Web scraping code of conduct

This all being said, if you adhere to the following simple rules, you will probably be fine.

1. Ask nicely. If your project requires data from a particular organisation, for example, you can try asking them directly if they could provide you what you are looking for. With some luck, they will have the primary data that they used on their website in a structured format, saving you the trouble.
2. Don't download copies of documents that are clearly not public. For example, academic journal publishers often have very strict rules about what you can and what you cannot do with their databases. Mass downloading article PDFs is probably prohibited and can put you (or at the very least your friendly university librarian) in trouble. If your project requires local copies of documents (e.g. for text mining projects), special agreements can be reached with the publisher. The library is a good place to start investigating something like that.
3. Check your local legislation. For example, certain countries have laws protecting personal information such as email addresses and phone numbers. Scraping such information, even from publicly available web sites, can be illegal (e.g. in Australia).
4. Don't share downloaded content illegally. Scraping for personal purposes is usually OK, even if it is copyrighted information, as it could fall under the fair use provision of the intellectual property legislation. However, sharing data for which you don’t hold the right to share is illegal.
5. Share what you can. If the data you scraped is in the public domain or you got permission to share it, then put it out there for other people to reuse it (e.g. on datahub.io). If you wrote a web scraper to access it, share its code (e.g. on GitHub) so that others can benefit from it.
6. Don't break the Internet. Not all web sites are designed to withstand thousands of requests per second. If you are writing a recursive scraper (i.e. that follows hyperlinks), test it on a smaller dataset first to make sure it does what it is supposed to do. Adjust the settings of your scraper to allow for a delay between requests. By default, Scrapy uses conservative settings that should minimize this risk.
7. Publish your own data in a reusable way. Don’t force others to write their own scrapers to get at your data. Use open and software-agnostic formats (e.g. JSON, XML), provide metadata (data about your data: where it came from, what it represents, how to use it, etc.) and make sure it can be indexed by search engines so that people can find it.
8. View robots.txt file. Robots.txt is a file used by websites to let 'bots' know if or how the site should be crawled and indexed. When you are trying to extract data from the web, it is critical to understand what robots.txt is and how to respect it to avoid legal ramifications. This file can be accessed for any domain by accessing <domain_url>/robots.txt. For eg: monash.edu/robots.txt, facebook.com/robots.txt, linkedin.com/robots.txt.

Happy scraping!

References

- The Web scraping Wikipedia page has a concise definition of many concepts discussed here.
- This case study is a great example of what can be done using web scraping and a stepping stone to a more advanced python library scrapy.
- This recent case about Linkedin data is a good read.
- A crisp and simple explanation to robots.txt can be found here.
- Commencing 25 May 2018, Monash University will also become subject to the European Union’s General Data Protection Regulation (GDPR).
- Software Carpentry is a non-profit organisation that runs learn-to-code workshops worldwide. All lessons are publicly available and can be followed indepentently. This lesson is heavily inspired by Software Carpentry.
- Data Carpentry is a sister organisation of Software Carpentry focused on the fundamental data management skills required to conduct research.
- Library Carpentry is another Software Carpentry spinoff focused on software skills for librarians.