■1.スクレイピング

スクレイピングとは収集したデータを解析して、必要なデータを取得すること

Pythonはネットへアクセスするライブラリが充実していて、

クローリング

ネット上に公開されている

データを集めること

クローリングやスクレイピングがしやすい言語です。



◆スクレイピングの3つのルール

|  |
| --- |
| **1.著作権を守る** |
| 公的機関や二次利用を許可しているものを利用しましょう。 |
| **2.アクセスの集中で業務妨害をしない** |
| 逮捕のイラストサーバに負荷を掛けますので、プログラムでも1アクセス後1秒待つなど工夫をしましょう  (市立図書館のシステムにスクレイピングを行い**実際に逮捕**された方が居ます) |
| **3.クローリング禁止を守る** |
| サイト側で禁止している場合があります、HTML内の[robots meta]タグを確認しましょう |

◆クローリング禁止かの確認方法 : robots.txtを参照する

Robots.txtとは？

クローリング禁止などの意志表示のファイルがサイトのルーツディレクトリに置いています

サイトURL/robots.txtで、配置されている場合は参照できます

このファイルを参照してクローリングして良いか確認しましょう

例) <https://www.google.com/robots.txt>

----------------------------------------------------------------------------------------------

User-agent:どのクロールを制御するかを指定する。「＊」→全てのクロールが対象。

Allow:クロールが許可されているもの。

Disallow:クロールが許可されていないもの。

Sitemap:クロールして欲しいURLを列挙するXMLファイル。

Crawl-delay:クロールする間隔（秒）

Sitemap:巡回してほしいURLをクローラーに伝えることができる。

----------------------------------------------------------------------------------------------

◆requestsモジュール

インターネット(Http通信)にアクセスできるモジュールです。

まずは簡単なプログラムを書いてみましょう！

今回はサンプル用に作成したページ(<http://10.201.10.34/~sk1a22/scrapingSample.html>)

そのページのHTMLをプログラムで解析してみましょう！

--------------------------------------------------------------------------------------------------

ファイル名: ­analysisSample.py

まずはインポートを行い使える状態にしましょう



今回はJavaでいうMainメソッドの役割のif文を紹介及び作成します。



requestsで取得できる情報

.text 文字列データ

.content バイナリデータ

.url アクセスしたURL

.apparent\_encoding

　推測されるｴﾝｺｰﾃﾞｨﾝｸﾞ方式

.status\_code HTTPステータスコード

.headers レスポンスヘッダー

◆\_\_name\_\_ と ‘\_\_main\_\_’について

|  |  |
| --- | --- |
| \_\_name\_\_ | ‘\_\_main\_\_’ |
| 自動で生成される変数  モジュールとして実行された場合は’\_\_ファイル名\_\_’  スクリプトとして実行された場合は’\_\_main\_\_’  上記が格納されている。 | スクリプトとして実行された場合に  格納される文字列 |

実行すると、対象ページのHTMLデータを取得し表示出来ていることを確認出来ますね。

◆取得したデータをもとにテキストファイルに保存

次に取得したデータを保存する処理も学んでおきましょう

ファイルを保存する処理を関数化して追加していきます。

---------------------------------------------------------------------------------------------

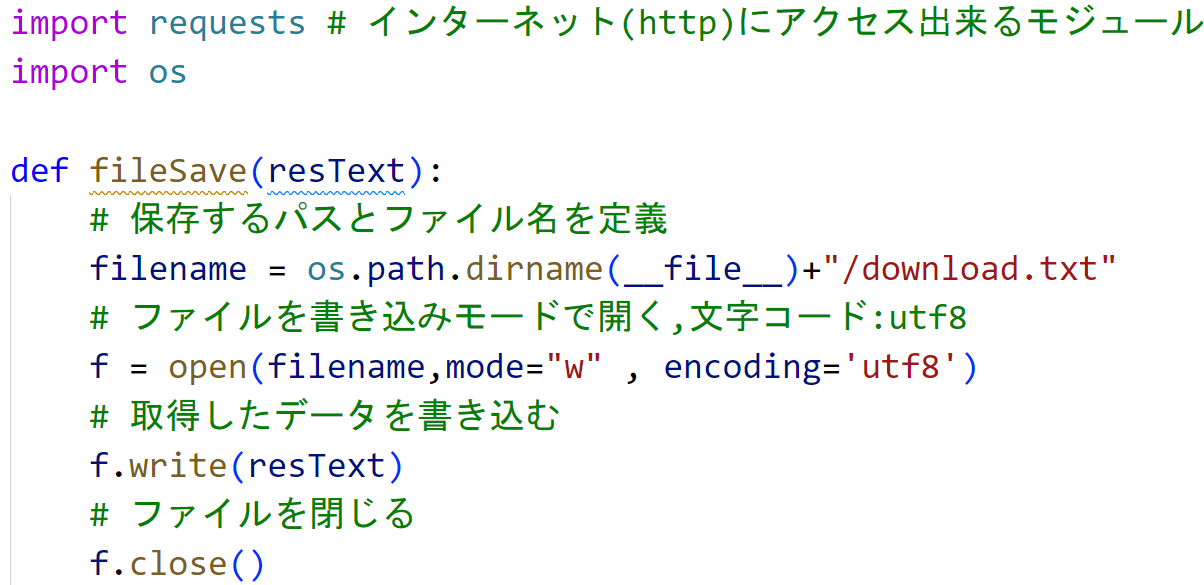
落とし穴Point:

pythonはスクリプト言語のため、1行ずつ解析しながら進んでいきます

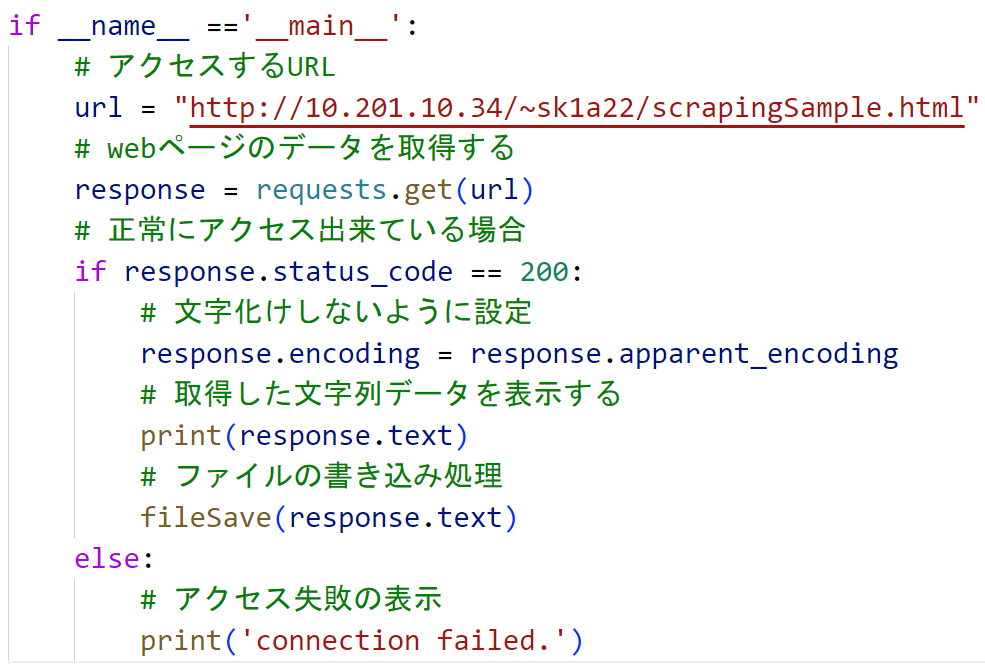
つまり、関数を定義する場合は先程の’\_\_main\_\_’のif文より上(先に)定義する必要があります。

---------------------------------------------------------------------------------------------

テキストファイルの書き込み: open , closeを行う関数を定義



\_\_main＿のif文内の処理にもうえで作成した関数の呼び出しを追加しましょう



これで実行すると「download.txt」というテキストファイルが生成されています。

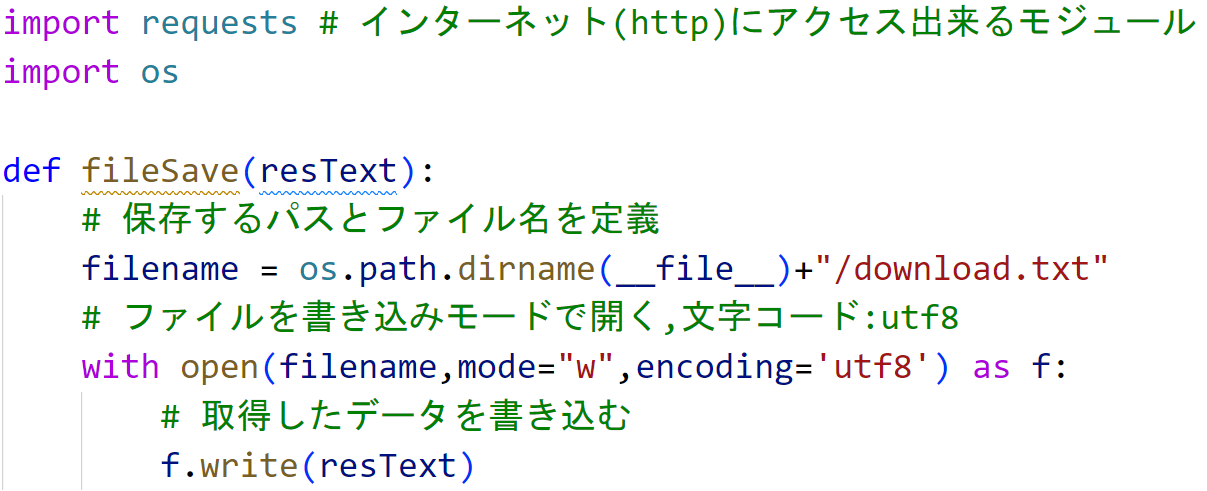
この中に取得したwebページのデータが保存されています、確認してみましょう！

◆テキストファイルに書き込む with文

ファイルを開いたら、最後に閉じるのはセットです、それを省略して書くのがwith文です

[with as]でファイルを開き[開いたときにすること]をインデントして記述しましょう

先程の内容を書き替えてみましょう



closeの記述を省略できたので少しスッキリしましたね!

■HTMLの解析

◆Beautiful Soupモジュールで解析

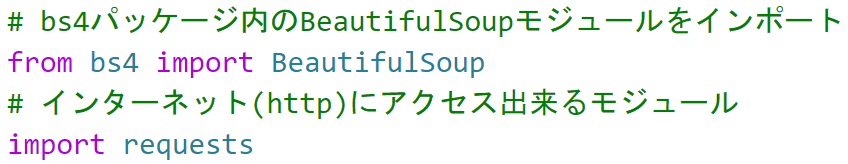
Beautiful SoupはHTMLファイルやXMLファイルからデータを抽出するためのライブラリです。

まずは使ってみましょう！

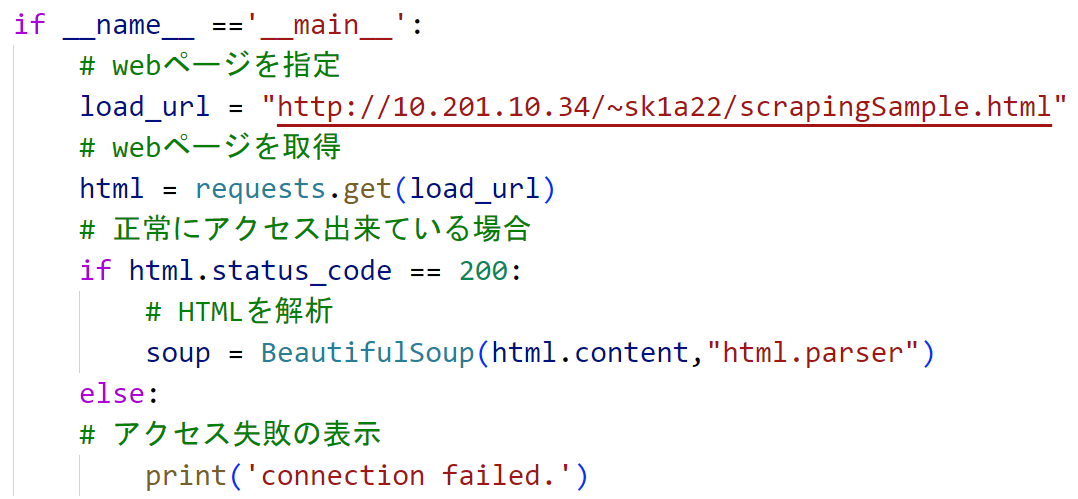
-------------------------------------------------------------------------------------------------------------

ファイル名：scrapingSample.py

まずは必要なモジュールのインポートを行います



どのページを解析するか設定していきましょう



BeautifulSoup(解析対象のHTML/XML, 利用するパーサー)

１つ目の引数には、解析対象のHTML/XMLを渡します。

２つ目の引数として解析に利用するパーサー（解析器）を指定します。

解析した内容を表示させるように出力の処理を追加しましょう



この状態で実行してみましょう！

どのような結果が表示されましたか？

結果にはページ内のHTMLの解析情報が表示されます。

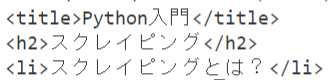
◆タグを参照して表示

解析が終わりましたので次のステップへ

こんどは指定した要素を取り出してみましょう！



この状態で実行してみましょう！



検索結果を取ることが出来ましたね！ただし、見つかったのは何故か「最初の１つ」だけでした

これは.findの仕様で「最初に見つけた要素のみ表示」するだからです。

◆.find\_all()で全て表示

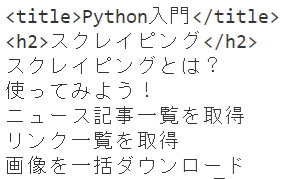
今度はliタグの部分を「.find\_all(“タグ名”)」に変更して活用してみましょう！

.find\_all (“タグ名”)により、検索した結果を配列で取得できます。

配列ならば繰り返し処理で順番に処理していけば使えそうですね



この状態で実行してみましょう！



右図のように出ましたが

これでは、どの項目のliタグ内のものか判別が出来ません。

◆.find(タグ名=値)で絞込み

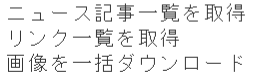
次は範囲を絞り込んでみましょう！

|  |  |
| --- | --- |
| idで絞り込んで要素を取得 | classで絞り込んで要素を取得 |
| 要素 = soup.find(id=”id名”) | 要素 = soup.find(class\_ = “class名”) |

※classでは予約語になるので、[ class\_ ]と表現

以下のように変更しましょう



このプログラムで実行してみましょう！

右図のように、chap2内のliで絞り込んで取得できました！

ここまでの内容を活用して、ニュース記事一覧を取得してみましょう！

■ニュース記事一覧を取得

今回はECCコンピュータの新着ニュース記事一覧を取得してみましょう！

<https://comp.ecc.ac.jp/news/>

先に対応するHTMLを確認し、どの要素が必要か調べます！

webページ内で右クリックし「開発者ツールで調査する」を選択



①のセレクトボタンをクリック

①クリック

②webページ内のトピック一覧をクリック



②クリック

③どのタグが目的の要素なのか特定できます

(※　サイトや日によってPG内容は更新されている可能性が高いので注意です)



③色が変わる

「最新トピックス一覧の範囲」である「class="news\_li\_wrap"」を見付ける事が出来ます！



それではpythonファイルを作成しプログラミングを行っていきましょう！

-------------------------------------------------------------------------------------------

ファイル名: newsAnalysis.py



BeautifulSoupとは

HTMLファイルやXMLファイルを解析するライブラリで、ファイルデータのダウンロードは行いません。

◆HTMLの解析

次にwebページの解析を行いましょう



BeautifulSoup(解析対象のHTML/XML, 利用するパーサー)

１つ目の引数には、解析対象のHTML/XMLを渡します。

２つ目の引数として解析に利用するパーサー（解析器）を指定します。

解析後、目的の要素で絞り込みを行います



先程、特定した

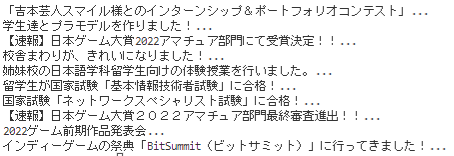
最新記事一覧の範囲のクラス名

記事が入っている要素を取得出来たので、1記事ごとに分解して表示しましょう



記事タイトルはnews\_titクラス内にありましたね

これで実行を行うと、記事の一覧を表示することが出来ます。



「…」も含まれているので

更に工夫は必要ですね。

■ページ内の画像一括ダウンロード

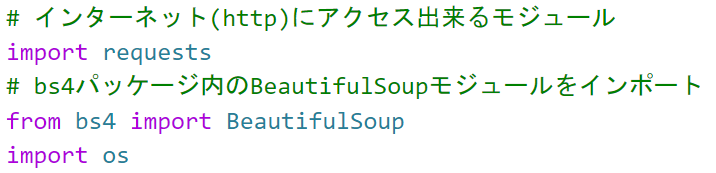
HTMLだけでなく画像の取得も一括で行うことが可能です。

新規にファイルを作成し、プログラミングしていきましょう

---------------------------------------------------------------------------------

ファイル名：imageSave.py

まずはインポートしておきましょう



どのページを解析するか決めましょう

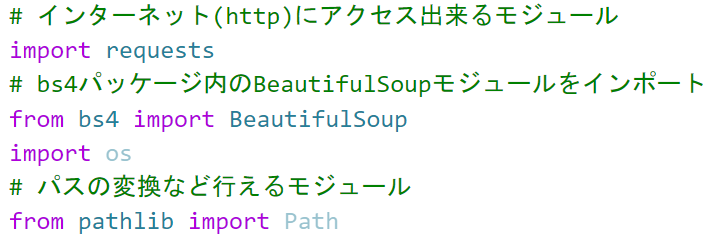


◆フォルダ作成

今回は画像を保存していきたいので

保存用のフォルダを作成する処理を行っていきます。

パスを扱うモジュールのインポートを行いましょう



どの場所に保存用フォルダを作成するか決めましょう

今回は実行ファイルと同じ階層に[download\_Image]というフォルダを作成します



この時点で実行してみましょう。

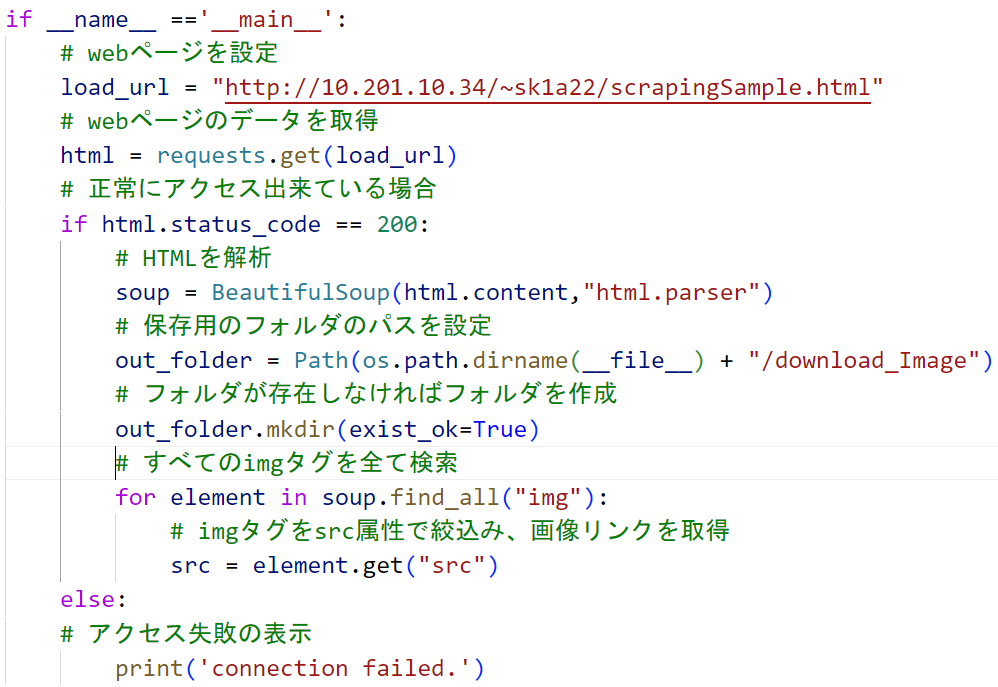
download\_Imageフォルダが

imageSave.pyと同じ階層に生成されましたね。

◆画像リンク情報の取得

次に絞込みを行いimgタグの要素を取得

更にその中の画像リンク情報を取得しましょう

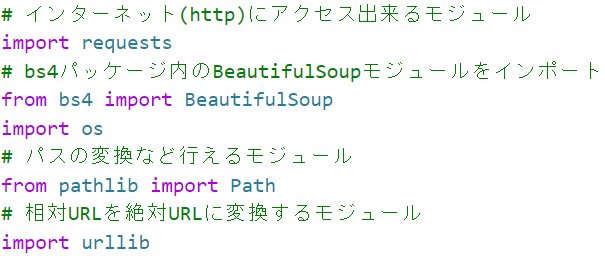


◆相対URLから絶対URLへの変換

画像リンクを取得しても、まだ相対URLの可能性があります( ./ファイル名など)

そのため、相対URLを絶対URLに変換しましょう！

上部でurl変換用のモジュールをインポートしておきます



先程のimgタグの検索for文の中に戻り絶対URLに変換しましょう



現時点でどのようなアドレスかも実行して確認しておきましょう

確認した上で、画像データを取得します



次に取得した画像データを、画像ファイルとして保存する為の準備を行います。

ファイル名をそのまま使いたいですが、どこからがファイル名なのかを指定する必要があります。

◆URLの分解

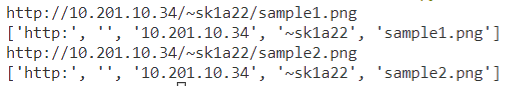
どこからがファイル名なのかを判別する為には、URLを分解すれば最後のブロックがファイル名ですね。

ではURLを/(スラッシュ)を基準に分解してみましょう。

分解には .split(基準値)メソッドを使用します



実行してみると



0

4

3

2

1

-5

-4

-3

-1

-2

配列となって、それぞれ代入されていることがわかります。

(添え字の-1とは末尾の要素から1つ目という意味になります)

これを確認すれば、ファイル名は末尾の要素であることもわかります。

◆画像ファイルの保存

ではファイル名のみを抜き出し、保存する準備を行います。

ファイルを開き、画像ファイルを保存しましょう！

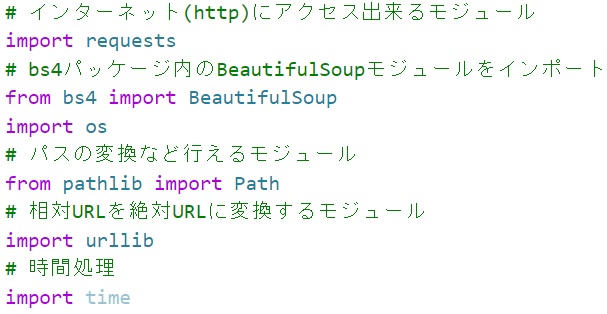
(modeのwbとはバイナリーでの書き込みモードという意味です)



◆スクレイピングのマナー

サーバに迷惑を掛けないよう、1アクセスしたら1秒待つようにしましょう

先に上部でtimeモジュールをインポートします



1ファイル操作後に1秒待つ処理を追加

time.sleep(秒数)メソッドを利用



これで実行してみましょう！

ページ内の画像ファイルが指定したフォルダ内に作成されていますね

