9 web scrap

9.1 Crawling the Web – wget

Basic usage

`wget` 是一个在命令行中使用的下载工具，用于从 Web 上下载文件。它支持 HTTP、HTTPS 和 FTP 协议，具有许多功能，如断点续传、递归下载、后台下载等。你可以使用类似以下的命令来下载一个文件：

```Bash
wget [URL]
```

例如，要下载一个名为 `example.zip` 的文件，可以使用以下命令：

```Bash
wget https://example.com/example.zip
```

你还可以通过添加参数来控制下载的行为，比如 `-O` 参数可以指定保存的文件名：

```Bash
wget -O filename.zip https://example.com/example.zip
```

Page requisites

对于使用 `wget` 下载网页及其相关资源，你可以使用 `-p` 选项。此选项将下载页面的所有必需文件，包括样式表、图像等。

```Bash
wget -p https://example.com
```

这将下载 `example.com` 网页及其所有必需的资源。

至于 `robots.txt` 文件，`wget` 默认情况下不会下载该文件，因为它是一个针对爬虫的指令文件，通常用于指示搜索引擎爬虫哪些页面可以被抓取。如果你想下载 `robots.txt` 文件，可以使用 `-e robots=off` 选项来禁用 `robots.txt` 文件的处理：

```bash
wget -p -e robots=off https://example.com
```

这样就会下载页面及其相关资源，同时忽略 `robots.txt` 文件。

`robots.txt` 是一个用于网站爬虫控制的标准文件。它是一个简单的文本文件，通常存储在网站的顶级目录中。`robots.txt` 文件用于指定哪些网络爬虫（user-agents）被允许访问网站的哪些资源。

通过编辑 `robots.txt` 文件，网站所有者可以控制搜索引擎爬虫对其网站的访问行为。例如，网站所有者可以通过 `robots.txt` 文件阻止搜索引擎爬虫访问某些敏感页面或限制它们的访问频率，以避免不必要的流量或保护特定内容。

爬虫在访问网站时通常会首先查找 `robots.txt` 文件，以了解网站所有者对爬取行为的偏好。

Recursive downloading

递归下载是指下载网页及其链接的页面。使用 `wget` 命令进行递归下载时，您可以使用以下参数：

- `-r`：表示进行递归下载。
- `-l N`：指定递归的深度级别，其中 N 代表允许递归的级别。默认情况下，如果省略该参数，默认递归级别为 5。

举例来说，如果您想要下载指定 URL 的网页及其链接页面，可以使用以下命令：

```bash
wget -r -l 2 <url>
```

这将递归地下载指定 URL 的网页及其链接页面，递归深度为 2 级。

Mirroring

要下载整个网站及其所有内容，您可以使用以下命令：

```bash
wget -m -w 1 <url>
```

这将使用标准默认设置下载整个网站，并在每个请求之间等待 1 秒以避免对服务器造成干扰。

这里的 `-m` 和 `-w` 是 `wget` 命令的选项：

- `-m` 表示递归下载，它会下载指定页面及其链接的所有页面和资源，构建一个本地副本。
- `-w` 后面跟着一个数字，表示等待时间（以秒为单位），用于设置每个请求之间的等待时间。这有助于避免对服务器造成过多负载，也可以用来控制下载速度。

当使用 wget 命令时，可以结合不同的选项来实现各种功能。以下是一些常用选项及其作用的示例：

1. 下载单个文件：

   ```bash
   wget http://example.com/file.txt
   ```

2. 下载整个网站：

   ```bash
   wget -m http://example.com
   ```

3. 递归下载到指定深度：

   ```bash
   wget -r -l 2 http://example.com
   ```

4. 下载文件并保留远程目录结构：

   ```bash
   wget -r --no-parent http://example.com/path/to/files
   ```

5. 下载文件并限制带宽：

```Bash
wget --limit-rate=100k http://example.com/file.zip
```

6. 使用用户代理标识：

```Bash
wget --user-agent="Mozilla/5.0" http://example.com
```

7. 下载文件并保存到指定位置：

```Bash
wget -O output.zip http://example.com/file.zip
```

8. 断点续传下载：

```Bash
wget -c http://example.com/largefile.zip
```

9. 后台下载：

```Bash
wget -b http://example.com/largefile.zip
```

这些选项可以根据具体需求进行组合和调整，以实现所需的下载功能。

9.2 BeautifulSoup

Requirements

```Python
Python requirements:
$ sudo apk add python3 py3-pip
BeautifulSoup itself:
$ pip install bs4
Test using interpreter.
```

Loading a page

Python

```
>>> file = "cattax/index.html"
>>> soup = BeautifulSoup(open(file, 'r'))
```

Printing page text

Python

```
 text = soup.get_text()
>>> print(text)
```

Navigating page elements

Python

```
>>> soup.title

>>> soup.body.main
```

Finding page elements

Python

```
# 查找第一个匹配项
soup.find('strong')

# 查找所有匹配项
soup.find_all('strong')
```

Exercise 1

CSS

```
wget localhost:8080/index.html
wget -p localhost:8080/index.html
wget -r -l 1 localhost:8080/index.html
```

```
wget -r -l 2 localhost:8080/index.html
rm -r localhost:8080
wget -m localhost:8080/index.html
```

这些命令都是用于在命令行中从 Web 服务器下载文件或整个网站的工具，常用于 Linux 系统。以下是每个命令的解释：

1. `wget localhost:8080/index.html`：这个命令将从 `http://localhost:8080/index.html` 下载单个文件 `index.html` 并保存在当前目录下。
2. `wget -p localhost:8080/index.html`：这个命令将下载 `http://localhost:8080/index.html` 页面，并同时下载页面中引用的所有相关资源（如样式表、脚本和图像），并保存在当前目录下。
3. `wget -r -l 1 localhost:8080/index.html`：这个命令将递归地下载 `http://localhost:8080/index.html` 页面中链接的所有文件，但仅限于当前页面的链接（深度为 1），并保存在当前目录下。
4. `wget -r -l 2 localhost:8080/index.html`：这个命令将递归地下载 `http://localhost:8080/index.html` 页面中链接的所有文件，包括当前页面的链接以及它链接的页面（深度为 2），并保存在当前目录下。
5. `rm -r localhost:8080`：这个命令尝试删除名为 `localhost:8080` 的文件或目录。然而，在这里使用 `rm` 命令删除服务器的方式并不合适，因为 `rm` 命令用于删除本地文件或目录。
6. `wget -m localhost:8080/index.html`：这个命令将递归地下载 `http://localhost:8080/index.html` 页面中链接的所有文件，并保存在当前目录下。这个命令类似于前面的 `-r` 选项，但 `wget -m` 通常会更智能地处理连接，以便下载整个网站而不是下载无限数量的页面。

1. **阅读 `man wget` 以了解 `-i` `--force-html` 和 `--spider` 选项的作用。下载此网页的副本（您当前正在阅读的网页）并用于 `wget` 测试页面上的所有链接。是否有任何损坏的链接?**

这些选项是 `wget` 命令的一部分，用于下载文件或网站。下面是它们的作用：

1. `-i` 选项：指定包含要下载的 URL 列表的文件。使用 `-i` 选项，你可以在一个文件中列出多个 URL，然后 `wget` 将逐个下载这些 URL。示例：

   ```bash
   wget -i urls.txt
   ```

   在这个示例中，`wget` 将从 `urls.txt` 文件中读取 URL 列表，并依次下载其中的每个 URL。

2. `--force-html` 选项：强制 `wget` 将下载的文件当作 HTML 文件处理。有时，服务器返回的文件的 MIME 类型可能不正确，这可能导致 `wget` 不会正确处理文件。使用 `--force-html` 选项，你可以告诉 `wget` 将下载的文件当作 HTML 文件来处理，无论其实际的 MIME 类型是什么。

   ```bash
   wget --force-html http://example.com/index
   ```

   在这个示例中，`wget` 将下载 `http://example.com/index` 页面，并将其当作 HTML 文件处理。

3. `--spider` 选项：不下载任何文件，而只检查链接是否有效。使用 `--spider` 选项，`wget` 将只发送 HEAD 请求，而不会下载任何文件。这个选项通常用于检查链接的有效性，而不实际下载文件。

   ```bash
   wget --spider http://example.com/index.html
   ```

   在这个示例中，`wget` 将向 `http://example.com/index.html` 发送 HEAD 请求，以检查链接的有效性，但不会下载任何文件。

这些选项可以根据需要单独使用，也可以与其他 `wget` 选项一起使用。

1. **告诉 `wget` 在对本地主机上运行的服务器的请求中使用不同的用户代理字符串。检查您的服务器的请求是什么样的。**

要在 `wget` 命令中使用不同的用户代理字符串，你可以使用 `-U` 或 `--user-agent` 选项。这个选项允许你指定自定义的用户代理字符串，以替换 `wget` 默认的用户代理字符串。示例如下：

```bash
wget -U "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
```

在这个示例中，`-U` 选项后面跟着自定义的用户代理字符串。`wget` 将使用指定的用户代理字符串发送请求，而不是默认的用户代理字符串。

要检查服务器收到的请求，你可以查看服务器的访问日志。通常情况下，服务器会记录每个收到的请求，包括请求的 URL、用户代理字符串、时间戳等信息。你可以查看服务器的访问日志文件，以了解 `wget` 发送的请求的确切细节。具体的日志文件位置和格式取决于服务器的配置和使用的 Web 服务器软件（如 Apache、Nginx 等）。

1. **与有何 `wget -r -l 1 http://example.com` 不同 `wget -p http://example.com`？*（提示：考虑外部资源）*。**

`wget -r -l 1 http://example.com` 和 `wget -p http://example.com` 都是使用 wget 命令来下载指定网站的内容，但它们之间有一些区别，尤其涉及外部资源的下载：

1. `wget -r -l 1 http://example.com`：
    - `-r` 选项表示递归下载，它会下载指定网站的所有链接，并且继续递归地下载链接中的链接，以此类推，直到达到指定的递归深度。
    - `-l 1` 选项指定了递归深度为 1，即只下载指定网站的直接链接，不再递归下载链接中的链接。这样做可以控制下载的范围，避免下载整个网站的内容。
2. `wget -p http://example.com`：
    - `-p` 选项表示下载页面所需的所有文件，包括 HTML、CSS、JavaScript、图片等，但不会递归下载其他页面。它会下载页面上引用的所有外部资源，以确保页面能够正确显示。

因此，主要区别在于 `-r` 选项会递归下载整个网站的链接，而 `-p` 选项只会下载指定页面及其相关的外部资源，不会递归下载其他页面。

1. **在联机帮助页中查找“递归接受/拒绝选项” `wget`。您将如何 `wget` 从多个不同域抓取页面?**

在联机帮助页中查找“递归接受/拒绝选项”可以使用 `man wget` 命令来查找，或者在终端中运行 `wget --help`，然后搜索相关内容。

要使用 wget 从多个不同域抓取页面，您可以使用 `-H` 选项（`--span-hosts`），这将允许 wget 递归地下载与原始 URL 的主机不同的链接。例如：

bash

Copy code

```
wget -r -H http://example1.com
```

这将下载 `http://example1.com` 页面上链接的所有页面，以及这些页面链接的其他主机上的页面。

1. **查一下有什么 `-nc` 作用什么是破坏，为什么你愿意或不愿意这样做?**

`-nc` 选项在 wget 中用于禁止覆盖已经存在的文件。它的全称是 `--no-clobber`。当使用该选项时，如果下载的文件已经存在于目标位置，则不会重新下载，而是保留现有的文件。

这个选项的破坏性在于，如果您需要确保下载的文件是最新的，那么使用该选项可能会导致您得到一个过时的文件，因为 wget 不会重新下载已存在的文件。

您可能会愿意使用 `-nc` 选项的情况是，如果您只对本地文件系统中不存在的文件感兴趣，或者您已经确信下载的文件是最新的，并且不希望重新下载它们。

您可能不愿意使用 `-nc` 选项的情况是，如果您需要确保下载的文件是最新的，或者您希望覆盖现有文件以获取更新的内容。

Exercise 2

美丽汤

```CSS
from bs4 import BeautifulSoup 如果此操作完成，则您已成功导入该库。

file = "cattax/index.html"
soup = BeautifulSoup(open(file, 'r'))
```

现在您有了一个“soup”对象，它是一个 Python 对象，具有多种与 HTML（和 XML）文档交互并访问其内容的方法。首先，只需输入 `soup` 您的解释器即可。Python 将打印出对象的基本文本表示，这只是 `index.html` 页面的源代码。接下来，让我们尝试最常见的抓取用例之一。

```css
soup.get_text()
```

您应该看到一个包含网页文本内容的字符串。如果您调用 `print` 结果，您应该会看到文本的布局与可见文本在网页上的显示方式非常接近。

```css
text = soup.get_text()
print(text)
```

例如，如果您要使用网页作为 NLP 系统的输入，则从页面中获取可见文本是一个常见的要求。您还可以访问页面的不可见部分——`soup.title` 将为您提供网页的标题元素，并 `soup.title.text` 为您提供标题元素中的文本内容。请注意区别：`soup.title` 是一个 BeautifulSoup 元素（类型为 `Tag`），并且具有与标签相关的方法；`soup.title.text` 只是一个字符串，并且只有适用于字符串的方法。

由于 HTML 文档是结构化的，因此您可以以结构化的方式与它们交互。`soup.head` 将为您提供一个反映 HTML 结构的“head”部分的 soup 对象，并将 `soup.head.findChildren()` 返回一个包含 head 内所有“子”元素的列表。通过了解文档的结构，您可以通过编程方式导航到某些元素。这不仅仅与标签相关：您还可以访问属性的值。`soup.head.meta['charset']` 将访问 `charset` 文档头部元标记的属性。

1. **查看 BeautifulSoup 文档中的一些其他示例，特别是有关该 `.find_all()` 方法的使用的示例。**

在 Beautiful Soup 文档中，`.find_all()` 方法用于查找文档中符合特定条件的所有标签。这个方法可以接受一系列参数，用于指定搜索的条件，例如标签名、属性、文本内容等。以下是一些关于 `.find_all()` 方法的示例用法：

1. **根据标签名查找所有标签：**

```python
soup.find_all('a')  # 查找所有<a>标签
```

2. **根据 class 属性查找所有标签：**

```python
soup.find_all(class_='item')  # 查找所有class属性为"item"的标签
```

3. **根据 id 属性查找所有标签：**

```python
soup.find_all(id='main-content')  # 查找所有id属性为"main-content"的标
```

4. **根据多个条件查找标签：**

```python
soup.find_all('a', class_='item', id='link-1')  # 查找所有<a>标签，cl
```

5. **查找文档中的所有链接：**

```python
for link in soup.find_all('a'):
    print(link.get('href'))
```

6. **使用正则表达式查找匹配的标签：**

```python
import re
soup.find_all(href=re.compile(r'^https://'))  # 查找href属性以"https
```

以上是一些`.find_all()`方法的常见用法示例，您可以根据自己的需求灵活运用这些方法来搜索和提取 HTML 文档中的标签信息。

1. **使用解释器访问`<strong>`网页中所有元素的列表，并弄清楚如何仅打印其中包含的文本。**

要访问网页中所有元素的列表，您可以使用 Beautiful Soup 库来解析 HTML 文档并提取其中的元素。以下是一个使用解释器的示例，演示如何获取网页中所有元素的列表，并打印出其中包含的文本：

```python
from bs4 import BeautifulSoup

# 假设html是您要解析的HTML文档的字符串形式
html = """
<html>
<head><title>示例</title></head>
<body>
<div id="content">
    <h1>这是标题</h1>
    <p>这是第一段文字。</p>
    <p>这是第二段文字。</p>
    <ul>
        <li>列表项1</li>
        <li>列表项2</li>
    </ul>
</div>
</body>
</html>
"""

# 创建Beautiful Soup对象并指定解析器
soup = BeautifulSoup(html, 'html.parser')

# 获取网页中所有元素的列表
all_elements = soup.find_all()

# 打印所有元素中包含的文本
for element in all_elements:
    text = element.get_text()
    if text.strip():  # 如果文本不为空
        print(text.strip())
```

这段代码首先创建了一个 Beautiful Soup 对象，然后使用 `find_all()` 方法获取网页中所有的元素。接着遍历所有元素，使用 `get_text()` 方法获取每个元素中的文本内容，并使用 `strip()` 方法去除文本前后的空白字符。最后，将非空文本打印出来。

1. **您将如何 `.find_all()` 查找 `<div>` 具有特定类值（例如“容器”）的所有元素？如果 div 有多个类，你的方法会起作用吗？**

您可以使用 `find_all()` 方法来查找具有特定类值的所有 `<div>` 元素。如果 `<div>` 有多个类，您可以使用 `class_` 参数来指定要查找的类值。

以下是一个示例代码，演示如何查找具有特定类值（例如“container”）的所有 `<div>` 元素，以及处理具有多个类的情况：

Python

```
from bs4 import BeautifulSoup

# 假设html是您要解析的HTML文档的字符串形式
html = """
<html>
<head><title>示例</title></head>
<body>
<div class="container">第一个容器</div>
<div class="container">第二个容器</div>
<div class="container sidebar">第三个容器（具有多个类）</div>
<div class="content">普通内容</div>
</body>
</html>
"""

# 创建Beautiful Soup对象并指定解析器
soup = BeautifulSoup(html, 'html.parser')

# 查找具有特定类值的所有<div>元素
divs_with_class = soup.find_all('div', class_='container')

# 打印找到的元素
for div in divs_with_class:
    print(div.get_text())

# 如果<div>具有多个类，则使用部分匹配查找（也可以使用完整的类名）
divs_with_multiple_classes = soup.find_all('div', class_='sidebar')

# 打印具有多个类的<div>元素
for div in divs_with_multiple_classes:
    print(div.get_text())
```

在这个示例中，我们首先使用 `find_all()` 方法查找具有类值“container”的所有 `<div>` 元素。然后，我们使用 `find_all()` 方法查找具有类值“sidebar”的 `<div>` 元素，注意到它有多个类。

抓取脚本

```CSS
from bs4 import BeautifulSoup
import os


for file in os.listdir('cattax'):
  if file[-4:] == 'html':
    soup = BeautifulSoup(open('cattax/'+file,'r'), features='html.pars
    print(soup.title.text + " : " + soup.h1.text)
```

将此 python 脚本下载到包含您的文件夹的目录 `cattax` （_而不是_其本身 `cattax` ）。在命令行上，您应该能够使用 `python3 scrape.py` .您将看到它打印出与 中所有文件相关的一系列行 `cattax` 。

在编辑器中打开 `scrape.py` 并检查它正在做什么。该脚本导入了两个库，一个是 BeautifulSoup，另一个是 `os` ，它允许脚本使用某些操作系统功能。`os.listdir` 然后用于列出 `cattax` 目录的内容并迭代它们。我们通过检查哪些文件名以字符串“html”结尾来过滤文件名，如果是，则我们打开文件并将其解析为 BeautifulSoup 对象。然后，我们打印某些元素中的文本，并用“:”分隔。了解这是如何运作的，如果您不确定，请向助教或讲师寻求帮助。

练习

1. **进行修改 `scrape.py` ，使其_还_打印出每页中“info”段落的内容（这可以是第二个打印语句）。再次运行该脚本以测试其是否有效。**

```CSS
from bs4 import BeautifulSoup
import os

for file in os.listdir('cattax'):
    if file[-5:] == '.html':
```

```
        soup = BeautifulSoup(open('cattax/' + file, 'r', encoding='utf
        print(soup.title.text + " : " + soup.h1.text)

        # 打印每页中“info”段落的内容
        info_paragraph = soup.find('p', class_='info')
        if info_paragraph:
            print("Info: " + info_paragraph.text.strip())
        else:
            print("Info not found")
```

1. **目前该脚本会为_每一_页打印一些内容。修改它，以便它只为叶节点打印一些内容——那些没有自己的“容器”元素的页面。**

```css
from bs4 import BeautifulSoup
import os

def is_leaf_node(soup):
    # 检查页面是否包含链接或带有特定类的容器元素
    if soup.find('a') or soup.find('div', class_='container'):
        return False
    return True

for file in os.listdir('cattax'):
    if file.endswith('.html'):
        with open('cattax/' + file, 'r', encoding='utf-8') as f:
            soup = BeautifulSoup(f, 'html.parser')
            if is_leaf_node(soup):
                print(soup.title.text + " : " + soup.h1.text)

                # 查找并打印每页中“info”段落的内容
                info_paragraph = soup.find('p', class_='info')
                if info_paragraph:
                    print("Info: " + info_paragraph.text.strip())
                else:
                    print("Info not found")
```

1. **打印出来的东西可能很有用，但我们通常希望存储为以后的编程工作抓取的值。不是打印出信息，而是为所有叶节点创建并更新一个 Python 字典，其中字典键是页面标题，值是“信息”框的相应内容。*运行您的脚本*** `python3 -i`

`scrape.py`，它将执行您的脚本，然后在脚本执行后立即将您置于交互式会话中。然后，您可以通过与解释器中的 dict 对象交互来检查字典的内容是否符合您的预期。

```CSS
from bs4 import BeautifulSoup
import os

page_info = {}

def is_leaf_node(soup):
    # 检查页面是否包含链接或带有特定类的容器元素
    if soup.find('a') or soup.find('div', class_='container'):
        return False
    return True

for file in os.listdir('cattax'):
    if file.endswith('.html'):
        with open('cattax/' + file, 'r', encoding='utf-8') as f:
            soup = BeautifulSoup(f, 'html.parser')
            if is_leaf_node(soup):
                title = soup.title.text
                h1_text = soup.h1.text
                info_paragraph = soup.find('p', class_='info')
                if info_paragraph:
                    info_text = info_paragraph.text.strip()
                else:
                    info_text = "Info not found"

                page_info[title] = {'Title': title, 'H1': h1_text, 'In

# 进入交互式会话
if __name__ == "__main__":
    import code
    code.interact(local={**globals(), **locals()})
```