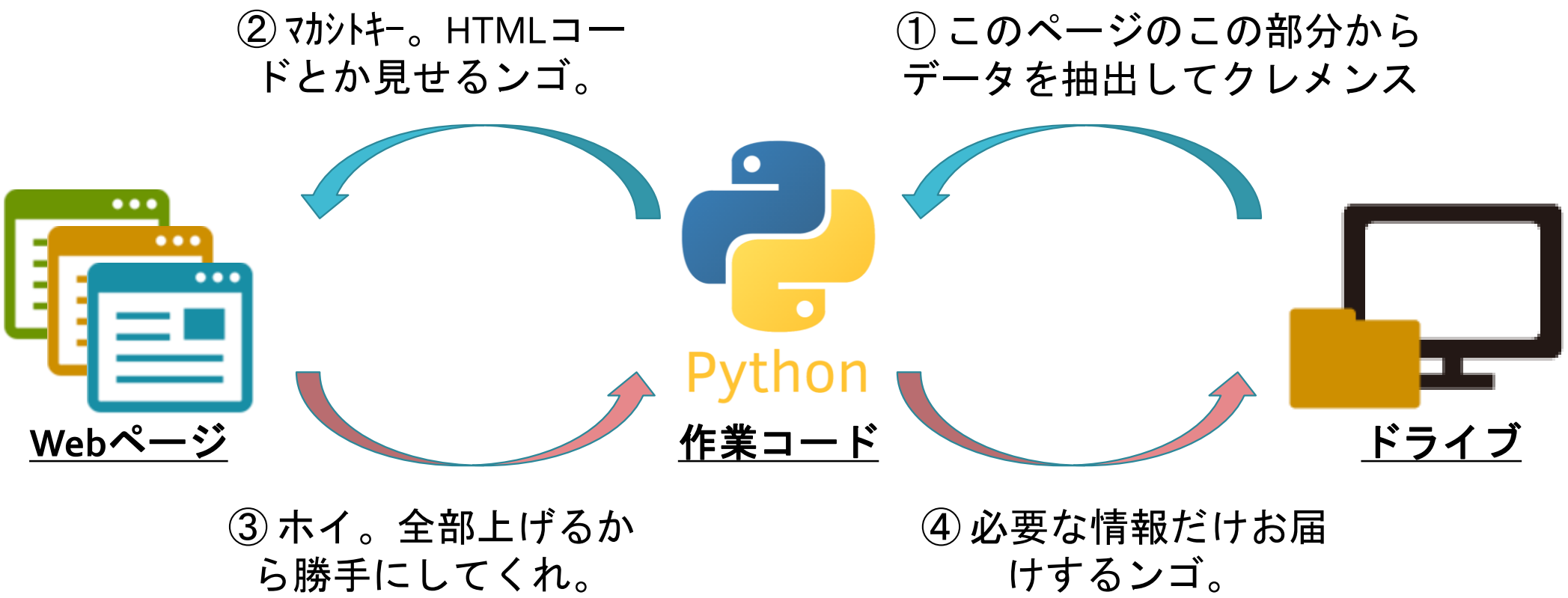
Webスクレイピングによる非恣意的情報収集について

～html と Python の読み方・使い方～

Webスクレイピングとは



プログラミング言語で作業内容を指示することによって、自動的に
必要な情報をオンラインソースから集めることができる一連の作業。

Webスクレイピングの有用性

① 統計データの作成

オンラインに落ちている情報は作成した人物の思想や周辺状況など、様々なパラメータによってバイアスが生じているだけでなく、虚偽、虚構まで含むいわば玉石混交である。統計処理はバイアスというノイズを平均化することでゼロに近づけることができ、また虚偽・虚構などは外れ値として除外することができる。これによって全く別個の動機を背景とした、多数の匿名記名の発信情報から統計的な真実を抽出することができる。

② データベースの作成

様々なデータベースが有償無償で存在するが、各データベース間の関連性は薄く、常時必要としている関連性を持つデータベースが得られるわけではない。これらのデータベースをつなぎ合わせ、自分が欲しているオリジナルのデータベースを作成することがWebスクレイピングでは可能となる

③ リアルタイムに変化する情報に対するリスクヘッジ

商品やオークション、株価など、値が変動するのもがネットには多く存在しているが、これらを常に観測し、対応することは実質不可能である。そこでスクレイピングを連続的に行い、特定の値の変動に対する応答をプログラムにすることで、機会や価値の損失などのリスクを管理することができる。

Webスクレイピングに必要なツールと知識

① 必要なツール

Google chromeのみ。つまり何も必要ないに等しい。

② 必要な知識

HTMLの見方(基礎だけでいい)とPythonの使い方

つまり、勉強する時間だけ確保できれば誰にでもできる。

（今までやってきた解析的なプログラムの使い方とは全く違いました）

ネット環境さえあればPythonはいらない

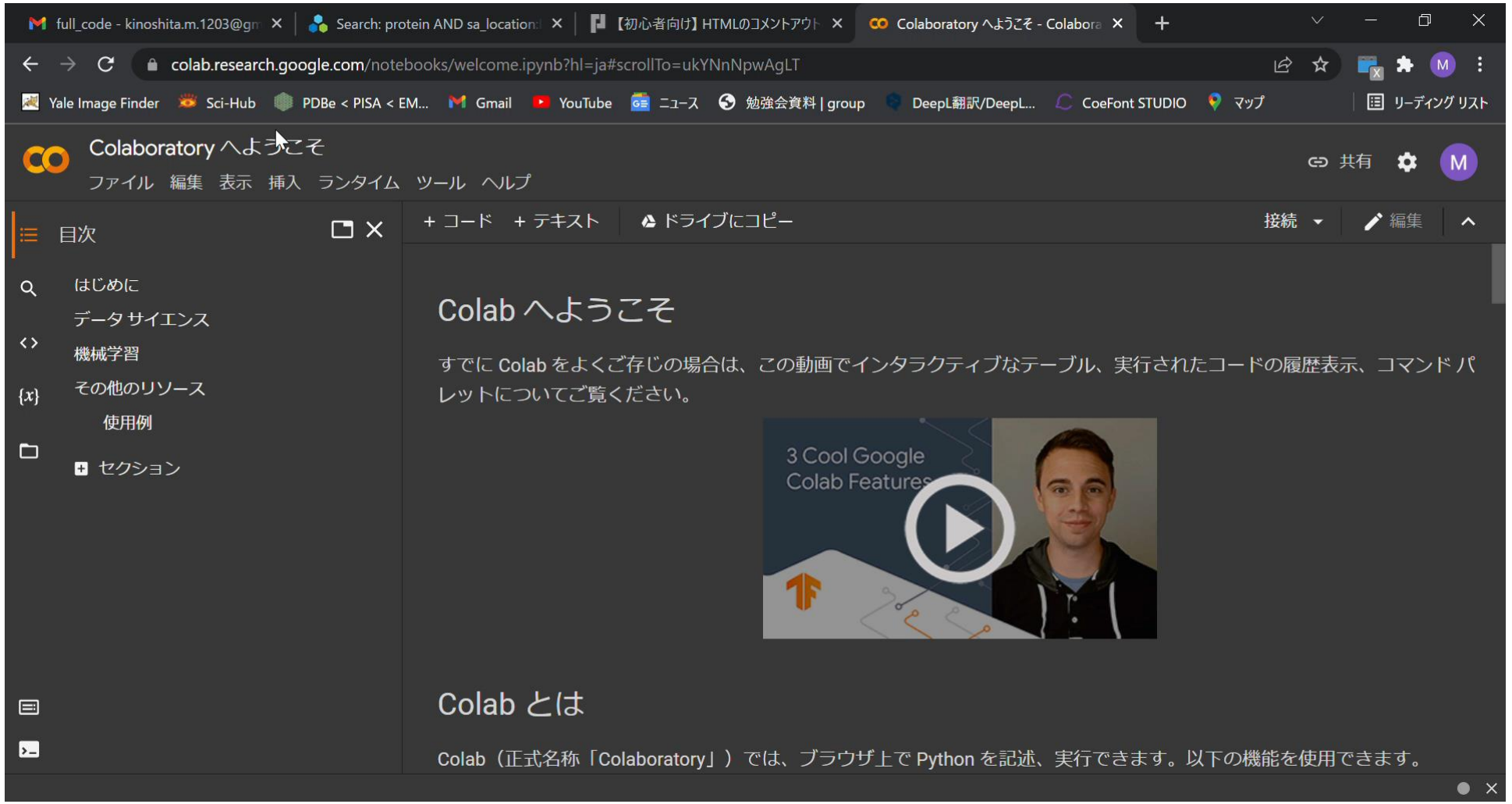
Google Colaboratory (colab)

・Google Colab は、ブラウザから Python を実行できるサービス。従来であれば自分のPCにPythonを導入し、必要なライブラリを入れる必要があったが、一切いらない。ある程度必要なライブラリはインストールされているし、必要であれば追加できる。ちょっと使ってみるくらいならこれで十分。

・さらに言えば、作業はGoogleの鯖で行われるため、会社において通常アク禁になっているページにもアクセスできる。

・GPUの演算領域も割り当て得られているため、PCの処理はブラウザを表示することだけで済む。

Colabを使ってみる



対話型のinterfaceとして使える。

必要なライブラリ

BeautifulSoup: htmlの読み込みや検索、抽出などができる。これが９割といっていい。

Selenium: Pythonでブラウザを使用するライブラリ。これが５分。あと残り全部で５分。

Pydrive:pythonで google drive APIの処理

Google.colab: colabがgoogle driveと連携するためのライブラリ

Oauth2client: なんだろうね？よくわかんない。(←これ、超大事)

Colabにインストールされていないライブラリのインストール方法

```
!pip install -q -U selenium
!pip install -U -q PyDrive
```

このコードを最初に加えることでColabにない各々のライブラリをインストールして使えるようにしてくれる。で、その下流でimportすればOK!

Webページの構成要素

Html: Hypertext Markup Language

Webページを作成するための最も基礎となるコード。この土台にCSSやJavaScriptを加えることで所謂webページになる。

ですが、スクレイピングをするだけなら
htmlが読めるだけでいい！

css も JavaScriptも読む必要はない。何なら書く能力なんて一切いらない。

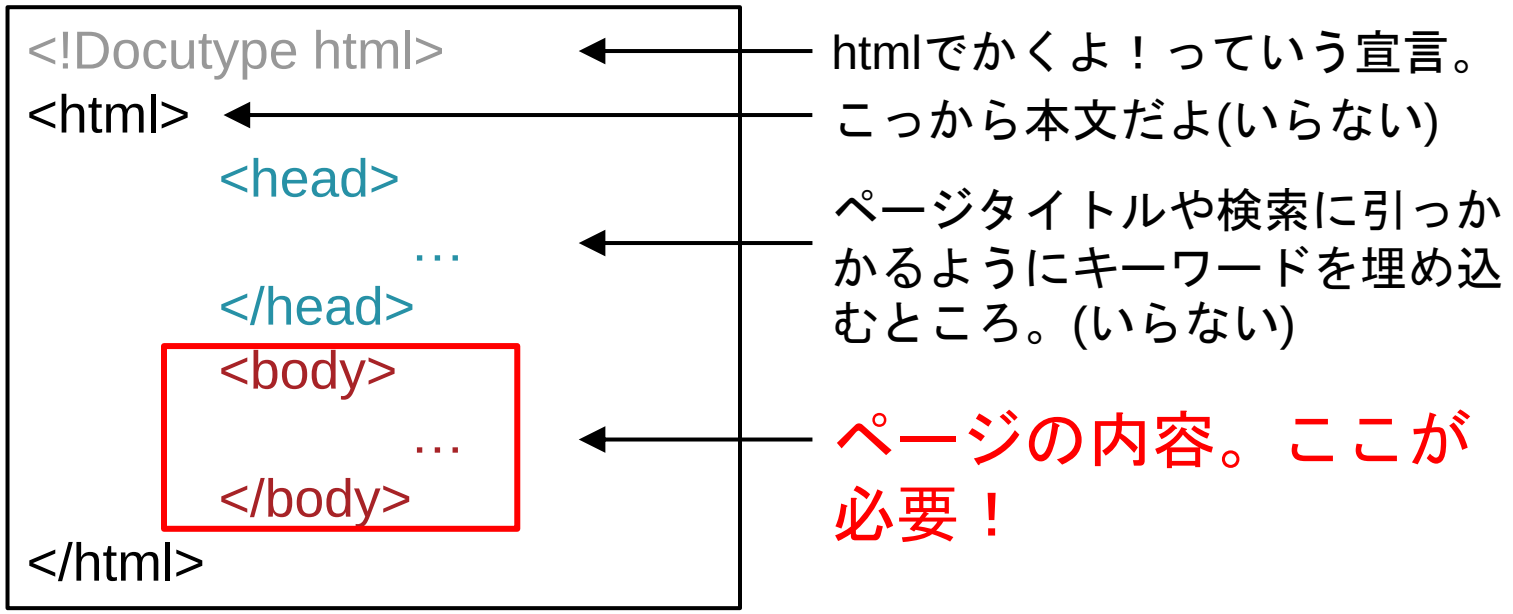
Htmlを読む。

じゃあ、yahooニュースにアクセスして適当な空白のところで右クリック、
「ページのソースを表示」を押してみよう。

うん。いみわかんねぇな。

大丈夫。まあ、落ち着けよ。

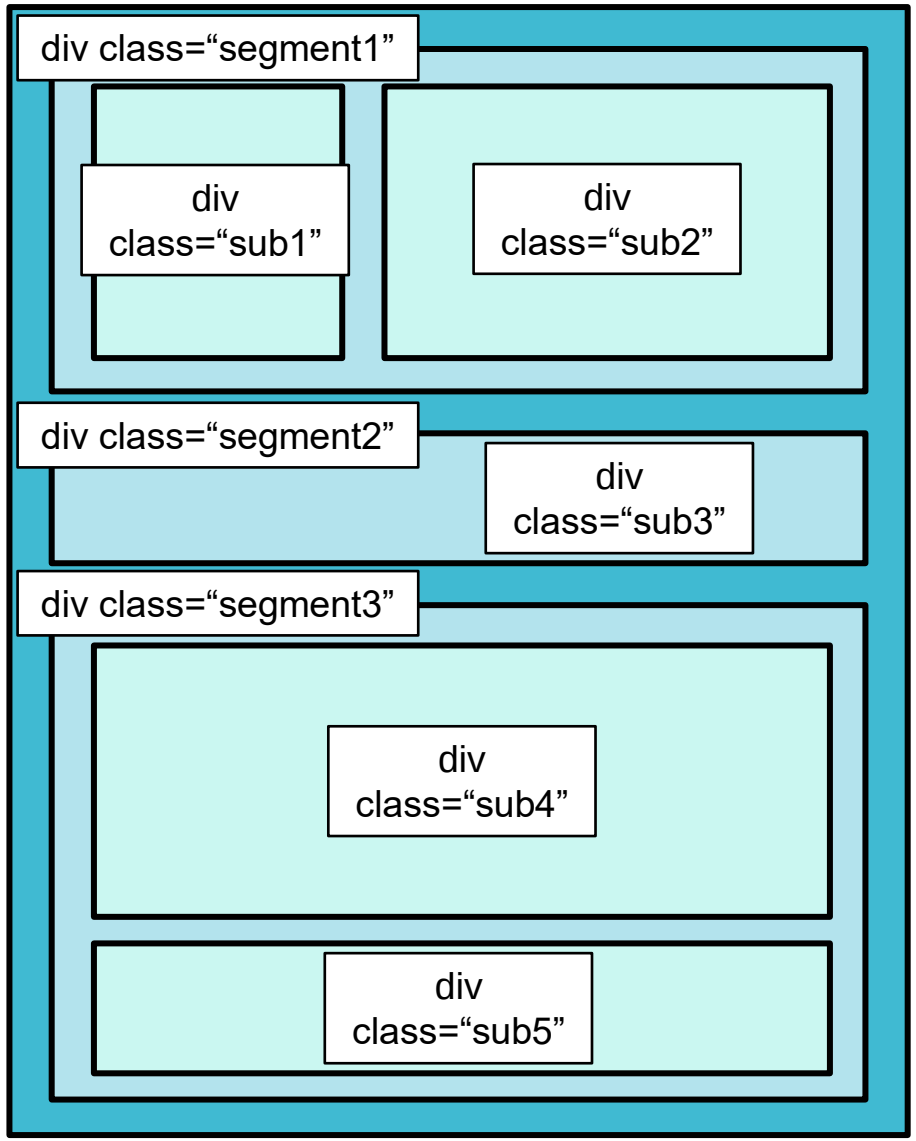
htmlの基本構造



つまり、半分位捨てていい。

Htmlを読む。Bodyの中身

ページのブロック構成



ページのhtmlコード

```
<body>
  <div class=“segment1”>
    <div class=“sub1”>…</div>
    <div class=“sub2”>…</div>
  </div >
  <div class=“segment2”>
    <div class=“sub3”>…</div>
  </div>
  <div class=“segment3”>
    <div class=“sub4”>…</div>
    <div class=“sub5”>…</div>
  </div >
</body>
```

基本は入れ子構造。

Htmlを読む。divの中身

基本以下の６つが分かればよくね、と思う。

Anchor “a” tag (<a>…): URLとかの情報を付加するためのテキストを作るタグ。参考文献は「こちら」とかで別のページに飛ぶじゃん？あれ。所謂ハイパーリンク。

Paragraph “p” tag (…): ただの文章。長い、クリックできない文章があるとしたらこれ。

Heading “h” tag (<h>…</h>): 大体単語か短い文。よくh1とかh2みたいに数字がつく。ほしい情報はけっこうこれだったリする。

Image “img” tag (<img src=… >): 画像を表示するためのタグ。それだけ。

List “li” tag (
…): いくつかの項目のリストの細項目を表示するタグ。ほしいリストの項目はこれであらわされていることが多い。

Table “td” tag (<td>…</td>): 表の各カラムのデータを表すタグ。<li>タグと同じくらい対象にする。

スクレイピングの作業フロー

1. Seleniumとgoogle.driverを起動してpyhon上でブラウザを操作する準備をする
2. URLを指定し、目的のデータが存在するページにアクセスする。
3. 必要であればdriver.find_elementでクリックや検索ワードの入力を行いページの加工を行う
4. BeautifulSoupでhtmlコードを取得する。
5. beautifulSoupでコード内の必要なセグメントを検索して抽出する
6. さらにその内層にあるデータを検索し、listや直接csvに加工してメモリに一時保存する。
7. データをDLする。

```
<!Docutype html>
<html>
 <head>…</head>
 <body>
  <div class="segment1">
   <div class="sub1">…</div>
   <div class="sub2">
    <h1> テーブルタイトル</h1>
    <p> テーブル内容の説明</p>
    <table>
     <tr>
      <td> data1 </td>
     </tr>
     <tr>
      <td> data2 </td>
     </tr>
     <tr>
      <td> data3 </td>
     </tr>
    </table>
   </div>
   <div class="sub3">…</div>
  </div >
 </body>
</html>
```

実際書いてみたコード

- **目標**: neo4jで使用するためのタンパク質と関連疾患のノードとリレーションデータの作成
- **データソース**: The human protein atras (人体を構成するタンパク質の臓器ごとの局在とか濃度、他データベースとのリンクとか)、UniProt (タンパク質ごとの機能、疾患との関連、自然発生する変異体、細胞内での局在、構造等の総合的なデータベース)、BioGrid (タンパク質間相互作用のデータベース)
- **なんか工夫した点**: 基本コピペ (これができるからpythonはいい)

ブラウザで説明するわ。Pptではまとめられねぇわ