このプログラムは、ネットワーク越しに一度にすべてのデータを読み込んで、コンピュータのメイン・メモリの変数imgに格納し、次にficover.jpgを開いてディスクにデータを書き出す。open()の引数wbは、書き込み専用のバイナリファイルをオープンします。このプログラムは、fileのサイズがあなたのコンピュータのメモリのサイズより小さい場合に動作します。

しかし、これが大きなオーディオファイルやビデオファイルである場合、コンピュータがメモリ不足になると、このプログラムはクラッシュするか、少なくとも動作が極端に遅くなるかもしれません。メモリ不足を避けるために、データをブロック（またはブーフ）単位で取得し、次のブロックを取得する前に各ブロックをディスクに書き込むようにしています。こうすることで、プログラムはコンピュータのメモリを使い切ることなく、どんなサイズのファイルでも読み込むことができるのです。

```
インポート urllib.request, urllib.parse, urllib.error

img = urllib.request.urlopen('http://data.pr4e.org/cover3.jpg')
fhand = open('cover3.jpg', 'wb')
size = 0
while True :
    info = img.read(100000)
    if len(info) < 1: break
    size = size + len(info)
    fhand.write(info)

print(size, 'コピーされた文字') fhand.close()


# コード: http://www.py4e.com/code3/curl2.py
```

この例では、一度に10万文字だけ読み、その文字を
次の10万文字のデータをウェブから取得する前に、cover3.jpg file を取得します。こ

のプログラムは次のように実行されます：

```
python curl2.py
230210文字がコピーされました。
```

HTMLの解析とウェブのスクレイピング

Pythonのurllib機能の一般的な使い方の1つに、ウェブスクレイピングがあります。ウェブスクレイピングとは、ウェブブラウザのふりをしてページを取得し、そのページ内のデータを調べてパターンを探すプログラムを書くことです。

例として、Googleなどの検索エンジンは、あるウェブページのソースを見て、他のページへのリンクを抽出し、そのページを取得し、リンクを抽出する、といった具合です。この手法を使って、Googleはウェブ上のほぼすべてのページを*スパイダーで調べています。*

また、Googleは、見つけたページから特定のページへのリンクの頻度を、そのページがどれだけ「重

要」であり、検索結果でどれだけ上位に表示されるべきかを示す一つの指標としています。言語を選択する▼。

PPYa4Es(ht/wMwwL.py4egomr)egular表現。

HTMLを解析する簡単な方法のひとつに、正規表現を使って特定のパターンに一致する部分文字列を繰り返し検索して抽出する方法があります。

ここでは、簡単なウェブページを紹介します：

```
<h1>最初のページ</h1>。
<p>
お好みで、切り替えることができます。
<a href="http://www.dr-chuck.com/page2.htm">セカ
ンドページ</a>です。
</p>
```

上記のテキストからリンク値をマッチングして抽出するための整形された正規表現を次のように構築することができます：

```
href="http[s]?://.+?"
```

この正規表現では、"href="http://" または "href="https://" で始まり、1つ以上の文字 (.+?) が続き、さらにダブルクォートが続く文字列を探します。s]の後ろのクエスチョンマークは、"http " の後に0個または1個の "s "が続く文字列を検索することを示します。

.+?に加えられたクエスチョンマークは、マッチが「貪欲」な方法ではなく「非貪欲」な方法で行われることを示します。非貪欲なマッチは可能な限り*小さな*マッチング文字列を見つけようとし、貪欲なマッチは可能な限り*大きな*マッチング文字列を見つけようとする。

正規表現に括弧を付けて、マッチした文字列のどの部分を取り出すかを示すと、次のようなプログラムになります：

```
# URL入力内のリンク値を検索する
import urllib.request, urllib.parse, urllib.error
import re.
輸入ssl

# SSL証明書のエラーを無視する ctx =
ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

url = input('Enter - ')
html = urllib.request.urlopen(url, context=ctx).read()
links = re.findall(b'href="(http[s]?://.*?)"', html)
for link in links：
    print(link.decode())
```

コード: http://www.py4e.com/code3/urlregex.py

言語を選択する▼。

sslライブラリにより、このプログラムはHTTPSを厳密に強制するWebサイトにアクセスすることができます。readメソッドは、HTTPResponseオブジェクトを返す代わりに、HTMLソースコードをbytesオブジェクトとして返します。findall正規表現メソッドは、正規表現にマッチするすべての文字列のリストを提供し、二重引用符の間のリンクテキストのみを返します。

プログラムを実行し、URLを入力すると、次のような出力が得られます：

```
Enter - https://docs.python.org
https://docs.python.org/3/index.html
https://www.python.org/
https://docs.python.org/3.8/
https://docs.python.org/3.7/
https://docs.python.org/3.5/
https://docs.python.org/2.7/
https://www.python.org/doc/versions/
https://www.python.org/dev/peps/
https://wiki.python.org/moin/BeginnersGuide
https://wiki.python.org/moin/PythonBooks
https://www.python.org/doc/av/
https://www.python.org/
https://www.python.org/psf/donations/
http://sphinx.pocoo.org/
```

正規表現は、HTMLが適切にフォーマットされ、予測可能である場合に非常にうまく機能します。しかし、世の中には「壊れた」HTMLページがたくさんあるので、正規表現だけを使ったソリューションでは、有効なリンクを見逃してしまったり、悪いデータで終わってしまったりする可能性があります。

これは、堅牢なHTMLパースライブラリを使用することで解決できます。

BeautifulSoupを使ったHTMLのパース

HTMLがXML[1] のように見え、一部のページがXMLであるように注意深く構成されていても、ほとんどのHTMLは一般に、XMLパーサがHTMLのページ全体を不適切な形成として拒絶するような方法で壊れています。

Pythonには、HTMLを解析してページからデータを抽出するのに役立つライブラリが多数存在します。それぞれのライブラリには長所と短所があり、ニーズに応じて選択することができます。

例として、*BeautifulSoup*ライブラリを使用して、いくつかのHTML入力をパースし、リンクを抽出するだけです。BeautifulSoupは、高度に浮き上がったHTMLを許容し、それでも必要なデータを容易に抽出することができます。BeautifulSoupのコードは、以下のサイトからダウンロードしてインストールすることができます：

https://pypi.python.org/pypi/beautifulsoup4 (https://pypi.python.org/pypi/beautifulsoup4)

Python Package Indexツール`pip`を使ったBeautifulSoupのインストールに関する情報は、以下から入手できます：

言語を選択する▼。

urllibでページを読み込み、BeautifulSoupでアンカー(a)タグからhref属性を抽出します。

```
# これを実行するには、BeautifulSoupのzipファイルをダウン
ロードする # http://www.py4e.com/code3/bs4.zip
# そして、このファイルと同じディレクトリに解凍してください。

import urllib.request, urllib.parse, urllib.error
from bs4 import BeautifulSoup
輸入ssl

# SSL証明書のエラーを無視する ctx =
ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

url = input('Enter - ')
html = urllib.request.urlopen(url, context=ctx).read()
soup = BeautifulSoup(html, 'html.parser')

# アンカータグをすべて取得する tags =
soup('a')
for tag in tags:
    print(tag.get('href', None))
```

コード: http://www.py4e.com/code3/urllinks.py

このプログラムでは、ウェブアドレスの入力を求め、ウェブページを開いてデータを読み取り、BeautifulSoupパーサーにデータを渡し、アンカータグをすべて取得して各タグのhref属性を出力します。

プログラムが実行されると、次のような出力が得られます：

```
PEYnt4erE-  (#7.poyrg4e.com)
 gindex.html
 py-modindex.html
 https://www.python.org/
 #.
 whatsnew/3.6.html
 whatsnew/index.html
 tutorial/index.html
 library/index.html
 reference/index.html
 using/index.html
 howto/index.html
 install/index.html
 distributing/index.html
 extending/index.html
 c-api/index.html
 faq/index.html
 py-modindex.html
 genindex.html
 glossary.html
 search.html
 contents.html
 bugs.html
 about.html
 license.html
 copyright.html
 download.html
 https://docs.python.org/3.8/
 https://docs.python.org/3.7/
 https://docs.python.org/3.5/
 https://docs.python.org/2.7/
 https://www.python.org/doc/versions/
 https://www.python.org/dev/peps/
 https://wiki.python.org/moin/BeginnersGuide
 https://wiki.python.org/moin/PythonBooks
 h t t p s : / / w w w . p y t h o n . o r g / d o c / a v /
 genindex.html
 py-modindex.html
 https://www.python.org/
 #.
 copyright.html
 https://www.python.org/psf/donations/
 bugs.html
 http://sphinx.pocoo.org/
```

HTMLのアンカータグの中には、相対パス（例：tutorial/index.html）やページ内参照（例：'#'）で、正規表現の要件である「http://」「https://」を含まないものがあるため、このリストはもっと長くなっています。

また、BeautifulSoupを使えば、各タグの様々な部分を引き出すことができます：

```
# この例を実行するには、BeautifulSoup.zipファイルをダウンロードする # http://www.py4e.com/code3/bs4.zip

# そして、このファイルと同じディレクトリに解凍してください。

from urllib.request import urlopen
from bs4 import BeautifulSoup
import ssl

# SSL証明書のエラーを無視する ctx =
ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

url = input('Enter - ')
html = urlopen(url, context=ctx).read()
soup = BeautifulSoup(html, "html.parser")

# アンカータグをすべて取得する tags =
soup('a')
for tag in tags:
    # タグの部分を見る print('TAG:',
    tag)
    print('URL:', tag.get('href', None))
    print('Contents:', tag.contents[0])
    print('Attrs:', tag.attrs)

# コード: http://www.py4e.com/code3/urllink2.py
```

```
python urllink2.py
エンター - http://www.dr-chuck.com/page1.htm
TAG: <a href="http://www.dr-chuck.com/page2.htm">セカン
ドページ</a>。
URL：http://www.dr-chuck.com/page2.htm 内容
はこちら：['ivセカンドページ']です。
Attrs：[('href', 'http://www.dr-chuck.com/page2.htm')] を参照してください。
```

`html.parser` は Python 3 の標準ライブラリに含まれる HTML パーサーです。他のHTMLパーサーの情報は、下記で入手できます：

http://www.crummy.com/software/BeautifulSoup/bs4/doc/#installing-a-parser（http://www.crummy.com/software/BeautifulSoup/bs4/doc/#installing-a-parser）

これらの例は、HTMLの解析に関して、BeautifulSoupの力を示すに過ぎません。

Unix / Linuxユーザーのためのボーナスセクション

Linux、Unix、Macintoshのコンピュータをお使いの方は、HTTPやFTP（File Transfer）プロトコルを使ってプレーンテキストやバイナリファイルを取得するコマンドをOSに組み込んでいるのではないでしょうか？これらのコマンドの1つが`curl`です：

```
P$Yc4urEl - ( 0hhttttpp$//4/coe:qonqm)
```

curlコマンドは "copy URL "の略で、urllibでバイナリファイルを取得するために先に挙げた2つのサンプルは、curlコマンドと同様の機能を実装しているので、www.py4e.com/code3 (https://www.py4e.com/code3)ではcurl1.pyとcurl2.pyという賢い名前が付けられています。また、実際にあなたが書いているプログラムでこのパターンを使いたい場合に備えて、このタスクをもう少し効果的に行うcurl3.pyのサンプルプログラムも用意されています。

よく似た機能を持つ第2のコマンドとして、wget があります：

```
$ wget http://www.py4e.com/cover.jpg
```

これらのコマンドはいずれも、ウェブページやリモートファイルを簡単に取得することができます。

用語集

ビューティフルスープ
HTML文書を解析し、ブラウザが一般的に無視するHTMLの不完全性の大部分を補償するデータをHTML文書から抽出するためのPythonライブラリです。BeautifulSoupのコードは、www.crummy.com (http://www.crummy.com)からダウンロードできます。

ポート
サーバーにソケット接続する際に、一般的にどのアプリケーションに接続しているかを示す番号。例として、ウェブトラフィックは通常ポート80を使用し、電子メールトラフィックはポート25を使用します。

擦過傷
プログラムがウェブブラウザのふりをしてウェブページを取得し、そのウェブページの内容を見ること。多くの場合、プログラムはあるページのリンクをたどって次のページを見つけ、ページのネットワークやソーシャルネットワークを横断することができます。

ソケット
2つのアプリケーション間のネットワーク接続で、アプリケーションはどちらの方向にもデータを送受信することができます。

クモ
ウェブ検索エンジンが、あるページを検索し、そのページからリンクされているすべてのページを検索し、インターネット上のほぼすべてのページを検索インデックスに登録することです。

エクササイズ

練習1: ソケットプログラム **socket1.py** を変更し、ユーザーに URL の入力を促し、任意のウェブページを読めるようにします。**split('/')**を使ってURLを構成要素に分解し、ソケット**接続**呼び出しのためのホスト名を抽出することができます。ユーザーが不適切な書式や存在しないURLを入力した場合に対処するために、tryと**except**を使用してエラーチェックを追加します。

言語を選択する ▼。

練習2：受信した文字数をカウントし、3000文字表示したらテキストの表示をやめるように、ソケットプログラムを変更する。このプログラムは、文書全体を取得し、総文字数をカウントし、文書の最後に文字数のカウント値を表示する必要があります。

練習3: `urllib` を使って、(1) URL から文書を取得し、(2) 3000 文字まで表示し、(3) 文書の全体の文字数をカウントするという前回の練習を再現してください。この演習ではヘッダーは気にせず、単に文書内容の最初の3000文字を表示してください。

練習4：`urllinks.py`のプログラムを変更して、取得したHTML文書から段落（p）タグを抽出して数え、その数をプログラムの出力として表示しなさい。段落のテキストは表示せず、数を数えるだけにしてください。あなたのプログラムを、いくつかの小さなウェブページと、いくつかの大きなウェブページでテストしてください。

練習5：（上級編）ソケットプログラムを変更し、ヘッダと空行を受信した後のデータのみを表示するようにします。`recv`は行ではなく、文字（改行など）を受信することを忘れないようにしましょう。

---

1. XML形式については、次章で説明する。

本書の内容に間違いがあった場合は、Github を使って、遠慮なくfixを送信してください。