面，用作判断
    res = s.get(url2)
    if "资料" in res.text:
        print("登录成功")
        return True
    else:
        print("登录失败")
        return False


if __name__ == '__main__':
    user = "dcs"
    passwd = "123"
    login_c(user, passwd)
    a = is_login(user)
    assert a
    # print(a)
```

```
获取页面token:
IjRiNGY0YTk0NmZhZGI3NWQ1NzQ0ZWFhMzAxNjBmYTMyYWUzMjdhN2Ei.EJaShw.5zBCiijJ0H6iyV
9s-EGXlfHy8Ik
登录成功
```

unittest中引用

```
import unittest, requests
from tsms.tsms_login_def import login_c, is_login


class Testlogin(unittest.TestCase):
    def setUp(self) -> None:
        pass

    def tearDown(self) -> None:
        pass

    def test_login_success(self):
        '''登录成功'''
        a = login_c("dcs", "123")
        self.assertTrue(is_login("dcs"))

    def test_login_fail(self):
        '''登录失败'''
        a = login_c("dcs", "1234")
        self.assertFalse(is_login("dcs"))


if __name__ == '__main__':
    # 执行本suite
    unittest.main(argv=['ignored', '-v'], exit=False)
```

```
test_login_fail (__main__.Testlogin)
登录失败 ... ok
test_login_success (__main__.Testlogin)
登录成功 ...

获取页面token:
IjBjOWMzNWI2OWM1M2UxMDQyMTczMDRiZWQxMTBiNzM0N2RlNDNhZjAi.EIlL2w.VWolAo1bFpKJhl
BMGViHX6bHIPw
登录失败
获取页面token:
IjBjOWMzNWI2OWM1M2UxMDQyMTczMDRiZWQxMTBiNzM0N2RlNDNhZjAi.EIlL2w.VWolAo1bFpKJhl
BMGViHX6bHIPw
登录成功
```

```
ok

----------------------------------------------------------------------
Ran 2 tests in 0.356s

OK
```

优化step3封装类

- 可以支持扩展，管理更方便

```
import requests, re


# 定义类
class TestLogin():

    def __init__(self):
        # 1. 全局变量，也是实例化一个对象
        self.s = requests.session()

    # self表示这个 login_c 属于 TestLogin
    def login_c(self, user, passwd):
        url = 'http://127.0.0.1:5001/login'  # 登录
        # 2. 获取token
        try:
            r = self.s.get(url)
            csrf_token = re.findall(r'csrf_token.*?value="(.*?)">-', r.text)
[0]
            print("获取页面token:", csrf_token)
        except:
            print("获取页面token失败")

        # 3. 参数化要发送的数据
        data = {
            "username": user,
            "password": passwd,
            "submit": "Login",
            "csrf_token": csrf_token
        }

        # 4. 发送请求，获取登录token
        r2 = self.s.post(url, data=data)
        return r2.text

    def is_login(self, user):
        url2 = 'http://127.0.0.1:5001/user/%s' % user  # 登录后的页面，用作判断
        res = self.s.get(url2)
```

```
        if "资料" in res.text:
            print("登录成功")
            return True
        else:
            print("登录失败")
            return False


if __name__ == '__main__':
    # 实例化对象
    lo = TestLogin()
    # 调用类的方法b
    a = lo.login_c("panwj", "123")
    # 继续调用类的方法
    assert lo.is_login("panwj")
```

```
获取页面token:
ImRjMjhjODJhYTgxNDgyYjUyYTM3MDUzOTJkYTVjNWRjNWM0NjBhNjki.EJU26A.OsvXjmJqDb93yh
LukbASPFBXiWs
登录成功
```

结合unittest

```
import unittest
from tsms.tsms_login_cls import TestLogin


class Tg(unittest.TestCase):
#     def setUp(self):
#         self.t = TestLogin()

#     def tearDown(self):
#         self.t.close()

    @classmethod
    def setUpClass(cls) -> None:
        # 实例化
        cls.t = TestLogin()

    @classmethod
    def tearDownClass(cls) -> None:
        cls.t.close()

    def test_login_success(self):
        '''测试登录成功'''
        a = self.t.login_c("dcs", "123")
        self.assertTrue(self.t.is_login("dcs"))
```

```
    def test_login_fail(self):
        '''测试登录失败'''
        a = self.t.login_c("dcs", "1234")
        self.assertFalse(self.t.is_login("dcs"))


if __name__ == '__main__':
    # 执行本suite
    unittest.main(argv=['ignored', '-v'], exit=False)
```

添加正则方法

将正则匹配的方法，封装到类里，整合成一个新的类，专门用于前端相关的接口和web测试

```
html = '''    <tr>
                                <td>
                                    <p>2019-10-20 08:48:39</p>
                                </td>
                                <td>
                                    <p>179</p>
                                </td>
                                <td>
                                    <p>名称</p>
                                </td>
                                <td>
                                    <p>b7R932QG</p>
                                </td>
                                <td>
                                    <p>reviewing</p>
                                </td>
                            </tr>

                            <tr>
                                <td>
                                    <p>2019-10-20 08:48:39</p>
                                </td>
                                <td>
                                    <p>180</p>
                                </td>
                                <td>
                                    <p>名称</p>
                                </td>
                                <td>
                                    <p>538YGtv3</p>
                                </td>
                                <td>
                                    <p>passed</p>
                                </td>
```

```
                        </tr>

                        <tr>
                            <td>
                                <p>2019-10-20 08:48:39</p>
                            </td>
                            <td>
                                <p>181</p>
                            </td>
                            <td>
                                <p>名称</p>
                            </td>
                            <td>
                                <p>45咿衝uz</p>
                            </td>
                            <td>
                                <p>reviewing</p>
                            </td>
                        </tr>'''
import re

# 1.方法1：一直重复.*?<p>(.*?)</p>
a = re.findall("<tr>.*?<p>(.*?)</p>.*?<p>(.*?)</p>.*?<p>(.*?)</p>.*?<p>(.*?)
</p>.*?<p>(.*?)</p>.*?</tr>", html, re.S)
print(len(a))
for i in a:
    print(i)
```

```
3
('2019-10-20 08:48:39', '179', '名称', 'b7R932QG', 'reviewing')
('2019-10-20 08:48:39', '180', '名称', '538YGtv3', 'passed')
('2019-10-20 08:48:39', '181', '名称', '45咿衝uz', 'reviewing')
```

- 封装成方法

```
import requests, re

# 定义类
class TsmsWeb():

    def __init__(self):
        # 1. 全局变量，也是实例化一个对象
        self.s = requests.session()

    # self表示这个 login_c 属于 TestLogin
    def login_c(self, user, passwd):
        url = 'http://127.0.0.1:5001/login'  # 登录
        # 2. 获取token
```

```
        try:
            r = self.s.get(url)
            csrf_token = re.findall(r'csrf_token.*?value="(.*?)">-', r.text)
[0]
            print("获取页面token:", csrf_token)
        except:
            print("获取页面token失败")

        # 3. 参数化要发送的数据
        data = {
            "username": user,
            "password": passwd,
            "submit": "Login",
            "csrf_token": csrf_token
        }

        # 4. 发送请求，获取登录token
        r2 = self.s.post(url, data=data)
        return r2.text

    def is_login(self, user):
        url2 = 'http://127.0.0.1:5001/user/%s' % user  # 登录后的页面，用作判断
        res = self.s.get(url2)
        if "资料" in res.text:
            print("登录成功")
            return True
        else:
            print("登录失败")
            return False

    def reg_sign(self, html):
        a = re.findall("<tr>.*?<p>(.*?)</p>.*?<p>(.*?)</p>.*?<p>(.*?)</p>.*?
<p>(.*?)</p>.*?</tr>", html, re.S)
        return a

    def reg_temp(self, html):
        pass

# 实际使用
# 1. 先完成登录
tb = TsmsWeb()
user = 'dcs'
passwd = "123"
tb.login_c(user, passwd)
assert tb.is_login(user)
# 2. 再进行正则解析
url = 'http://127.0.0.1:5001/user/dcs/sign'
res = tb.s.get(url)
html = res.text
```

```
sign_list = tb.reg_sign(html)
print(sign_list,len(sign_list))
```

说明：步骤复杂，调用的时候需要自己进行参数关联，并进行多个步骤，为了调用方便考虑进行封装 从两个方面考虑：

1. 关注功能点(关注返回结果)，本例的目的：从前端web中获取 签名表格内容
2. 关注场景变更(关注入参)：需要剥离哪些参数?
    - 域名是否会变更? http://127.0.0.1:5001 变更情况少，但是存在，考虑封装成类变量，并给一个默认值，从配置文件读取
    - 用户名是否会变更? 变更情况多，直接封装成方法的位置参数/默认参数
    - 查询接口名称是否会变更? 比如这次查sign，下次查temp，方式类似(正则匹配)，但接口名称不一样
        1. 封装成一个方法，通过标志性来走不同的流程，实现不同的功能
        2. 封装成两个方法，各自走不同的流程，实现不同的功能

优化传参方式

因为我们只需要知道某个用户下的签名信息，而不关注整个html内容是什么，改造传参方式

```
import requests, re, logging

logging.basicConfig(level=logging.INFO, format='%(asctime)-16s %(levelname)-8s
%(message)s')


# 定义类
class TsmsWeb(object):

    def __init__(self):
        # 1. 全局变量，也是实例化一个对象
        self.s = requests.session()
        self.url_head = "http://127.0.0.1:5001"

    # self表示这个 login_c 属于 TestLogin
    def login_c(self, user='dcs', passwd='123'):
        url = self.url_head + '/login'  # 登录
        # 2. 获取token
        try:
            r = self.s.get(url)
            csrf_token = re.findall(r'csrf_token.*?value="(.*?)">-', r.text)
[0]
            logging.info("[获取页面token]: {}".format(csrf_token))
        except:
            logging.WARN("[获取页面token失败]")
            return
```

```
        # 3. 参数化要发送的数据
        data = {
            "username": user,
            "password": passwd,
            "submit": "Login",
            "csrf_token": csrf_token
        }

        # 4. 发送请求，获取登录token
        r2 = self.s.post(url, data=data)
        return r2.text

    def is_login(self, user='dcs'):
        url2 = self.url_head + '/user/%s' % user  # 登录后的页面，用作判断
        res = self.s.get(url2)
        if "资料" in res.text:
            logging.info("登录成功")
            return True
        else:
            logging.WARN("登录失败")
            return False

    def reg_sign(self, user='dcs'):
        url = self.url_head + '/user/%s/sign' % user
        html = self.s.get(url).text
        a = re.findall("<tr>.*?<p>(.*?)</p>.*?<p>(.*?)</p>.*?<p>(.*?)</p>.*?
<p>(.*?)</p>.*?</tr>", html, re.S)
        return a

    def reg_temp(self, html):
        pass


if __name__ == '__main__':
    # 实例化
    tb = TsmsWeb()
    # 1. 实现登录
    user = 'dcs'
    passwd = '123'
    tb.login_c(user, passwd)
    tb.is_login(user)
    # 2. 匹配结果
    sign_list = tb.reg_sign(user)
    print(sign_list, len(sign_list))
```

- 结合测试用例

```
from tsms.tsms_web import TsmsWeb
```

```
from tsms.tsms_base import Tsmstest
import unittest
from time import sleep
from retrying import retry
class TestWeb(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.tb = TsmsWeb()
        cls.ts = Tsmstest()

    @classmethod
    def tearDownClass(cls):
        pass

    def test_add_sign(self):
        data = self.ts.sign_data(4, 8)
        self.ts.req_post('sign', data)
        assert self.ts.status_code == 200
        assert isinstance(self.ts.json["sign_id"], int)

        self.tb.login_c()
        # 确保前置动作是通过
        assert self.tb.is_login()
        sign_list = self.tb.reg_sign()
        # 获取需要的那行数据
        a = ''
        for record in sign_list:
            # 注意接口返回的sign_id是整型，这里必须转换成字符串
            if str(self.ts.json["sign_id"]) in record:
                a = record
        assert a != ''

        print(a, type(a))
        # 断言
        # 1. 接口返回的sign_id 和 页面查到的sign_id 一致
        assert str(self.ts.json["sign_id"]) == a[1]
        # 2. 请求body构造的随机name 和 页面查到的一致
        assert data["signature"] == a[2]
        # 3. 页面的审核状态是reviewing
        assert a[3] == 'reviewing'


if __name__ == '__main__':
    # 执行本suite
    unittest.main(argv=['ignored', '-v'], exit=False)
```

- 练习：将获取数据的功能，再方法再次进行封装

```
import requests, re, logging
```

```
logging.basicConfig(level=logging.INFO, format='%(asctime)-16s %(levelname)-8s
%(message)s')


# 定义类
class TsmsWeb(object):

    def __init__(self):
        # 1. 全局变量，也是实例化一个对象
        self.s = requests.session()
        self.url_head = "http://127.0.0.1:5001"

    # self表示这个 login_c 属于 TestLogin
    def login_c(self, user='dcs', passwd='123'):
        """执行登录"""
        url = self.url_head + '/login'  # 登录
        # 2. 获取token
        try:
            r = self.s.get(url)
            csrf_token = re.findall(r'csrf_token.*?value="(.*?)">-', r.text)
[0]
            logging.info("[获取页面token]: {}".format(csrf_token))
        except:
            logging.WARN("[获取页面token失败]")
            return

        # 3. 参数化要发送的数据
        data = {
            "username": user,
            "password": passwd,
            "submit": "Login",
            "csrf_token": csrf_token
        }

        # 4. 发送请求，获取登录token
        r2 = self.s.post(url, data=data)
        return r2.text

    def is_login(self, user='dcs'):
        """判断是否登录成功"""
        url2 = self.url_head + '/user/%s' % user  # 登录后的页面，用作判断
        res = self.s.get(url2)
        if "资料" in res.text:
            logging.info("登录成功")
            return True
        else:
            logging.WARN("登录失败")
            return False
```

```
    def reg_sign(self, user='dcs'):
        """正则匹配签名表单"""
        url = self.url_head + '/user/%s/sign' % user
        html = self.s.get(url).text
        a = re.findall("<tr>.*?<p>(.*?)</p>.*?<p>(.*?)</p>.*?<p>(.*?)</p>.*?
<p>(.*?)</p>.*?</tr>", html, re.S)
        return a

    def reg_temp(self, html):
        pass

    def get_res(self, exp_id, records_list):
        """从列表中获取指定的数据"""
        exp_res = ''
        for record in records_list:
            # 注意接口返回的sign_id是整型，这里必须转换成字符串
            if str(exp_id) in record:
                exp_res = record
        # 注意，在方法里，就不需要使用断言
        # assert exp_res != ''
        return exp_res

if __name__ == '__main__':
    # 实例化
    tb = TsmsWeb()
    # 1. 实现登录
    user = 'dcs'
    passwd = '123'
    tb.login_c(user, passwd)
    tb.is_login(user)
    # 2. 匹配结果
    sign_list = tb.reg_sign(user)
#     print(sign_list, len(sign_list))
    # 3. 解析结果
    a = tb.get_res(1099, sign_list)
    print(a)
```

```
2019-10-26 13:56:30,723 INFO     [获取页面token]:
IjI2ZWU4YmU5NDZlNzEzYjhiM2IwNTRmNjI0NzViZWQ5ZTAyM2UzNGQi.EJVxjg.1gHMssx6YZA6Uz
MI8s-P9SBz5Zg
2019-10-26 13:56:30,885 INFO     登录成功
```

```
('2019-10-26 09:45:00', '1099', 'u7H26S', 'reviewing')
```

- 将新增的方法引入到用例中

```
from tsms.tsms_web import TsmsWeb
from tsms.tsms_base import Tsmstest
import unittest
from time import sleep
from retrying import retry
class TestWeb(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.tb = TsmsWeb()
        cls.ts = Tsmstest()

    @classmethod
    def tearDownClass(cls):
        pass

    def test_add_sign(self):
        # 调接口
        data = self.ts.sign_data(4, 8)
        self.ts.req_post('sign', data)
        assert self.ts.status_code == 200
        assert isinstance(self.ts.json["sign_id"], int)
        # 查前端
        self.tb.login_c()
        assert self.tb.is_login()
        sign_list = self.tb.reg_sign()
        real_res = self.tb.get_res(self.ts.json["sign_id"], sign_list)
        assert str(self.ts.json["sign_id"]) == real_res[1]
        assert data["signature"] == real_res[2]
        assert real_res[3] == 'reviewing'


if __name__ == '__main__':
    # 执行本suite
    unittest.main(argv=['ignored', '-v'], exit=False)
```

- 再次优化前置/后置，得到最终版本

```
from tsms.tsms_web import TsmsWeb
from tsms.tsms_base import Tsmstest
import unittest
from time import sleep
from retrying import retry
class TestWeb(unittest.TestCase):
    @classmethod
    def setUpClass(cls):
        cls.tb = TsmsWeb()
        cls.ts = Tsmstest()
        # 登录
        cls.tb.login_c()
```

```
        assert cls.tb.is_login()

    @classmethod
    def tearDownClass(cls):
        pass

    def test_add_sign(self):
        # 调接口
        data = self.ts.sign_data(4, 8)
        self.ts.req_post('sign', data)
        assert self.ts.status_code == 200
        assert isinstance(self.ts.json["sign_id"], int)
        # 查前端
        sign_list = self.tb.reg_sign()
        real_res = self.tb.get_res(self.ts.json["sign_id"], sign_list)
        assert str(self.ts.json["sign_id"]) == real_res[1]
        assert data["signature"] == real_res[2]
        assert real_res[3] == 'reviewing'


if __name__ == '__main__':
    # 执行本suite
    unittest.main(argv=['ignored', '-v'], exit=False)
```

```
2019-10-26 14:05:07,097 INFO     [获取页面token]:
ImEwOWE3YjYyYzBmNGRlYzlmNDI1OWE5M2EzOWJmNTBkMDYzZGVkNTIi.EJVzkw.kXqHGgo46QmU07
xRXNuvFN8fsoA
2019-10-26 14:05:07,242 INFO     [现在登录页面是]: http://127.0.0.1:5001/user/dcs
2019-10-26 14:05:07,254 INFO     登录成功
test_add_sign (__main__.TestWeb) ... 2019-10-26 14:05:07,255 INFO     长度为:7
2019-10-26 14:05:07,256 INFO     [当前被调用方法是]: gen_ranstr
2019-10-26 14:05:07,258 INFO     [num_letters is]: 7
2019-10-26 14:05:07,260 INFO     [执行结果为]: zlJeeVH
2019-10-26 14:05:07,261 INFO     [当前被调用方法是]: req_post
2019-10-26 14:05:07,262 INFO     [url_type is]: sign
2019-10-26 14:05:07,265 INFO     [data is]: {'signature': 'zlJeeVH', 'source':
'深圳', 'pics': ['tu1']}
2019-10-26 14:05:07,266 INFO     [当前请求的地址是]:
http://127.0.0.1:5001/v1/signature
2019-10-26 14:05:07,267 INFO     [发送内容是]: {'signature': 'zlJeeVH',
'source': '深圳', 'pics': ['tu1']} <class 'dict'>
2019-10-26 14:05:07,410 INFO     [返回码是:] 200
2019-10-26 14:05:07,411 INFO     [返回内容是]: {'sign_id': 1172}
2019-10-26 14:05:07,411 INFO     [执行结果为]: {'sign_id': 1172}
2019-10-26 14:05:07,412 INFO     [现在要正则匹配的页面是]:
http://127.0.0.1:5001/user/dcs/sign
ok

----------------------------------------------------------------------
```

```
Ran 1 test in 0.372s

OK
```

```
```