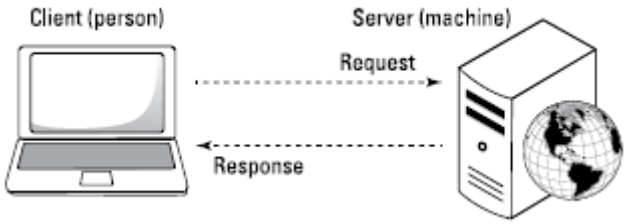
**Interakcja z Internetem**

Jak zapewne wiesz, Internet jest domem dla praktycznie całej światowej wiedzy. Większość z nas cały czas korzysta z sieci World Wide Web (znanej również jako sieć internetowa), aby znaleźć informacje. Robimy to za pomocą przeglądarki internetowej, takiej jak Safari, Google Chrome, Firefox, Opera, Internet Explorer lub Edge. Aby odwiedzić witrynę internetową, wpisz adres URL (jednolity lokalizator zasobów) w pasku adresu przeglądarki i naciśnij klawisz Enter lub kliknij łącze, które automatycznie przekieruje Cię na tę stronę. Alternatywą dla przeglądania Internetu za pomocą przeglądarki internetowej jest programowy dostęp do jego zawartości. Innymi słowy, możesz używać języka programowania, takiego jak Python, do publikowania informacji w Internecie, a także do uzyskiwania dostępu do informacji w Internecie. W pewnym sensie sprawiasz, że Internet staje się Twoją osobistą bazą wiedzy, z której Twoje aplikacje mogą dowolnie pobierać informacje. W tej części poznasz dwa główne moduły umożliwiające programowy dostęp do sieci w Pythonie: urllib i Beautiful Soup.

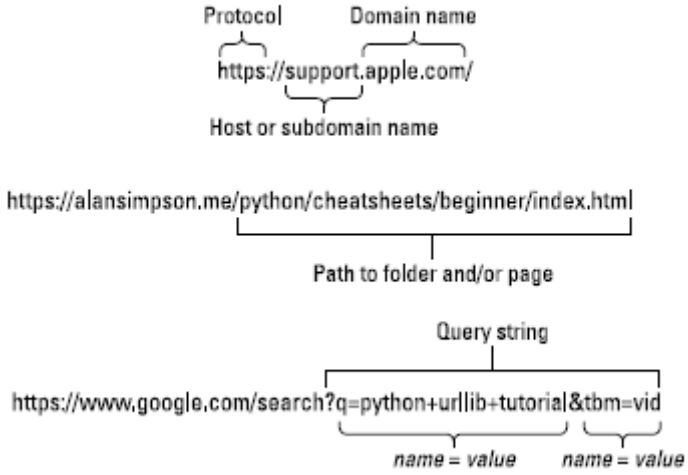
**Jak działa sieć**

Kiedy otwierasz przeglądarkę internetową i wpisujesz adres URL lub klikasz łącze, ta czynność wysyła żądanie do Internetu. Internet kieruje twoje żądanie do odpowiedniego serwera WWW, który z kolei wysyła odpowiedź z powrotem do twojego komputera. Zazwyczaj odpowiedzią jest strona internetowa, ale może to być dowolny plik. Lub może to być komunikat o błędzie, jeśli rzecz, o którą prosiłeś, nie istnieje już w tej lokalizacji. Ale ważną rzeczą jest to, że ty, użytkownik (istota ludzka) i twój agent użytkownika (program, którego używasz do uzyskiwania dostępu do Internetu) jesteście po stronie klienta. Serwer, który jest tylko komputerem, a nie osobą, odsyła swoją odpowiedź, jak pokazano na rysunku




**Zrozumienie tajemniczego adresu URL**

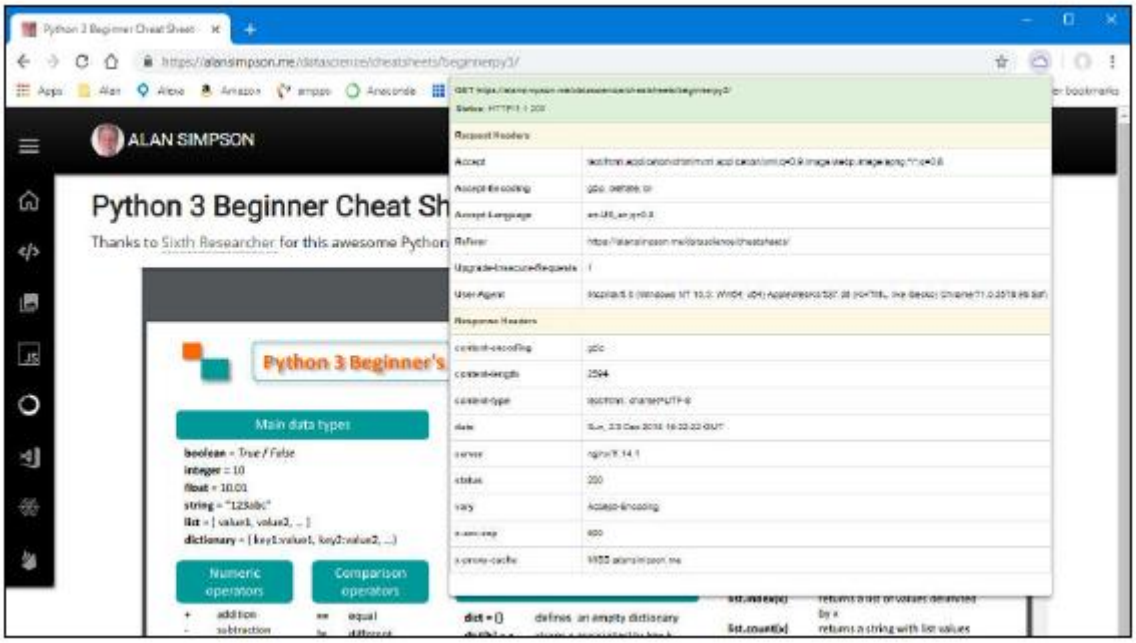
Adres URL jest kluczowym elementem całej transakcji, ponieważ w ten sposób Internet znajduje poszukiwany zasób. W sieci wszystkie zasoby korzystają z protokołu HTTP (Hypertext Transfer Protocol), dlatego ich adresy URL zaczynają się od http:// lub https://. Różnica polega na tym, że http:// wysyła użytkownika przez sieć w jego surowej postaci, co czyni go podatnym na ataki hakerów i innych osób, które mogą „wywąchać" ruch. Protokół https jest bezpieczny, ponieważ dane są szyfrowane, co oznacza, że zostały przekonwertowane na tajny kod, który nie jest tak łatwy do odczytania. Zazwyczaj każda witryna, z którą prowadzisz interesy i której przekazujesz poufne informacje, takie jak hasła i numery kart kredytowych, korzysta z protokołu https, aby zapewnić poufność i bezpieczeństwo tych informacji. Adres URL dowolnej witryny może być stosunkowo prosty, na przykład adres URL Alana https://AlanSimpson.me. Lub dodanie większej ilości informacji do żądania może być skomplikowane. Rysunek pokazuje części adresu URL, z których część mogłaś zauważyć w przeszłości.




Pamiętaj, że kolejność ma znaczenie. Na przykład adres URL może zawierać ścieżkę do określonego folderu lub strony (zaczynając się ukośnikiem zaraz po nazwie domeny). Adres URL może również zawierać ciąg zapytania, który jest zawsze ostatni i zawsze zaczyna się od znaku zapytania (?). Po znaku zapytania pojawia się jedna lub więcej par nazwa=wartość, w zasadzie ta sama składnia, którą widziałeś w słownikach danych i JSON. Jeśli istnieje wiele par nazwa=wartość, są one oddzielone ampersandami. Znak #, po którym następuje nazwa po nazwie strony na końcu adresu URL, nazywany jest fragmentem, który wskazuje określone miejsce na stronie docelowej. Za kulisami w kodzie strony znajduje się zwykle tag <a id="name"></a>, który kieruje przeglądarkę do miejsca na stronie, do którego powinna przejść po otwarciu strony.

**Ujawnianie nagłówków HTTP**

Gdy korzystasz z sieci, jedyne, na czym Ci zależy, to to, co widzisz na ekranie. Na głębszym, nieco ukrytym poziomie, dwa komputery zaangażowane w transakcję komunikują się ze sobą za pośrednictwem nagłówków HTTP. Nagłówki zwykle nie są widoczne dla ludzkiego oka, ale są dostępne dla Pythona, przeglądarki internetowej i innych programów. Jeśli chcesz, możesz wyświetlić nagłówki, co może być bardzo przydatne podczas pisania kodu umożliwiającego dostęp do sieci. Produkt, którego najczęściej używamy do przeglądania nagłówków, nazywa się HTTP Headers i jest rozszerzeniem Google Chrome. Jeśli masz przeglądarkę Chrome i chcesz ją wypróbować, użyj przeglądarki Chrome, aby przejść do https://www.esolutions.se/, przewiń w dół do sekcji Rozszerzenia Google Chrome, kliknij Nagłówki HTTP i postępuj zgodnie z instrukcjami, aby zainstalować rozszerzenie. Aby zobaczyć nagłówki, których dotyczy właśnie odwiedzenie witryny, kliknij ikonę nagłówków HTTP na pasku narzędzi Chrome (wygląda jak chmura), a zobaczysz informacje nagłówka HTTP, jak na rysunku.




Dwie najważniejsze rzeczy w nagłówkach HTTP znajdują się na samej górze, gdzie widzisz GET, po którym następuje adres URL. Oznacza to, że wysłano żądanie GET, co oznacza, że adres URL jest tylko prośbą o informacje, nic nie jest przesyłane na serwer. Adres URL po słowie GET to żądany zasób. Innym rodzajem odpowiedzi jest POST, co oznacza, że wysyłasz do serwera pewne informacje, na przykład gdy publikujesz coś na Facebooku, Twitterze lub w dowolnej innej witrynie, która akceptuje dane wejściowe od Ciebie. Drugi wiersz poniżej GET pokazuje status żądania. Pierwsza część wskazuje używany protokół. W przykładzie na rysunku jest to HTTP1.1, co oznacza po prostu żądanie WWW zgodne z regułami komunikacji protokołu HTTP w wersji 1.1.

| GET https://alansimpson.me/datascience/cheatsheets/beginnerpy3/<br>Status: HTTP/1.1 200 | |
|---|---|
| **Request Headers** | |
| Accept | text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8 |
| Accept-Encoding | gzip, deflate, br |
| Accept-Language | en-US,en;q=0.9 |
| Referer | https://alansimpson.me/datascience/cheatsheets/ |
| Upgrade-Insecure-Requests | 1 |
| User-Agent | Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chr |
| **Response Headers** | |

Liczba 200 to kod statusu, który w tym przypadku oznacza „ok, wszystko poszło dobrze”. Typowe kody stanu wymieniono w tabeli.

**Typowe kody stanu HTTP**

**Kod: Znaczenie: Powód**

200 : OK : Żadnych problemów.

400: Błędne żądanie: Serwer jest dostępny, ale nie może zrozumieć Twojego żądania, zwykle dlatego, że coś jest nie tak z Twoim adresem URL.

403 : Zabronione : Witryna wykryła, że uzyskujesz do niej programowy dostęp i nie zezwala na to.

404 : Nie znaleziono Albo adres URL jest nieprawidłowy, albo adres URL jest prawidłowy, ale treści, która pierwotnie tam była, już tam nie ma.

Wszystko, o czym tutaj mówiliśmy, ma znaczenie, ponieważ jest związane z programowym dostępem do sieci za pomocą Pythona, jak zobaczysz dalej.

**Otwieranie adresu URL z Pythona**

Aby uzyskać dostęp do sieci z poziomu programu Pythona, musisz użyć aspektów pakietu urllib. Nazwa urllib jest skrótem od URL Library. Ta jedna biblioteka w rzeczywistości składa się z modułów, z których każdy zapewnia możliwości przydatne w różnych aspektach programowego dostępu do Internetu. Tabela  podsumowuje pakiety.

**Pakiety z biblioteki urllib Pythona**

**Pakiet: Cel**

request : użyj tego do otwierania adresów URL

response : wewnętrzny kod obsługujący otrzymaną odpowiedź; nie musisz pracować z tym bezpośrednio

błąd: obsługuje wyjątki żądań

parse : dzieli adres URL na mniejsze części

robotparser : analizuje plik robots.txt witryny, który przyznaje uprawnienia botom próbującym programowo uzyskać dostęp do witryny

Przez większość czasu prawdopodobnie będziesz pracować z modułem żądań, ponieważ to ten, który pozwala otwierać zasoby z Internetu. Składnia dostępu do prostego pakietu z biblioteki to

from library import module

. . . gdzie library to nazwa większej biblioteki, a moduł to nazwa konkretnego modułu. Aby uzyskać dostęp do możliwości modułu odpowiedzi urllib, użyj tej składni u góry kodu (komentarz nad kodem jest opcjonalny):

# import the request module from urllib library.

from urllib import request

Aby otworzyć stronę internetową, użyj tej składni:

variablename = request.urlopen(url)

Zamień variablename na dowolną wybraną przez siebie nazwę zmiennej. Adres URL żądania z adresem URL zasobu, do którego chcesz uzyskać dostęp. Musisz go ująć w pojedyncze lub podwójne cudzysłowy, chyba że jest przechowywany w zmiennej. Jeśli adres URL jest już przechowywany w jakiejś zmiennej, wystarczy nazwa zmiennej bez cudzysłowów. Po uruchomieniu kodu wynikiem będzie obiekt HTTPResponse. Jako przykład, oto kod, który możesz uruchomić w notatniku Jupyter lub dowolnym

pliku .py, aby uzyskać dostęp do przykładowej strony HTML, którą Alan dodał do swojej witryny tylko w tym celu:

```
# import the request module from urllib library.

from urllib import request

# URL (address) of the desired page.

sample_url = 'https://AlanSimpson.me/python/sample.html'

# Request the page and put it in a variables named the page.

thepage = request.urlopen(sample_url)

# Put the response code in a variable named status.

status = thepage.code

# What is the data type of the page?

print(type(thepage))

# What is the status code?

print(status)
```

Uruchomienie tego kodu wyświetla następujące dane wyjściowe:

```
<class 'http.client.HTTPResponse'>

200
```

Oznacza to, że zmienna o nazwie thepage zawiera plik obiektu http.client.HTTPResponse . . . czyli wszystko, co serwer odesłał w odpowiedzi na żądanie. 200 to kod stanu, który mówi, że wszystko poszło dobrze.

**Publikowanie w Internecie za pomocą Pythona**

Nie wszystkie próby uzyskania dostępu do zasobów internetowych przebiegną tak gładko, jak w poprzednim przykładzie. Na przykład wpisz ten adres URL w pasku adresu przeglądarki i naciśnij Enter:

```
https://www.google.com/search?q=python web scraping tutorial
```

Google zwraca wynik wyszukiwania wielu stron i filmów, które zawierają samouczek dotyczący skrobania stron w języku Python. Jeśli spojrzysz na pasek adresu, możesz zauważyć, że wpisany adres URL nieco się zmienił, a spacje zostały zastąpione znakami %20, jak w poniższym wierszu kodu:

```
https://www.google.comsearch?q=python%20web%20scraping%20tutorial
```

Ten %20 to kod ASCII, szesnastkowo, dla spacji, a przeglądarka robi to tylko po to, aby uniknąć wysyłania rzeczywistych spacji w adresie URL. Nic takiego. Zobaczmy teraz, co się stanie, jeśli uruchomisz ten sam kod, co powyżej, ale z adresem URL Google, a nie oryginalnym adresem URL. Oto ten kod:

```
from urllib import request

# URL (address) of the desired page.
```

```
sample_url = ' https://www.google.com/search?q=python%20web%20scraping%20
tutorial'
# Request the page and put it in a variables named the page.
thepage = request.urlopen(sample_url)
# Put the response code in a variable named status.
status = thepage.code
# What is the data type of the page?
print(type(thepage))
# What is the status code?
print(status)
```

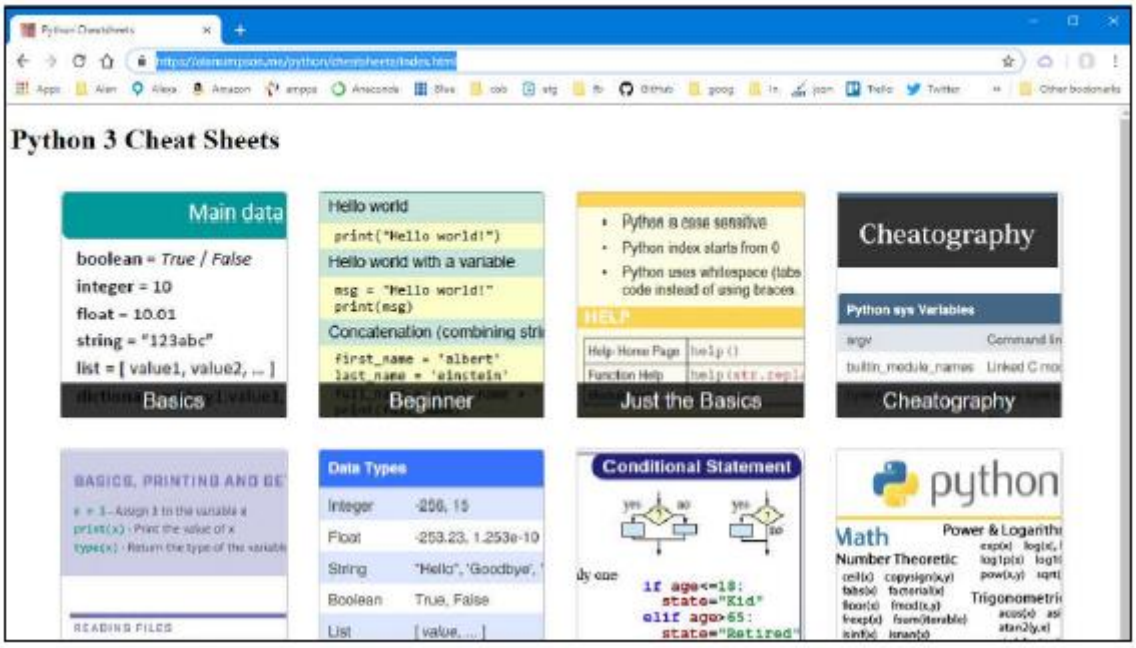
Po uruchomieniu tego kodu wszystko nie idzie tak gładko. Możesz zobaczyć kilka komunikatów o błędach, ale najważniejszy jest ten, który zwykle brzmi mniej więcej tak:

**Błąd HTTP: Błąd HTTP 403: Zabronione**

„Błąd" nie dotyczy Twojego kodu. Jest to raczej błąd HTTP. W szczególności jest to numer błędu 403 dla „Zakazany". Zasadniczo twój kod zadziałał. Oznacza to, że adres URL został wysłany do Google. Ale Google odpowiedział: „Przepraszamy, możesz przeszukiwać naszą witrynę z przeglądarki, ale nie z takiego kodu Pythona". Google nie jest jedyną witryną, która to robi. Wiele dużych witryn odrzuca próby uzyskania dostępu do ich treści w sposób programowy, po części w celu ochrony swoich praw do własnych treści, a po części w celu uzyskania pewnej kontroli nad przychodzącym ruchem. Dobrą wiadomością jest to, że witryny, które nie pozwalają na bezpośrednie publikowanie przy użyciu Pythona lub innego języka programowania, często umożliwiają publikowanie treści. Ale musisz to zrobić za pośrednictwem ich API (interfejs programowania aplikacji). Nadal możesz używać Pythona jako języka programowania. Musisz tylko przestrzegać ich zasad, kiedy to robisz. Łatwym sposobem sprawdzenia, czy witryna ma taki interfejs API, jest po prostu wyszukanie w Google swojej intencji i języka. Na przykład opublikuj na Facebooku za pomocą Pythona lub opublikuj na Twitterze za pomocą Pythona lub czegoś podobnego. Nie będziemy próbować podawać tutaj przykładu rzeczywistego robienia czegoś takiego, ponieważ często zmieniają zasady i wszystko, co powiemy, może być nieaktualne w momencie, gdy to czytasz. Ale wyszukiwanie w Google powinno sprawić, że będziesz chciał wiedzieć. Jeśli otrzymasz wiele wyników, skup się na tych, które zostały opublikowane jako ostatnie.

**Skrobanie sieci za pomocą Pythona**

Za każdym razem, gdy żądasz strony z sieci, jest ona dostarczana jako strona internetowa, zwykle składająca się z kodu HTML i treści. HTML to kod znaczników, który w połączeniu z innym językiem zwanym CSS mówi przeglądarce, jak wyświetlić zawartość pod względem rozmiaru, położenia, czcionki, obrazów i wszystkich innych wizualnych, stylistycznych kwestii. W naszej przeglądarce internetowej nie widzisz tego kodu HTML ani CSS. Widzisz tylko treść, która zazwyczaj jest zawarta w blokach kodu HTML na stronie. Jako działający przykład użyjemy stosunkowo prostej strony pokazanej na rysunku




Kod, który mówi przeglądarce, jak wyświetlić zawartość tej strony, jest niewidoczny w przeglądarce, chyba że przeglądasz kod źródłowy. W większości przeglądarek możesz to zrobić, naciskając klawisz F12 lub klikając prawym przyciskiem myszy puste miejsce na stronie i wybierając Wyświetl źródło lub Sprawdź lub inną podobną opcję, w zależności od używanej marki i wersji. Na większości stron internetowych rzeczywista treść — to, co widzisz w przeglądarce — znajduje się między tagami <body> ... </body>. W treści strony mogą znajdować się sekcje na nagłówek, stopkę paska nawigacyjnego, być może reklamy lub cokolwiek innego. Na tej konkretnej stronie prawdziwe „mięso” treści znajduje się między tagami <article> ... </article>. Każda karta widoczna w przeglądarce jest zdefiniowana jako link w tagach <a> ... </a>. Rysunek  pokazuje część kodu HTML strony z rysunku 3.

```
<body>
    <h1>Python 3 Cheat Sheets</h1>
    <article>
        <a href="http://www.sixthresearcher.com/python-3-reference-cheat-sheet-for-beginners/">
            <img src="../../datascience/python/basics/basics256.jpg" alt="Python Basics Cheat Sheet">
            <span>Basics</span>
        </a>
        <a href="https://alansimpson.me/datascience/python/beginner/">
            <img src="../../datascience/python/beginner/beginner256.jpg" alt="Python Beginner Cheat Sheet">
            <span>Beginner</span>
        </a>
    </article>
</body>
```

Pokazujemy tylko kod pierwszych dwóch linków na stronie, ale wszystkie linki mają tę samą strukturę. Wszystkie są zawarte w sekcji oznaczonej parą znaczników <article> ... </article>.

Zauważ, że każdy link składa się z kilku tagów, jak podsumowano tutaj:

<a> ... </a>: Tagi a (czasami nazywane kotwicą) określają, dokąd przeglądarka powinna skierować użytkownika po kliknięciu łącza. Część href= tagu <a> to dokładny adres URL strony, na którą użytkownik powinien zostać przeniesiony.

<img>: tag img określa obraz wyświetlany dla każdego łącza. Atrybut src= w tym tagu określa źródło obrazu — innymi słowy dokładną lokalizację i nazwę pliku, które mają być wyświetlane dla tego łącza.

 ... : Na dole linku znajduje się tekst ujęty w tagi  ... . Ten tekst pojawia się u dołu każdego łącza jako biały tekst na czarnym tle.

Termin „web scraping” odnosi się do otwierania strony internetowej w celu programowego wyodrębnienia jej informacji do wykorzystania w inny sposób. Python ma świetne możliwości skrobania stron internetowych i jest to gorący temat, o którym większość ludzi chce się dowiedzieć. Tak więc w pierwszych częściach tej części skupimy się na tym, używając przykładowej strony, którą właśnie pokazaliśmy jako przykład roboczy. Termin „skrobanie ekranu” jest również używany jako synonim „web scrapingu”. Chociaż, jak zobaczysz tutaj, tak naprawdę nie zeskrobujesz treści z ekranu komputera. Wyskrobujesz go z pliku, który jest wysyłany do przeglądarki, aby przeglądarka mogła wyświetlić informacje na ekranie. W kodzie Pythona będziesz musiał zaimportować dwa moduły, z których oba są dostarczane z Anacondą, więc powinieneś już je mieć. Jednym z nich jest moduł żądania z urllib (skrót od URL Library), który umożliwia wysłanie żądania do sieci w celu uzyskania zasobu i odczytanie tego, co sieć oddaje. Drugi nazywa się BeautifulSoup i pochodzi z piosenki z książki Alicja w Krainie Czarów. Ten zapewnia narzędzia do analizowania strony internetowej, którą pobrałeś, pod kątem określonych elementów danych, którymi jesteś zainteresowany. Aby rozpocząć, otwórz notatnik Jupyter lub utwórz plik .py w VS Code i wpisz pierwsze dwa wiersze w następujący sposób:

```
# Get request module from url library.

from urllib import request

# This one has handy tools for scraping a web page.

from bs4 import BeautifulSoup
```

Następnie musisz powiedzieć Pythonowi, gdzie w Internecie znajduje się interesująca strona. W tym przypadku adres URL to

https://alansimpson.me/python/scrape_sample.html

Możesz to sprawdzić, wpisując adres URL w pasku adresu przeglądarki i naciskając Enter. Ale aby zeskrobać stronę, musisz umieścić ten adres URL w kodzie Pythona. Możesz nadać mu krótką nazwę, na przykład page_url, przypisując go do zmiennej takiej jak ta:

```
# Sample page for practice.

page_url = 'https://alansimpson.me/python/scrape_sample.html'
```

Aby umieścić stronę internetową w tej lokalizacji w swojej aplikacji w Pythonie, utwórz kolejną zmienną, którą nazwiemy rawpage, i użyj metody urlopen modułu request, aby odczytać tę stronę. Oto jak wygląda ten kod:

```
# Open that page.

rawpage = request.urlopen(page_url)
```

Aby stosunkowo łatwo przeanalizować tę stronę w kolejnym kodzie, skopiuj ją do obiektu BeautifulSoup. W naszym kodzie nazwiemy zupę obiektu. Będziesz także musiał powiedzieć BeautifulSoup, w jaki sposób chcesz przeanalizować stronę. Możesz użyć html5lib, który jest również dostarczany z Anacondą. Więc po prostu dodaj te linie:

```
# Make a BeautifulSoup object from the html page.

soup = BeautifulSoup(rawpage, 'html5lib')
```

Parsowanie części strony

Większość stron internetowych zawiera dużo kodu treści w nagłówku, stopce, paskach bocznych, reklamach i wszystkim innym, co dzieje się na stronie. Główna treść często znajduje się tylko w jednej sekcji. Jeśli możesz zidentyfikować tylko tę sekcję, Twój kod analizujący będzie działał szybciej. W tym przykładzie, w którym sami stworzyliśmy stronę internetową, umieściliśmy całą główną treść między parą tagów <article> ... </article>. W poniższym kodzie przypisujemy ten blok kodu do zmiennej o nazwie content. Później kod na stronie przeanalizuje tylko tę część strony, co może pomóc poprawić szybkość i dokładność.

# Isolate the main content block.

content = soup.article

**Przechowywanie przeanalizowanej treści**

Twoim celem podczas skrobania strony internetowej jest zazwyczaj zebranie tylko określonych danych, które Cię interesują. W tym przypadku potrzebujemy tylko adresu URL, źródła obrazu i tekstu dla wielu linków. Wiemy, że będzie więcej niż jedna linia. Na początek łatwym sposobem ich przechowywania byłoby umieszczenie ich na liście. W tym kodzie tworzymy w tym celu pustą listę o nazwie links_list, używając tego kodu:

# Create an empty list for dictionary items.

links_list = []

Następnie kod musi przechodzić przez każdy tag linku w treści strony. Każdy z nich zaczyna się i kończy znacznikiem <a>. Aby powiedzieć Pythonowi, aby przechodził przez każde łącze z osobna, użyj metody find_all BeautifulSoup w pętli. W poniższym kodzie, przeglądając linki, przypisujemy bieżące łącze do zmiennej o nazwie link:

# Loop through all the links in the article.

for link in content.find_all('a'):

Kod każdego linku będzie wyglądał mniej więcej tak, chociaż każdy będzie miał unikalny adres URL, źródło obrazu i tekst:

<a href="https://alansimpson.me/datascience/python/lists/">

<img src="../datascience/python/lists/lists256.jpg" alt="Python lists">

Lists

</a>

Trzy elementy danych, których potrzebujemy, to:

* Adres URL linku, który jest ujęty w cudzysłowy po href= w tagu <a>.

* Źródło obrazu, które jest ujęte w cudzysłowy po src= w tagu img.

* Tekst linku, który jest zawarty w znacznikach  ... .

Poniższy kod przedstawia każdy z tych komponentów za pomocą metody .get() w BeautifulSoup, aby wyizolować coś wewnątrz łącza (to znaczy pomiędzy znacznikami <a> i </a> oznaczającymi początek i koniec każdego łącza ). Aby uzyskać część adresu URL łącza i umieścić ją w zmiennej o nazwie url, używamy tego kodu:

url = link.get('href')

Musisz upewnić się, że jest to wcięcie pod pętlą, aby było wykonywane dla każdego łącza. Aby uzyskać źródło obrazu i umieścić je w zmiennej o nazwie img, użyliśmy tego kodu:

img = link.img.get('src')

Tekst znajduje się między  ...  tekstem w dolnej części łącza. Aby pobrać to i umieścić w zmiennej o nazwie text, dodaj ten wiersz kodu:

text = link.span.text

Nie musisz do tego używać .get(), ponieważ tekst nie znajduje się w atrybucie HTML, takim jak href= lub src=. To tylko tekst między tagami  ... . Na koniec musisz to wszystko zapisać, zanim przejdziesz do następnego linku na stronie. Łatwym sposobem na to jest dodanie wszystkich trzech elementów danych do listy links_list przy użyciu tego kodu:

links_list.append({'url' : url, 'img': img, 'text': text})

To tyle, jeśli chodzi o przechowywanie wszystkich danych dla jednego łącza przy każdym przejściu przez pętlę. Jest jednak jeszcze jedno zastrzeżenie. Przeglądarki internetowe bardzo wybaczają błędy w kodzie HTML. Jest więc możliwe, że tu i ówdzie pojawi się błędnie wpisany kod lub brakujący kod, co może spowodować awarię pętli. Zwykle będzie to miało postać błędu atrybutu, gdy Python nie może znaleźć jakiegoś atrybutu. Jeśli brakuje danych, wolimy, aby Python po prostu pominął błędną linię i kontynuował, zamiast zawieszać się i palić, pozostawiając nas bez danych. Powinniśmy więc wypróbować cały biznes chwytania części: blok, który, jeśli się nie powiedzie, pozwala Pythonowi po prostu pominąć to jedno łącze i przejść do następnego. Kod do tego wygląda tak:

# Try to get the href, image url, and text.

try:

url = link.get('href')

img = link.img.get('src')

text = link.span.text

links_list.append({'url' : url, 'img': img, 'text': text})

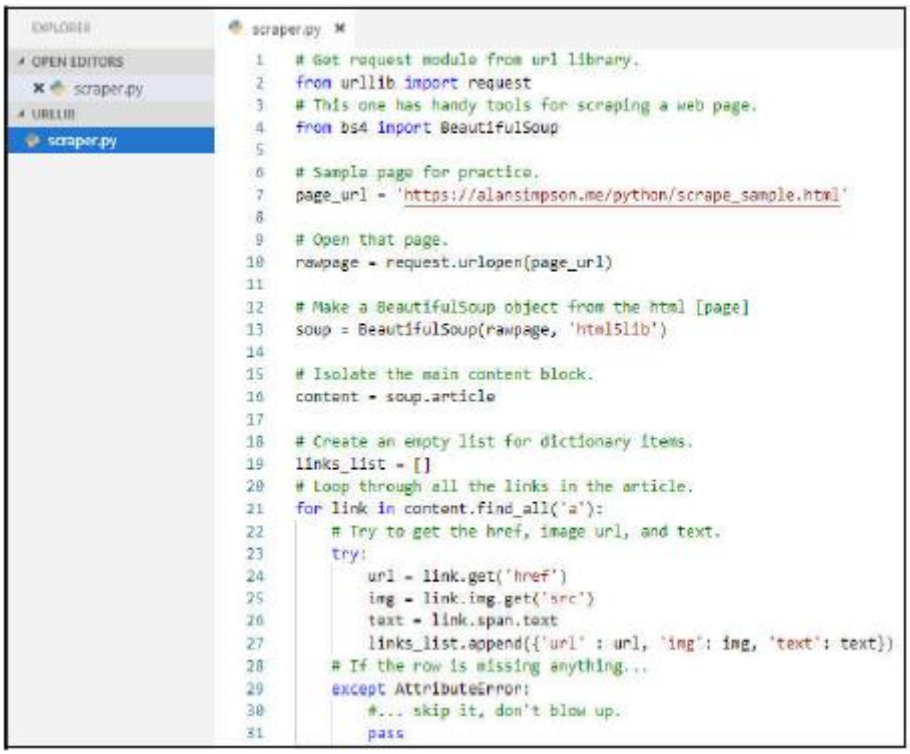
# If the row is missing anything...

except AttributeError:

#... skip it, don't blow up.

pass

Rysunek pokazuje cały kod w obecnej postaci. Jeśli uruchomisz to w ten sposób, lista link_list zostanie wypełniona wszystkimi zebranymi danymi. Ale to niewiele ci da. Są szanse, że będziesz chciał zapisać je jako dane do wykorzystania w innym miejscu. Możesz to zrobić, zapisując dane w pliku JSON, pliku CSV lub w obu, w zależności od tego, co jest dla Ciebie najwygodniejsze. W poniższych sekcjach pokażemy, jak zrobić jedno i drugie.




**Zapisywanie zeskrobanych danych do pliku JSON**

Aby zapisać zeskrobane dane w pliku JSON, najpierw zaimportuj moduł json u góry kodu, tak jak poniżej:

# If you want to dump data to json file.

import json

Następnie pod pętlą (bez wcięcia, ponieważ nie chcesz powtarzać kodu przy każdym przejściu przez pętlę) najpierw otwórz plik do pisania, używając tego kodu. Możesz nadać sobie dowolną nazwę. Zdecydowaliśmy się nazwać nasze links.json, jak widać w poniższym kodzie:

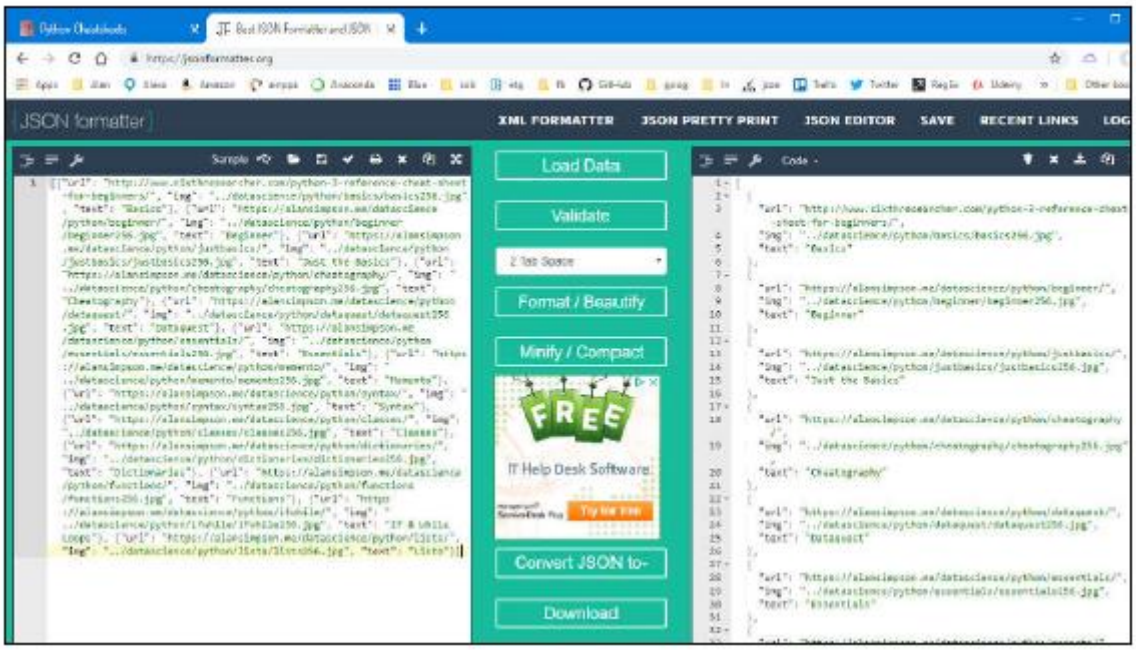
# Save as a JSON file.

with open('links.json', 'w', encoding='utf-8') as links_file:

Następnie, wcięcie pod tym wierszem, użyj funkcji json.dump(), aby zrzucić zawartość links_list do tego pliku JSON. Zazwyczaj dodajemy parametr sure_ascii=false tylko po to, aby zachować obce znaki, ale jest to całkowicie opcjonalne:

json.dump(links_list, links_file, ensure_ascii=False)

Otóż to! Po uruchomieniu kodu będziesz mieć plik o nazwie links.json, który zawiera wszystkie zeskrobane dane w formacie JSON. Jeśli otworzysz go z VS Code, będzie wyglądał jak jedna długa linia, ponieważ nie dodaliśmy żadnych podziałów linii ani spacji do wcięcia. Ale kiedy zobaczysz to jako jedną długą linię, możesz skopiować/wkleić całość do witryny takiej jak jsonformatter.org, która wyświetli dane w bardziej czytelnym formacie bez zmiany zawartości pliku, jak na rysunku




**Zapisywanie zeskrobanych danych do pliku CSV**

Jeśli z jakiegoś powodu wolisz przejść bezpośrednio do pliku CSV z naszymi zeskanowanymi danymi, zacznij od zaimportowania modułu csv u góry kodu, tak jak poniżej:

# If you want to save to CSV.

import csv

Następnie poniżej i poza pętlą, która tworzy links_list, otwórz w trybie zapisu plik z wybraną nazwą pliku. W poniższym kodzie nazwaliśmy nasz links.csv. Użyliśmy również newline='', aby uniknąć wstawiania dodatkowej nowej linii na końcu każdego wiersza.

# Save it to a CSV.

with open('links.csv', 'w', newline='') as csv_out:

Poniżej tego otwartego wcięcia utwórz program piszący csv, który kieruje plik na podstawie nazwy przypisanej na końcu z . . . linia:

csv_writer = csv.writer(csv_out)

Pierwszy wiersz pliku CSV zazwyczaj zawiera nazwy pól (lub nagłówki kolumn, jak się je również nazywa). Następnym krokiem jest dodanie tego wiersza do tabeli, używając dowolnych nazw, które chcesz zastosować do nagłówków, na przykład:

# Create the header row

csv_writer.writerow(['url','img','text'])

Następnie możesz zapisać wszystkie dane z link_list do pliku CSV, przechodząc przez link_list i zapisując trzy elementy danych, oddzielone przecinkami, w nowych wierszach. Oto ten kod:

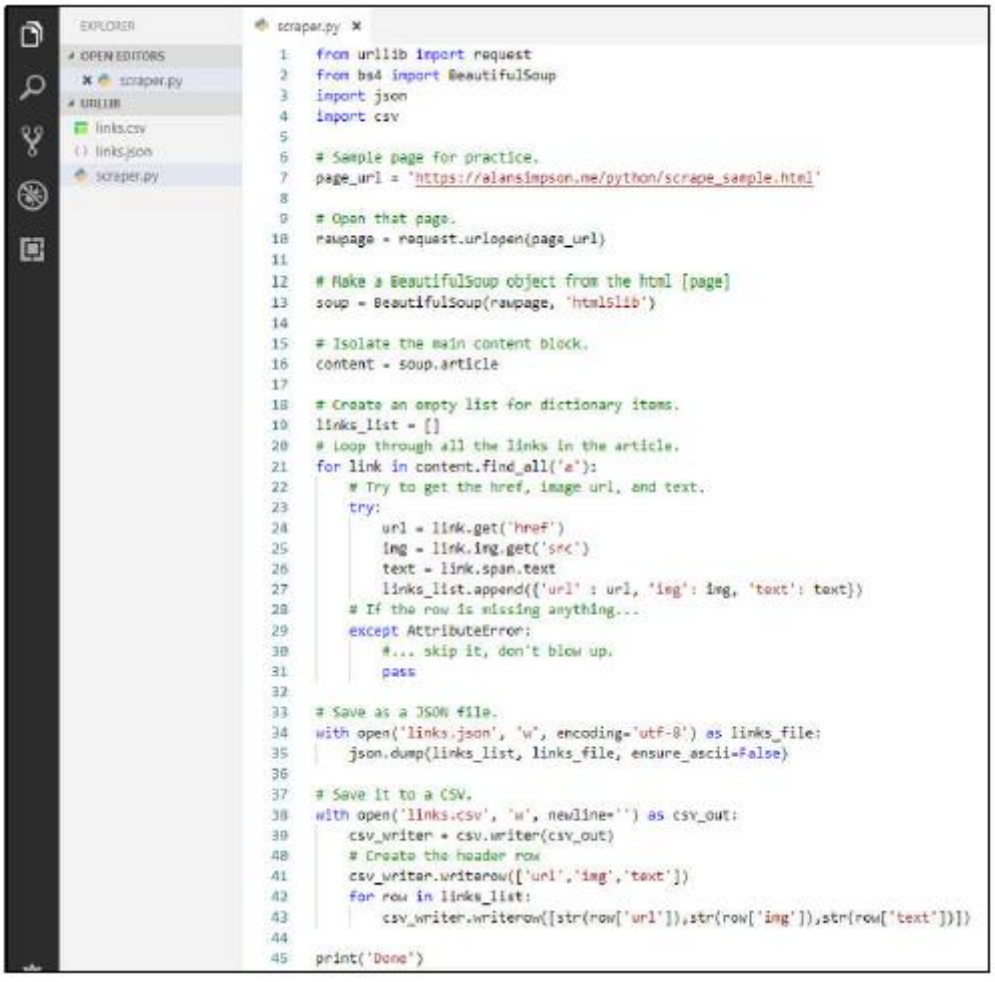
for row in links_list:

csv_writer.writerow([str(row['url']),str(row['img']),str(row['text'])])

Uruchomienie kodu tworzy plik o nazwie links.csv. Jeśli otworzysz ten plik w programie Excel lub innym arkuszu kalkulacyjnym, zobaczysz, że dane są uporządkowane w tabeli z kolumnami oznaczonymi jako url, img i text, jak na rysunku

| | A | B | C |
|---|---|---|---|
| 1 | url | img | text |
| 2 | http://www.sixthresearcher.com/python-3-reference-cheat-sheet | ../datascience/python/basics/basics256.jpg | Basics |
| 3 | https://alansimpson.me/datascience/python/beginner/ | ../datascience/python/beginner/beginner256.jpg | Beginner |
| 4 | https://alansimpson.me/datascience/python/justbasics/ | ../datascience/python/justbasics/justbasics256.jpg | Just the Basics |
| 5 | https://alansimpson.me/datascience/python/cheatography/ | ../datascience/python/cheatography/cheatography256.jpg | Cheatography |
| 6 | https://alansimpson.me/datascience/python/dataquest/ | ../datascience/python/dataquest/dataquest256.jpg | Dataquest |
| 7 | https://alansimpson.me/datascience/python/essentials/ | ../datascience/python/essentials/essentials256.jpg | Essentials |
| 8 | https://alansimpson.me/datascience/python/memento/ | ../datascience/python/memento/memento256.jpg | Memento |
| 9 | https://alansimpson.me/datascience/python/syntax/ | ../datascience/python/syntax/syntax256.jpg | Syntax |
| 10 | https://alansimpson.me/datascience/python/classes/ | ../datascience/python/classes/classes256.jpg | Classes |
| 11 | https://alansimpson.me/datascience/python/dictionaries/ | ../datascience/python/dictionaries/dictionaries256.jpg | Dictionaries |
| 12 | https://alansimpson.me/datascience/python/functions/ | ../datascience/python/functions/functions256.jpg | Functions |
| 13 | https://alansimpson.me/datascience/python/ifwhile/ | ../datascience/python/ifwhile/ifwhile256.jpg | If & While Loops |
| 14 | https://alansimpson.me/datascience/python/lists/ | ../datascience/python/lists/lists256.jpg | Lists |

Rysunek przedstawia cały kod do zeskrobywania zarówno do formatu JSON, jak i CSV.




Usunęliśmy niektóre komentarze z góry kodu, aby wyświetlić je wszystkie na jednym zrzucie ekranu. Ale chcieliśmy się tylko upewnić, że możesz zobaczyć to wszystko w jednym miejscu. Zobaczenie całego kodu we właściwej kolejności powinno pomóc w debugowaniu własnego kodu, jeśli zajdzie taka potrzeba. Programowy dostęp do sieci otwiera zupełnie nowe światy możliwości zdobywania i organizowania wiedzy. W rzeczywistości istnieje cała dziedzina nauki, zwana nauką o danych, która

dotyczy właśnie tego. Dostępnych jest również wiele specjalistycznych narzędzi, które wykraczają daleko poza to, czego się tutaj nauczyłeś. Ale zanim przejdziemy do bardziej zaawansowanych i wyspecjalizowanych aplikacji Pythona, jest jeszcze jedna podstawowa koncepcja, którą musimy omówić. W tych pierwszych rozdziałach używałeś wielu różnych rodzajów bibliotek, modułów i tym podobnych. Z następnego rozdziału dowiesz się, jak dalekosiężne jest to pojęcie oraz jak szukać i znajdować to, czego potrzebujesz, kiedy tego potrzebujesz.