Purpose: The data being collected and verified is important as it will be used for traffic generation patterns that you may see one day.

**At a high level, how does the given code work?** :

“””

{<website\_url>:

{

‘Specifics”:<DATA>

“main\_menu”:<DATA>

Endpoints:{

login-page: finder\_type;find\_string

}

“Sub-endpoints”:{

Login-page: finder\_type;find\_string

}

}

}

“””

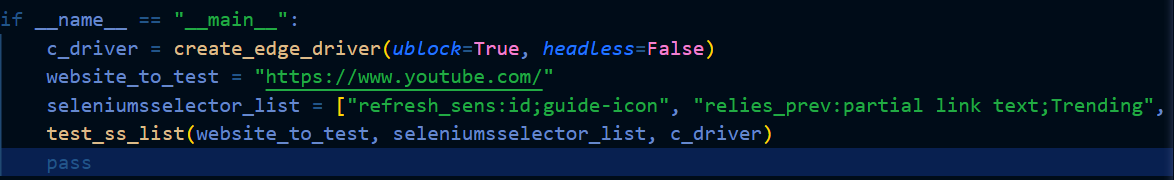


Data structure has a few main points, the website url, any additional information, the main menu endpoint (if it exists), main endpoints, and sub endpoints.

The identifiers for the endpoints also exist in the sub-endpoints so that the “current location” can be passed in as the dictionary key when selecting a branching sub-endpoint. The Data structure above and provided is the required result for each given website and must be verified thoroughly.

The provided testing code works in a few parts, first is the driver creation, each function does as described. The second function’s purpose is, given a url, a list of selectors and identifiers, and a webdriver to test the list of selectors sequentially.

The code would be utilized as below



Where ublock is if ad block is enabled, headless is if the webpage GUI is displayed

Where the list of selector values is the sequential path to take

Note that there is extra data per selector

“refresh\_sens:” means that once this element is clicked, that if the website is refreshed after the element is clicked that the state of the webpage is lost, and that it must be reclicked to return to the original state.

“relies\_prev:” means that the website is required to be in a specific state that is set by the previous action.

“rand\_ind:” for sequential elements, specifies that it should choose a random index and select that, for example, a random video on the trending page.

“ind\_X:” for sequential elements, specifies that it should choose the specified index ‘X’ and select that

The extra data can be chained together as such

<data1>~<data2>~<data3>:

Eg. relies\_prev~refresh\_sens~ind\_4

**How do I make the selector values?**

The first step is opening the webdriver of the target browser (Use Edge or **Chrome**)

Once you have the webpage for the target browser and web-page. Then right click on the target element and click “inspect”, sometimes there is a top layer that stops this, simply click inspect element on the target element again.

There are 9 different methods of finding an element in code.

| find\_type | Example selector string |
| --- | --- |
| id | In this example, you can see 2 different values for the “id” of the youtube button tablist. Both should be tested the lower level one (the more indented element) is likely correct  Eg: id;guide-icon |
| partial link text | In this example you can see the text between the ‘>’ ‘<’, this is the text that is being searched for with this selector.  Eg. partial link text;Trending |
| css selector | In this example you can see that inspecting the element went to too low of a level, this requires trial and error, but, for the highlighted element it is part of a repeating video element. And with this css selector, it will return a list of elements with that selector and with the “rand\_ind:” option, it will choose one of the repeating elements at random  Eg. rand\_ind:css selector;ytd-video-renderer |
| name | In this example, you can see the “name” variable, the “name” selector searches the webpage for references of this name variable and will match it  Eg. name;search\_query |
| tag name | Class names may be avoided, some websites generate class/tag names on run time |
| class name | Class names may be avoided, some websites generate class/tag names on run time |
| xpath | Xpath should be avoided as they change per window size/ current state. These are very prone to changing. |
| direct-link | Eg. direct-link;signin  (equiv, to “<webpage>/signin”) |
| link text | … use partial link text |

Limitations: Cannot handle a refresh-sensitive and indexed element.

First install vscode, then python, then git, then the browsers

Set the github username and email using these commands:

git config --global user.name "USERNAME"

git config --global user.email "EMAIL"

<https://github.com/Filamentous/PVT_SUP>

General Steps to take:

1. Go to webpage
2. Identify if there is a main menu, if it refresh sensitive, and identify any other starting endpoints
3. For each of the indentified endpoints create a list of sub endpoints
4. Recurse back to step 3 for every sub endpoint, dont memory leak :)
5. Test the elements until it is working reliable with the determined selectors, using the “test\_ss\_list()” function.
6. Build out the tree of endpoints in the data structure as necessary
7. Use the “test\_struct()” function to check for syntax errors within the data structure.
8. Repeat for given website