スクレイピング技術を用いた学務システムへの出欠入力の効率化

Improving the efficiency of attendance input to the academic affairs system using scraping technology

児玉　　満

1．はじめに

周南公立大学は、株式会社電翔が大学基幹業務全般を網羅する総合ソリューションとして開発したActive Academy Ad-vance（以下、本システム）というシステムを導入している。これはシステムを段階的に導入することができ、導入計画にあわせて各々のシステムを安価に素早く導入できるものである。またこのシステムは主に、学務系基幹システム＋学務系サービスシステムで成っており、様々な学生情報の管理も行える。また、授業関連においてもカリキュラムや時間割、成績、授業への出席なども管理している。

ここでは、様々な機能のうちの出席情報に注目する。本システムでは教員が授業科目ごとに履修している学生一人ひとりの出欠を直接入力する方法と、あらかじめパスワードを設定しておき、それを授業開始時間に学生に周知することで各々の端末（スマートフォンやPCなど）から出席を登録する方法がある。

こういった機能を利用することで学生の出席管理の負担を軽減することができ、旧来のように授業開始時に一人ひとりの名前を読み上げて出席簿に記録するといった時間を節約することができる。

また、本システムでは時間割と実時間が連動しており､授業開始とともに出席登録が可能になる。さらに、パスワードの有効時間を設定することで、授業開始から15分までは出席、15分以降30分までは遅刻、それ以降は学生からの出席登録をさせないようにし、教員が直接登録するといった方法がとれる。

しかしながら、システムを利用することで利便性が高まるだけではない。通常の授業であれば学生ごとの時間割から目的の授業を選択し出席を登録することは容易であるが、昨今のオンラインやハイフレックス、オンデマンドなどといった技法を取り入れた従来からの授業形態の変容で、システム側でそれらの出席情報の取得には完全には対応できず、毎時間教員が学生一人ひとりの出欠情報を入力しなければならないこともあり、学生数が多い授業では逆に負担の多い作業となっている。

そこで、本システムがWebシステムで動作していることを利用し、Webスクレイピングとプログラミングを用いて出欠入力の負担軽減ができないかを検証した。

2．スクレイピング技術の活用

スクレイピング技術とは、入手したひとまとまりのデータを解析し、不要な部分を削除したりして必要な部分だけを抽出したり、一部を置き換えたり、並べ替えたりして、目的に合う形に整形することをいい、主にWebサイトから必要な情報を取得し、扱いやすいようにデータを加工するための技術として知られており、こちらをWebスクレイピングという。

具体的には必要なWebサイトまたはページを選択し、巡回を行う。そこで収集したHTMLなどのWeb情報を解析して必要となるデータを抽出、汎用性のあるデータ形式に整形・変換して保存をする。これらの一連の動作をソフトウェアにより自動化されている。

この技術を利用することで、インターフェイスがWebシステムとして動作している本システムへのアクセスから授業科目のページへの遷移、学生情報の取得などが自動化でき、さらにWebフォームなどで取得した出欠情報をCSV形式で保存したデータとを照らし合わせ、ほぼ自動的に本システムの出欠情報へと反映させることが可能であると考えられる。

3．スクレイピングとプログラミング

スクレイピングはあくまで情報を収集し、データを抽出、扱いやすい形として加工する技術であるので、基本的な手順は次の通りである。

①クローリングをして対象のWebページからHTML等の内部情報を取得する
②スクレイピングをして取得したHTML等の内部情報

を解析し、必要な情報を特定し、抽出をする
③抽出した情報を扱いやすい形式に加工し、保存もしくは表示をする

この手順にさらに、プログラミングを用いることで、あらかじめ保存したCSV形式のデータとスクレイピングして得た情報を照らし合わせ、Webページを自動操作して適切な位置にデータを挿入させていくということが可能になる。

プログラミング言語にはC言語やJava、Ruby、PHP、JavaScriptといった様々な言語を用いることができるが、今回は世間一般によく利用されていることと、すく例ピングやクローリングに適したライブラリが多く揃っており、比較的簡単にプログラムが組めることなどからPythonを用いることとした。

3－1．Pythonとは

1991年にオランダ人のGuid van Ros-sum氏によって開発されたプログラミング言語である。もともとはAmoebaという分散オペレーティングシステムのシステム管理を行う目的で開発されたが、バージョンが更新されるにつれ様々な機能が追加され、現在では機械学習やWebアプリ開発などでも使われるようになり、またコードがシンプルに出来ているため、初心者から職業プログラマーまで幅広く利用されている。

3－2．Pythonライブラリ

Pythonでスクレイピングを行うには、ライブラリを用いることが一般的で、よく使われるライブラリに次の①～③などがある。

①Requests

PythonのHTTP通信ライブラリの１つであり、これを用いるとWebサイトの情報取得や画像の収集などが容易に行える。また、Pythonには標準でurllibモジュールがあるが、これよりシンプルかつ直感的にわかりやすいプログラムの記述ができる。

②Beautiful Soup

前述のRequestsモジュールやPython標準ライブラリのurllibモジュールなどで取得したWebページの情報を元に構文解析を行ったり、データの抽出をしたりすることができる。

③Selenium

Webページの取得とデータ抽出の両方が行え、さらにJavaScriptが使用されたサイトや、ログイン機構をもつサイトにも使用することができる。

今回は、本システムがログイン機構を持っていることからSeleniumが適していると考え使用することとした。

3－3．Seleniumについて

Seleniumは2004年　にJason HugginsがThoughtWorks社で内部ツールとして開発したものであるが、その後オープンソース化されている。

もともとはWebアプリケーションのUIテストの自動化や、JavaScriptのテストをするのが目的のフレームワークである。

しかしながら、テスト以外にもタスクの自動化やWebサイトのクローリングなども行えるため様々な用途で利用されている。

3－4．WebDriverについて

Webブラウザを外部のソフトウェアから操作したり、情報を取得したりできるようにするためのものである。

元々はSelenium 2.0の機能として導入されたもので、前述の通りWebアプリケーションのテストを自動実行させるために利用されるものである。

初期にはSeleniumが独自にDriverを実装していたようだが、現在は各ブラウザベンダーがそれぞれの以下のDriverを実装している。

- Microsoft Edge：Microsoft WebDriver
- Google Chrome：ChromeDriver
- FireFox：geckodriver

4．開発環境について

今回はプログラミング言語にPython、フレームワークにSeleniumを使用し、プログラムからGoogle Chromeを操作してデータの自動入力を行う方法で検証をする。

そのためのPythonの実行環境だが、システムを構築するわけではないので､IDE（統合開発環境）を使用せず、動作が軽くて手軽に実行が可能であり、カスタマイズ性の高いVisual Studio Codeを使用してプログラミングを行うこととした。

4－1．ブラウザの操作

PythonのコードからSeleniumフレームワークを用

いて、ブラウザの操作を行う。

操作のためにはWebDriverが必要になるが、今回はGoogle Chromeを利用するのでChromeDriverを用いた。これらをPythonでは以下のように記述する。

```
from selenium import webdriver
driver = webdriver.Chrome()
```

このようにWebDriverのChromeインスタンスを作成することでブラウザの操作が可能になる。

ただし、Google ChromeのバージョンとChromeDriverのバージョンが異なると実行時にエラーになり、Chromeが起動しない場合もあるためバージョン管理をする必要である。

４－２．指定したWebページを開く

本システムのWebシステムを開くためには、Seleniumのget関数を以下のように用いる。

```
driver.get('INPUT URL')
```

ただし、INPUT URLには実際のURLを記述する必要がある。これにより、ブラウザにURLが渡され、そのWebページを開くことができる。

４－３．要素の取得と操作

本システムにアクセスすると、最初にユーザーIDとパスワードを要求してくる。このログイン画面が表示されている時にHTMLをブラウザの「ページのソースを表示」などの機能で表示すると、ユーザーIDの入力を行う部分のHTMLが以下のように記述されている。

```
<input name="txtUserid" type="text"
maxlength="20" id="txtUserid" class="input">
```

Seleniumでは要素を取得するための関数としてfind_element_by_〇〇といった形式でいくつか準備されている。

〇〇にはxpath、id、cssSelecter、classNameなどが入り、いろいろな方法で取得することができるようになっているが、今回はidの情報を元に要素を取得する。

```
elem = driver.find_element_by_id('txtUserid')
```

このように記述することで、HTMLのidに記述された「txtUserid」を利用し、Seleniumで要素を取得して、elemにそれを格納している。

次に、取得した要素に対し、操作を行う。

操作関数としてはsend_keys、click、clearといったものがある。

txtUseridは入力欄のため、要素には値を設定する。

```
elem.send_keys('USERID')
```

ただし、USERIDには自身のユーザーIDを入力する。

こうすることで、HTMLのinputタグで記述された入力欄にユーザーIDを入力することができる。

また、予期しないタイミングで余計なデータが入ってしまっては誤動作の原因になるため、send_keysで値を設定する前に、安全性のために値が何も入っていない状態にしておく方が良いので、上記命令文の前に以下の１文を入れておくと良い。

```
elem.clear
```

こうしておくと、仮に何かのタイミングで値が入ってしまっていたとしても強制的に値をクリアすることができる。

同様にパスワードの入力欄の処理を行う。

まずは、パスワードの入力を行う部分のHTMLを確認すると以下のように記述されている。

```
<input name="txtUserpw" type="password"
id="txtUserpw" class="input">
```

ユーザーIDとほぼ同じ記述になっているので、idの情報を元に要素を取得する。

```
elem = driver.find_element_by_id('txtUserpw')
```

取得した要素に対し、send_keysで要素に値を設定する。

```
elem.clear
elem.send_keys('PASSWORD')
```

ただし、PASSWORDには自身のパスワードを入力する。この時、パスワードの文字列をそのまま入力することになるので他人に見られないように注意が必要。さらに、ここは実際に手入力をするようにしておくか、暗号化処理をするかなど対策を考えておかなければならない。

その後ログインボタンになっているinputタグのidの「ibtnLogin」を利用して要素を取得し、操作関数のclickを用いることでWebシステムへのログインが自動で行えるようになる。

以降、Webシステムを操作する方法に併せて順次コード化してページ遷移の自動化を行う。

４－４．データの取得

学生の出席情報はWebフォームを利用して収集を行い、それをCSV形式で出力したデータを作成する。

WebフォームとしてはGoogle Formsや、Microsoft365のFormsなどどれでも構わないが、最終的に記録情報をCSVで出力できるものが良い。

注意としては必ず学生のメールアドレスが自動で記録されるようにWebフォームの設定をしておく。

周南公立大学では全学生にメールアドレスを割り当てており、学内システム等を利用する際にはそのメールアドレスを利用しなければならない。また、そのメールアドレスはユーザー名が学籍番号になっているため、自動記録させておけばフォーム欄で学籍番号を手入力させる必要が無く、全角半角、大文字小文字の違いや入力ミスを防ぐことができる。

今回は、メールアドレスのユーザー名とWebシステムに表示される学籍番号を照合して各学生の出欠情報を操作するため、なるべく学生が手入力していない情報を用いる方が自動処理においては必要になる。

４－５．データの自動入力

学生の出席情報が記録されたCSVファイルをPythonで読み込む。

PythonにはCSVを扱うためのモジュールが標準で用意されているので、追加でインストールといったことをする必要はない。

ただし、インポートは必要なので、以下の1文を入れておく。

```
import csv
```

あとは、csv.readerを用いてCSVファイルの読み込みを行い、そこから行・列・要素を取得する。

以下、Webシステムへの出席情報の入力の流れを記載する。

①CSVファイルから一人分のデータ、つまり1行分のデータを読み出す。

②読みだしたデータからメールアドレスの情報を取得する。この時、「,」で区切られているので、その区切りを基準に前から何番目のデータかを特定しておく必要がある。

③メールアドレスからユーザー名を文字列操作で取り出す。ただし取り出す文字数が学年および学部で異なるため、文字数で取り出すのではなく、「@」を基準にそれよりも前にある文字列で取り出す必要がある。

④Webシステム側にある学籍番号と照合をする。ただし、Webシステムに記載されている学籍番号の記述が2年生以上ではメールアドレスのユーザー名と異なるため、変換するプログラムが必要となる。

⑤その情報が一致した時に、Webシステムの表組の行数を取得する。この時、Webシステムではtableタグで表組されており、さらに行数を含む形でidが振られている。

⑥出席のラジオボタンのidを⑤で取得した行数を組み合わせて生成する。

```
 'ctl00_cphMain_grdJugSearchList_ctl' +
'行数' + '_rdbShusseki'
```

このような形になるように生成を行う。

⑦生成したidを用いてseleniumの機能でチェックを入れる。チェックに関してはラジオボタンとしてHTMLの記述がなされているので、単純に操作関数のclickを用いる。

⑧①～⑦を繰り返しCSVに記載された学生数分（行数分）の処理をする。

⑨処理が終わった段階で、「登録」ボタンのidを利用して自動処理でクリックして登録を完了する。

５．考察

実際にスクレイピングとプログラミングを組み合わせて用いることで、本システムへの出欠情報の反映は問題なく行えることが分かった。また、プログラムで自動化することで手入力では起こりやすい入力ミスを無くせることや入力にかかる時間の軽減にもなることがわかり、非常に有用であった。

しかしながら、以下に記したような問題点もある。

① 1年生と2年生以上では学籍番号の振り方が根本的に異なるので、現時点では学籍番号の取得方法を2種類作成しなければならない。しかしながらこれは現在の1年生が4年生になれば学籍番号が統一されるのでお

のずと解消される。

② 2年生以上の学籍番号とメールアドレスのユーザー名の振り方のルールが異なる。たとえば、福祉情報学部の学生の場合、学籍番号は「W」から始まるが、ユーザー名は「wi」からになっている。また経済学部では学籍番号は「E」「B」から始まるが、ユーザー名は「e」「b」と小文字が使われている。

さらに、「W」や「E」「B」以降に2桁の数字があり、その次に「－」があり、その後ろの3桁の数字が付いているが、ユーザー名には「－」が入っていない。

そのため、学籍番号とユーザー名を照合する際には文字列操作をするプログラムを介さなければならないが、①と同様で1年生が4年生になった時点でこの問題は解消される。

③ChromeDriverを用いる方法の場合、先に記述したように、DriverとChromeのバージョンが大きく異なると自動処理が動作しないことがある。

④Google Chromeは起動時に自動的にアップデートの確認が行われ、上位バージョンがあった場合にはアップデートされるが、ChromeDriverの方は手動での監理となっているのでスクレイピングや自動処理プログラムの実行の際にはまずバージョン確認が必要となる。そのため、仮にプログラムをパッケージ化して単体で動作可能なアプリケーションとした場合でもDriverが自動更新されない限りは動作しなくなる可能性を取り去ることができないので、何か改善する方法を考えておかなければならない。

また、改善点としては以下のものがある。

①今回作成したプログラムは自動処理のテストを目的としていたため、プログラムの構造自体が処理の流れに沿って組んでおり、手続き型に近い形となっている。それが悪いわけではないが、プログラムの規模が大きくなると扱いづらいものになる可能性もあるので、オブジェクト指向の形にすべきと考える。

②CSVファイルから学生の出席情報を読み出し、Webシステムの学籍番号と照合して出席にチェックを入れる部分のプログラムで、照合の度にWebシステムに登録されている学生数分繰り返される処理にしたため、登録人数が多いとその分無駄な処理が多くなり、動作が遅くなる現象が出ているので、アルゴリズムの変更をして軽快な動作をさせる必要がある。

③CSVファイルを作成する部分は一切プログラミングしていないので、そちらは手動で作成しなければならないため、労力が必要である。この部分もプログラムに組み込むか、CSVを作成しなくとも、Webフォームから直接データを取り込む仕組みを作った方がより自動処理としては利点があると考える。

④現在はプログラムを直接実行する形の状態で利用しているが、今後はアプリケーション化やWebアプリ化をすることで誰でも利用できるように改良することが望ましいと考える。

今後はそういった問題点や改善点を考慮しつつプログラムの改良やスクレイピングの方法などを効率化することで様々な要件にも対応させ、自動処理の範囲を広げていきたい。

スクレイピングに関しては、自分が管理していないWebサイトの情報を抽出する場合、違法とされる可能性や、スクレイピングそのものを禁止しているサイトもあるので注意が必要である。

そのための注意点は以下の通り。

①スクレイピングを行うサイトの利用規約、著作権を事前に確認しておく。

②Webサイトには検索エンジンのクローラーからWebページへのアクセスを制限するためのrobots.txtというファイルがある。このファイルにはアクセス可能ページやアクセス不可のページについての記述があるので、該当ページのスクレイピングが許可されているか確認しておく。

③Webサイトはサーバー上にあるため、自動処理をさせてサーバーにアクセスするためには最低でも1秒以上の時間間隔で、サーバーに負荷をかけすぎないよう配慮が必要である。

④クローリングする際にはそのプログラムにクローラーを開発する際の連絡先を明示しておくとよい。これはサーバーに問題が発生した場合、管理者との連絡を容易にするためである。

6．参考文献

・平林純『なんでもPythonプログラミング平林万能IT技術研究所の奇妙な実験』株式会社技術評論社（2020年）
・児玉満『プログラミング教育に関する考察』西日本短期大学総合学術研究論集第8号（平成30年3月）
・児玉満『教育現場における「G Suite」についての考察』徳山大学総合研究所『紀要』第41号（平成31年3月）