《Python 程序设计高阶》

(2023-2024 学年第 1 学期)

实验报告二

学号:2021329600006　姓名: 陈昊天　班级: 计算机科学与技术 21(4) 班
学号:2021329621213　姓名: 陈佳伟　班级: 计算机科学与技术 21(3) 班
学号:2021329621257　姓名: 冯佳钧　班级: 计算机科学与技术 21(4) 班

§1　实验题 1

修改课件中的下载安全标准信息的爬虫，使之能够正常工作，将下载的信息保存在“安全.csv”文件中；参考的结果，看课件中安全.csv 这一节的输出结果；

§1.1　解答:

解答要对你实验结果截图，加上你的代码。

修改了 SSL 证书问题 [1]，在 requests.get 加入 verify=False 参数。

1.1.1　代码

```python
import requests  # 关键库，用于发送Http请求     # pip install requests
from lxml import etree  # xml解析包，ElementTree  # pip install lxml
import time
import random
import re
import csv
import ssl

# 忽略 SSL 证书警告
requests.packages.urllib3.disable_warnings(
    requests.packages.urllib3.exceptions.InsecureRequestWarning
)


class Spider:
    def __init__(self, name):
        self.name = name
        self.url = (
            "https://openstd.samr.gov.cn/bzgk/gb/std_list?p.p1=0&p.p2="
            + name
```

```
            + "&p.p90=circulation_date&p.p91=desc"
        )
        print(self.url)
        self.headers = [
            "Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML,
               like Gecko) Chrome/39.0.2171.95 Safari/537.36",
            "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2)
               AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153
               Safari/537.36",
            "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:30.0) Gecko/20100101
               Firefox/30.0",
            "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2)
               AppleWebKit/537.75.14 (KHTML, like Gecko) Version/7.0.3
               Safari/537.75.14",
            "Mozilla/5.0 (compatible; MSIE 10.0; Windows NT 6.2; Win64;
               x64; Trident/6.0)",
            "Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.11)
               Gecko/20071127 Firefox/2.0.0.11",
            "Opera/9.25 (Windows NT 5.1; U; en)",
            "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET
               CLR 1.1.4322; .NET CLR 2.0.50727)",
            "Mozilla/5.0 (compatible; Konqueror/3.5; Linux) KHTML/3.5.5
               (like Gecko) (Kubuntu)",
            "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.8.0.12)
               Gecko/20070731 Ubuntu/dapper-security Firefox/1.5.0.12",
            "Lynx/2.8.5rel.1 libwww-FM/2.14 SSL-MM/1.4.1 GNUTLS/1.2.9",
            "Mozilla/5.0 (X11; Linux i686) AppleWebKit/535.7 (KHTML, like
               Gecko) Ubuntu/11.04 Chromium/16.0.912.77 Chrome/16.0.912.77
               Safari/535.7",
            "Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:10.0) Gecko/20100101
               Firefox/10.0 ",
        ]

    def get_each_link(self):
        """获得首页链接, 返回列表"""
        index = 1  # 翻页
        link_list = []  # 储存每一个链接的列表
        while True:
            try:
                headers = {
                    "User-Agent": random.choice(self.headers)  #
                       随机选择请求头, 模拟不同的浏览器
                }
                session = requests.session()
```

```
                session.verify = False
                r = session.get(
                    self.url, headers=headers, verify=False
                )  # 请求数据,返回给Response对象
                # r = requests.get(self.url, headers=headers)    #
                    请求数据,返回给Response对象
                r.encoding = r.apparent_encoding  # 设置编码
                # print(r.status_code)
                html = r.text
                selector = etree.HTML(html)  # 将网页源码转换为 XPath
                    可以解析的格式
                Ids = selector.xpath(
                    '//tr/td[@style="text-align: left;"]/a/@onclick'
                )  # 选择结点
                print(Ids)

                # 判断是否获取结束
                if len(Ids) == 0:
                    break

                for each_id in Ids:
                    each_id = each_id.replace("showInfo('",
                        "").replace("');", "")
                    # link = 'http://www.gb688.cn/bzgk/gb/newGbInfo?hcno='
                        + each_id
                    link = (
                        "https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno="
                            + each_id
                    )

                    print(link)
                    link_list.append(link)
                index += 1
                if index > 1:  # 只请求1页
                    break
                # self.url =
                    'http://www.gb688.cn/bzgk/gb/std_list?r=0.40987171406906286&page='
                    + str(index) + '&pageSize=10&p.p1=0&p.p2=' + self.name +
                    '&p.p90=circulation_date&p.p91=desc'
                self.url = (
                    "https://openstd.samr.gov.cn/bzgk/gb/std_list?r=0.4098717140690628
                    + str(index)
                    + "&pageSize=10&p.p1=0&p.p2="
                    + self.name
```

```
                    + "&p.p90=circulation_date&p.p91=desc"
                )
                print(self.url)
                time.sleep(random.random() + random.randint(0, 1))
            except Exception as e:
                print(e)
                break
        return link_list

    def get_detial_from_linkList(self, link_list):
        """从列表的链接下载文件内容"""
        path = self.name + ".csv"
        i = 0  # 序号
        # 写入表头
        info = [
            "序号",
            "标准编号",
            "标准名称",
            "英文名称",
            "状态",
            "中国标准分类号（CCS）",
            "国际标准分类号（ICS）",
            "发布日期",
            "实施日期",
            "主管部门",
            "归口单位",
            "发布单位",
            "备注",
        ]
        with open(path, "a") as f:
            csv_writer = csv.writer(f)
            csv_writer.writerow(info)
        # with open(path, "a", newline="", encoding="utf-8-sig") as f:
        #     csv_writer = csv.writer(f)
        #     csv_writer.writerow(info)
        for link in link_list:
            try:
                headers = {"User-Agent": random.choice(self.headers)}
                r = requests.get(link, headers=headers, verify=False)
                r.encoding = r.apparent_encoding
                html = r.text
                selector = etree.HTML(html)
                print(selector.xpath("//td/h1/text()"))
                number =
```

```
        selector.xpath("//td/h1/text()")[0].strip().replace("标准号：",
        "")
    name = selector.xpath("//td/b/text()")[0].strip()
    E_name = re.findall("<td>英文标准名称：(.+)</td>", html)[
        0
    ]  # 这里使用正则表达式，因为xpath莫名其妙使用不了
    status =
        selector.xpath('//td[@align="left"]/span[@class]/text()')[
        0
    ].strip()
    ccs = selector.xpath(
        '//div[@class="col-xs-12 col-md-4 content"]/text()'
    )[
        0
    ].strip()  # 中国标准分类号（CCS）
    ics = selector.xpath(
        '//div[@class="col-xs-12 col-md-4 content"]/text()'
    )[
        1
    ].strip()  # 国际标准分类号（ICS）
    release_date = selector.xpath(
        '//div[@class="col-xs-12 col-md-4 content"]/text()'
    )[
        2
    ].strip()  # 发布日期
    implement_date = selector.xpath(
        '//div[@class="col-xs-12 col-md-4 content"]/text()'
    )[
        3
    ].strip()  # 实施日期
    department = selector.xpath(
        '//div[@class="col-xs-12 col-md-4 content"]/text()'
    )[
        4
    ].strip()  # 主管部门
    unit = selector.xpath(
        '//div[@class="col-xs-12 col-md-4 content"]/text()'
    )[
        5
    ].strip()  # 归口单位
    f_unit = selector.xpath(
        '//div[@class="col-xs-12 col-md-10 content"]/text()'
    )[
        0
```

```
            ].strip()  # 发布单位
            other = selector.xpath(
                '//div[@class="col-xs-12 col-md-10 content"]/text()'
            )[
                1
            ].strip()  # 备注

            i += 1
            info = []
            info.append(i)
            info.append(number)
            info.append(name)
            info.append(E_name)
            info.append(status)
            info.append(ccs)
            info.append(ics)
            info.append(release_date)
            info.append(implement_date)
            info.append(department)
            info.append(unit)
            info.append(f_unit)
            info.append(other)

            # # 用None替换不存在的项目
            # for j in range(len(info)):
            #     if len(info) == 0:
            #         info = "None"
            # 正确处理 info 列表为空的情况
            for j in range(len(info)):
                if info[j] == "":
                    info[j] = "None"

            with open(path, "a") as f:
                csv_writer = csv.writer(f)
                csv_writer.writerow(info)
            # with open(path, "a", newline="", encoding="utf-8-sig") as
               f:
            #     csv_writer = csv.writer(f)
            #     csv_writer.writerow(info)

        except Exception as e:
            print("Downloa Error:", link, e)


```

```
def main():
    print("开始下载有关‘安全’的国家标准")
    scrapy = Spider("安全")

    print("正在获取链接...")
    link_list = scrapy.get_each_link()
    print("获取完毕共%d个" % len(link_list))
    print("正在下载...")

    # link_list =
        ['http://www.gb688.cn/bzgk/gb/newGbInfo?hcno=E6CD13ACF8E3B3BEFAA16852499676D0']
    scrapy.get_detial_from_linkList(link_list)

    print("有关‘安全’的国家标准下载完毕")


if __name__ == "__main__":
    main()
```

1.1.2 截图

```
(base) nanmener@Haotians-MacBook-Pro 10. Spider % /Applications/Anaconda3/anaconda3/bin/python "/Users/nanmene
r/Github/zstu-study/Python程序设计-高阶/Python programming@ZSTU by Laitude/10. Spider/1.py"
开始下载有关‘安全’的国家标准
https://openstd.samr.gov.cn/bzgk/gb/std_list?p.p1=0&p.p2=安全&p.p90=circulation_date&p.p91=desc
正在获取链接...
["showInfo('B3460D59D0BF065AADE72E239964574F');", "showInfo('61473F823BB6A9F954DEDFF020D85111');", "showInfo('17D551A10
8AA6EDB56633768FCD9310F');", "showInfo('37D0509D06E0F6243DE88A54351426DE');", "showInfo('C4AB705DA0063542548212CD76070D
2D');", "showInfo('1271E1730AF7C01110E54AE5F77A18D0');", "showInfo('CCC8109C92E1B03B92A18F8B87B14C5D');", "showInfo('31
ECB92BAE10BBA109E2FA227D68B81D');", "showInfo('559E267F161240C4E953889C802A5518');", "showInfo('AF72BB94A5536336467DB7D
49D5210B3');"]
https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=B3460D59D0BF065AADE72E239964574F
https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=61473F823BB6A9F954DEDFF020D85111
https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=17D551A108AA6EDB56633768FCD9310F
https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=37D0509D06E0F6243DE88A54351426DE
https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=C4AB705DA0063542548212CD76070D2D
https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=1271E1730AF7C01110E54AE5F77A18D0
https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=CCC8109C92E1B03B92A18F8B87B14C5D
https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=31ECB92BAE10BBA109E2FA227D68B81D
https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=559E267F161240C4E953889C802A5518
https://openstd.samr.gov.cn/bzgk/gb/newGbInfo?hcno=AF72BB94A5536336467DB7D49D5210B3
获取完毕共10个
正在下载...
['标准号： GB 16798-2023 ']
['标准号： GB 16994.4-2023 ']
['标准号： GB 29753-2023 ']
['标准号： GB 43069-2023 ']
['标准号： GB 43068-2023 ']
['标准号： GB 43066-2023 ']
['标准号： GB 43067-2023 ']
['标准号： GB 7957-2023 ']
['标准号： GB 43203-2023 ']
['标准号： GB 43049-2023 ']
有关‘安全’的国家标准下载完毕
```

```
(base) nanmener@Haotians-MacBook-Pro 10. Spider % cat 安全.csv
序号,标准编号,标准名称,英文名称,状态,中国标准分类号（CCS）,国际标准分类号（ICS）,发布日期,实施日期,主管部门,归口单位,发布单位,备注
1,GB 16798-2023,食品机械安全要求,Food machinery safety requirements,即将实施,X90,67.260,2023-09-08,2024-04-01,工业和信息化部,工业和信息化部,国家市场监督管理总局、国家标准化管理委员会,None
2,GB 16994.4-2023,港口作业安全要求　第4部分：普通货物集装箱,Safety requirements for port operation—Part 4: General cargo container,即将实施,R43,03.220.40,2023-09-08,2024-04-01,交通运输部,交通运输部,国家市场监督管理总局、国家标准化管理委员会,None
3,GB 29753-2023,道路运输 易腐食品与生物制品 冷藏车安全要求及试验方法,Road transportation—Perishable foodstuffs and biological products—Safety requirement and test methods of refrigerated vehicle,即将实施,T54,43.160,2023-09-08,2024-01-01,工业和信息化部,工业和信息化部,国家市场监督管理总局、国家标准化管理委员会,None
4,GB 43069-2023,矿用电缆安全技术要求,Safety requirements for  mine cable,即将实施,K13,29.060.20,2023-09-08,2025-04-01,国家矿山安全监察局,国家矿山安全监察局,国家市场监督管理总局、国家标准化管理委员会,None
5,GB 43068-2023,煤矿用跑车防护装置安全技术要求,Safety technical requirements for coal mine-used car catcher,即将实施,D18,73.100.40,2023-09-08,2024-04-01,国家矿山安全监察局,国家矿山安全监察局,国家市场监督管理总局、国家标准化管理委员会,None
6,GB 43066-2023,煤矿用被动式隔爆设施安全技术要求,Safety technique requirements for  passive flameproof facilities of  coal mine,即将实施,D09,13.100,2023-09-08,2024-04-01,国家矿山安全监察局,国家矿山安全监察局,国家市场监督管理总局、国家标准化管理委员会,None
7,GB 43067-2023,煤矿用仪器仪表安全技术要求,Technical requirements for safety of environmental monitoring instruments for coalmines,即将实施,N21,13.100,2023-09-08,2024-04-01,国家矿山安全监察局,国家矿山安全监察局,国家市场监督管理总局、国家标准化管理委员会,None
8,GB 7957-2023,煤矿用矿灯安全技术要求,Safety technical requirement for cap lamps in coal mine,即将实施,K35,29.260.20,2023-09-08,2025-04-01,国家矿山安全监察局,国家矿山安全监察局,国家市场监督管理总局、国家标准化管理委员会,None
9,GB 43203-2023,选煤厂安全规程,Code of practice for coal preparation plant safety,即将实施,D16,73.120,2023-09-08,2024-07-01,国家矿山安全监察局,国家矿山安全监察局,国家市场监督管理总局、国家标准化管理委员会,None
10,GB 43049-2023,连铸机安全技术条件,Safety technical conditions for continuous casting machines,即将实施,H93,77.180,2023-09-08,2024-10-01,工业和信息化部,工业和信息化部,国家市场监督管理总局、国家标准化管理委员会,None
```

§2　实验题 2

选择一个在线视频网站，使用 selenium 编写一个爬虫，进行视频下载。

§2.1　解答

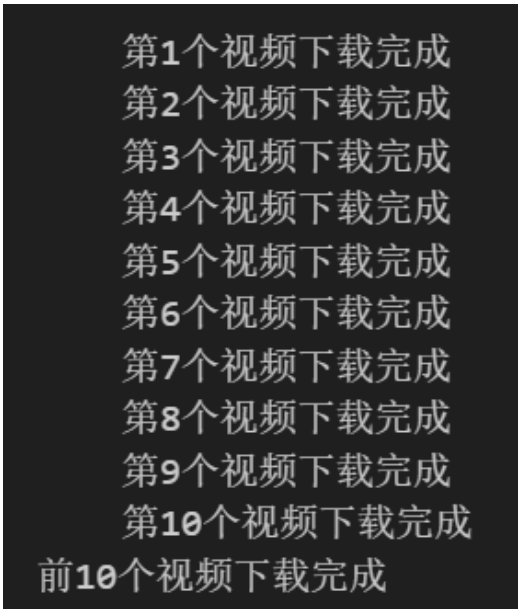
2.1.1　代码

```python
import requests
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from win32api import Sleep

chromedriver_path = r"C:\Program Files\Google\Chrome\Application\chromedriver.exe"
service = Service(chromedriver_path)
driver = webdriver.Chrome(service=service)

driver.get('https://v.jstv.com/xw360/')
Sleep(2000)
driver.find_element(By.XPATH, "//*[@id=\"J_bodyBd\"]/div/div/ul/li[1]/div[1]/a/img").click()
Sleep(2000)
driver.find_element(By.XPATH, "//*[@id=\"J_bodyBd\"]/div/div/ul/li[2]/div[1]/a/img").click()
Sleep(3000)
driver.find_element(By.XPATH, "//*[@id=\"J_bodyBd\"]/div/div/ul/li[3]/div[1]/a/img").click()
Sleep(3000)
driver.find_element(By.XPATH, "//*[@id=\"J_bodyBd\"]/div/div/ul/li[4]/div[1]/a/img").click()
```

```
Sleep(3000)
driver.find_element(By.XPATH,
    "//*[@id=\"J_bodyBd\"]/div/div/ul/li[5]/div[1]/a/img").click()
Sleep(3000)
driver.find_element(By.XPATH,
    "//*[@id=\"J_bodyBd\"]/div/div/ul/li[6]/div[1]/a/img").click()
Sleep(3000)
driver.find_element(By.XPATH,
    "//*[@id=\"J_bodyBd\"]/div/div/ul/li[7]/div[1]/a/img").click()
Sleep(3000)
driver.find_element(By.XPATH,
    "//*[@id=\"J_bodyBd\"]/div/div/ul/li[8]/div[1]/a/img").click()
Sleep(3000)
driver.find_element(By.XPATH,
    "//*[@id=\"J_bodyBd\"]/div/div/ul/li[9]/div[1]/a/img").click()
Sleep(3000)
driver.find_element(By.XPATH,
    "//*[@id=\"J_bodyBd\"]/div/div/ul/li[10]/div[1]/a/img").click()

Sleep(3000)
windows = driver.window_handles  # 获取当前所有页面句柄
for i in range(1, len(windows)):
    driver.switch_to.window(windows[i])  # 切换当新页面
    Sleep(3000)

    # print(driver.title)

    driver.find_element(By.XPATH, "//*[@id=\"videoxg\"]/xg-start").click()
    Sleep(3000)
    video_element = driver.find_element(By.TAG_NAME, 'video')
    video_url = video_element.get_attribute('src')
    # 使用Requests库下载视频
    response = requests.get(video_url)

    # 指定视频文件的保存路径
    save_path = str(driver.title)+'.mp4'

    # 将视频内容写入文件
    with open(save_path, 'wb') as file:
        file.write(response.content)
    # print(driver.title)
    driver.close()
    print(f"    第{i}个视频下载完成")
    driver.switch_to.window(windows[0])
```

```
    Sleep(3000)
driver.quit()
print("前10个视频下载完成")
```

2.1.2 截图



§3 总结

§3.1 分工情况

以下说明小组的分工情况，每个人完成了什么功能，完成的情况如何

陈昊天

负责报告的整合撰写；

负责实验题 1 的研究与解答；

陈佳伟

负责实验题 2 的研究与解答；

冯佳钧

解读实验题 1 的课件代码；

§3.2 解决方案评价

以下说明小组对解决方案的评价，是否完成了题目的要求，怎么证明完成了题目的要求，好的地方在哪里，哪里有不足。

实验题 1

完成了题目的要求，修改课件中的下载安全标准信息的爬虫，使之能够正常工作。

优点：忽略 SSL 证书验证，使 TLS 握手成功。

缺点：对于下载失败的情况没有做重试处理。

实验题 2

完成了题目的要求，能够使用 selenium 从一个在线视频网站下载视频。

优点：使用 selenium 模拟真人操作浏览器，并使用 Sleep() 函数模拟操作间隔，最终下载到视频文件，使网站服务方无法察觉爬虫的存在。

缺点：只下载了网站的前 10 个视频，没有对网页列出的所有视频进行下载。

§3.3 个人心得

收获与不足的个人总结

陈昊天

收获：学习了 TLS/SSL 相关知识，了解跳过证书验证的利弊

不足：在处理下载失败的情况时，缺乏相应的异常处理机制和重试策略。

陈佳伟

收获：实践 selenium 功能和代码实现，初步使用 chromedriver，熟悉了自动化测试工具的实际应用。

不足：未能全面覆盖网站上所有视频内容的下载。

冯佳钧

收获：学习浏览器 UA、request 请求相关知识，理解课件代码

不足：对网络爬虫工作原理的理解还不够深刻。

参考文献

[1] Wikipedia contributors. Transport layer security — Wikipedia, the free encyclopedia, 2023. [Online; accessed 28-November-2023].