Automatic Web Navigator (AWN)

Created by: Austin Moore

Table of Contents

Introduction....3
Definitions (Read this first)....4
Installation....6
Understanding the Queue System....8
Adding to Website Queue....9
Insert to Website Queue....10
Remove from Website Queue....11
Adding to Action Queue....12
Insert to Action Queue....13
Remove from Action Queue....14
Creating the Website-action Queue....15
Insert to Website-action Queue....16
Remove from Website-action Queue....17
Running the Website-action Queue....18
Saving a Queue....19
Loading a Queue....20
Printing/Clearing a Queue....21
Writing a Companion Script....22
Good Resources for Python and Selenium....24

Introduction

AWN is designed to automate many repetitive web tasks that would take a significant amount of time to do by hand. AWN allows a user to queue multiple websites and perform repetitive tasks on them with an automated web browser such as Chrome or Firefox.

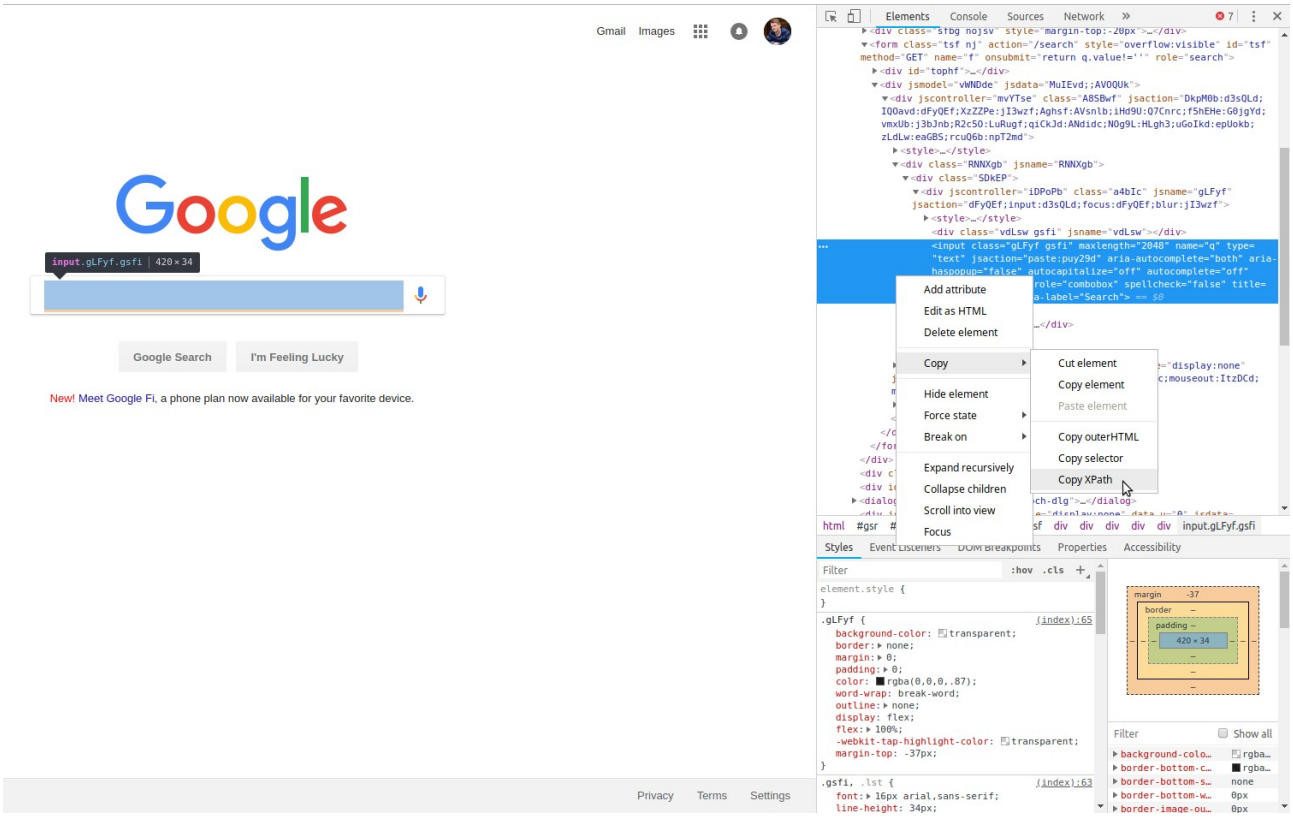
Since the user agent looks exactly like a regular web browser and can run JavaScript, AWN is able to access any website without being blocked.

Definitions (Read this first)

It is important to understand these terms before reading the rest of this manual.

XPath: Stands for “XML Path Language.” You can use XPath to find elements in the DOM. More about it here.

Most browsers will allow you to copy the XPath from the web developer tools:



Selenium: Tool for driving/automating the web browser. Originally designed for website testing but can also be used to automate many other repetitive tasks. Selenium has bindings for Python, Java, C#, Ruby, and Node.js.

Web Driver: Certain browsers such as Google Chrome and Mozilla Firefox can be automated through a web driver download. You can find the downloads here.

Companion Script: The term I use for denoting Python scripts that can be used to extend the functionality of *navigator.py*. This is the most powerful feature of the program and will be further explained in a later section. Adding a Python script allows *navigator.py* to import the script_main function of the specified Python script, pass the current web driver, and run the script_main function. This allows for extended functionality such as copying data from an Excel file and then pasting that data into a website.

Installation

AWN does not yet come packaged with an installer. As of right now, you will need to install Python on your system and run the *navigator.py* script. The following software is required to run the program.

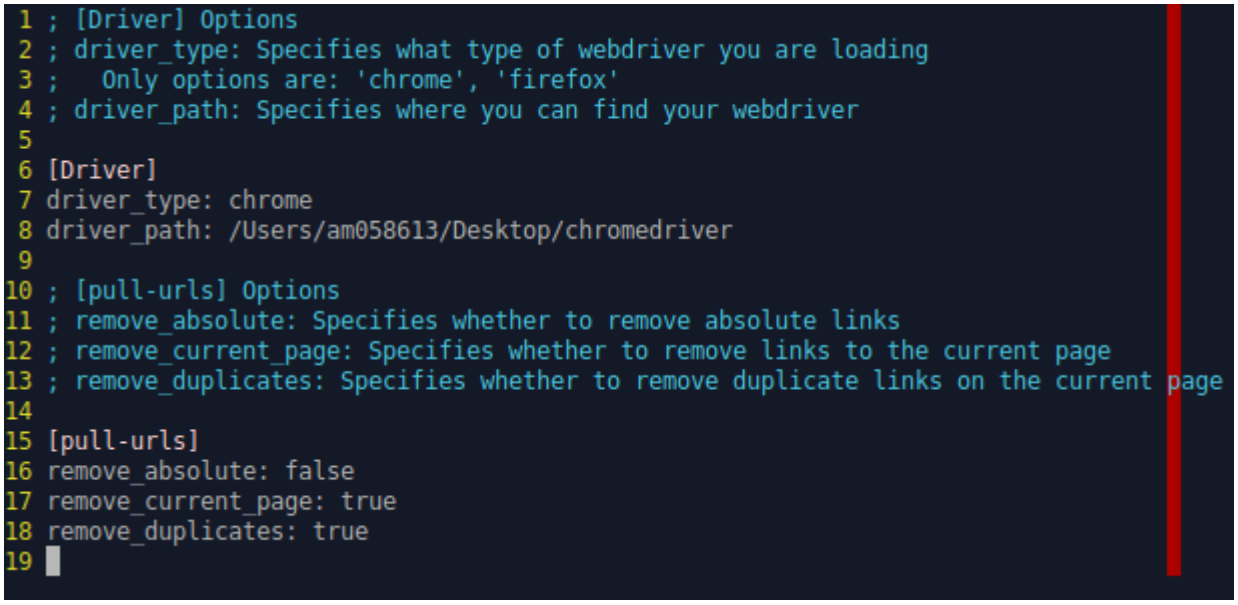
- Python 2
- Python 2 pip
- Python 2 modules
  - selenium

    pip install selenium
  - bs4

    pip install bs4
- Web Driver (One of the following)
  - Chrome Driver
  - Firefox Driver

Python 2 is used in place of Python 3 since the machine I have at work only comes installed with Python 2. In the interest of not making too many download requests at work, I have opted to stick with Python 2 for the time being.

You will also need to configure the *config.ini* file to tell it what driver you are using and where *navigator.py* can find the web driver.



```
; [Driver] Options
; driver_type: Specifies what type of webdriver you are loading
;   Only options are: 'chrome', 'firefox'
; driver_path: Specifies where you can find your webdriver

[Driver]
driver_type: chrome
driver_path: /Users/am058613/Desktop/chromedriver

; [pull-urls] Options
; remove_absolute: Specifies whether to remove absolute links
; remove_current_page: Specifies whether to remove links to the current page
; remove_duplicates: Specifies whether to remove duplicate links on the current page

[pull-urls]
remove_absolute: false
remove_current_page: true
remove_duplicates: true
```

You can leave the [pull-urls] section like it is. It is the configuration options for *pull-urls.py* which will not be covered in this manual.

Understanding the Queue System

AWN comes with three different kinds of queues. There is the website queue, action queue, and the website-action queue.

The website queue is a queue of all websites we will eventually like to perform specified actions on.

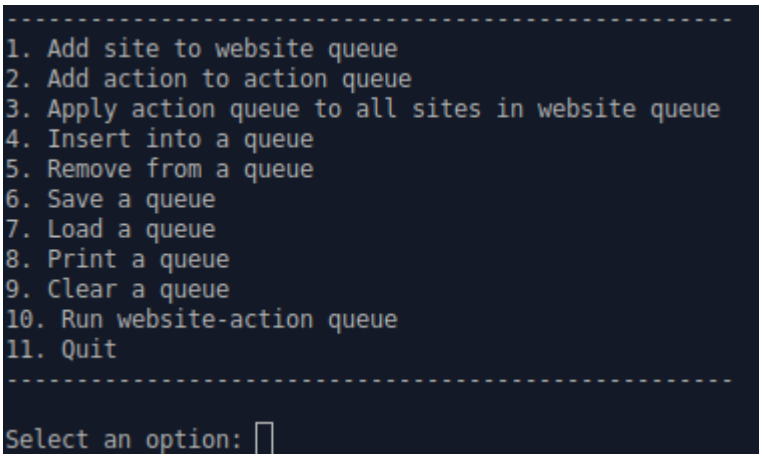
The action queue is a queue of all actions we will eventually like to perform on the website queue.

The website-action queue is the combination of a website queue and action queue when all actions have been applied to the website queue. This queue will be the queue the web browser will follow when it executes all of the actions we wish to perform on our specified pages.

The user is able to add, insert, and remove websites from the website queue at any time in the *navigator.py* program. The user is also able to add, insert, and remove actions from the action queue at any time. Finally, the user can also add, insert, and remove any websites or actions in the website-action queue at any time.

Adding to Website Queue

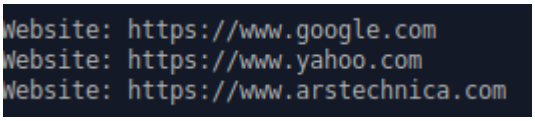
Upon starting the *navigator.py* program, the user will be greeted with this menu:



To add a website to the website queue:

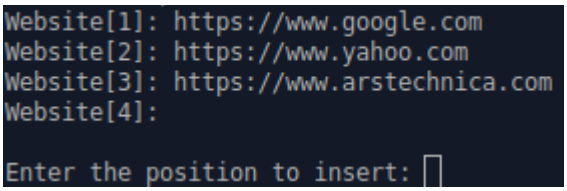
Type “1”, ENTER, and then enter the name of the website.

From the main menu, if you go to “Print a queue”, you will be able to print the website queue and all sites in it:



Insert to Website Queue

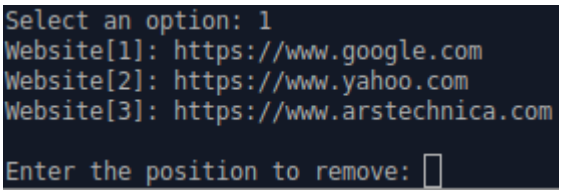
From the menu, if you go to “Insert into a queue” and then go to “Insert into website queue”, you will be given a list like this:



From here, you can enter a position to insert a new website into the website queue. For instance, typing “1” will give you the option to insert a website *before* the website currently in position “1”. Typing “4” will insert a website at the end of the list.

Remove from Website Queue

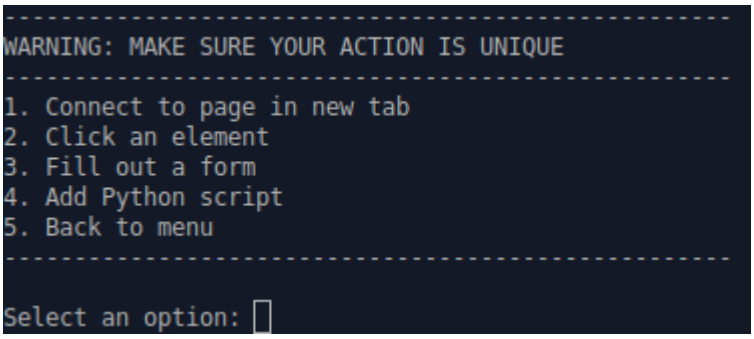
From the menu, if you go to “Remove from a queue” and then go to “Remove from website queue”, you will be given a list like this:



Typing in the position will remove the corresponding website from the website queue.

Adding to Action Queue

From the menu, if you go to “Add action to action queue,” you will be prompted with this:



The following options are described:

Connect to page in new tab: Connect to the current web page in a new tab

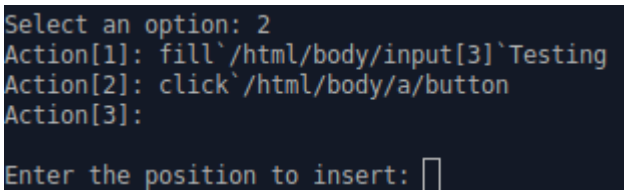
Click an element: Click an element in the DOM. You will be prompted to enter an XPath for the element to click.

Fill out a form: Fill out an element in the DOM. You will be prompted to enter an XPath for the element to be filled out. After entering the XPath, you will be asked what you wish to enter into the element.

Add Python script: Allows the ability to add a companion script to the program.

Insert to Action Queue

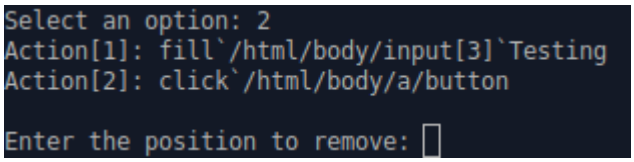
From the menu, if you go to “Insert into a queue” and then go to “Insert into action queue”, you will be given a list like this:



From here, you can enter a position to insert a new action into the action queue. For instance, typing “1” will give you the option to insert an action *before* the action currently in position “1”. Typing “3” will insert an action at the end of the list. Note that the back tick in all actions is just the way *navigator.py* separates the action, element, and input.

Remove from Action Queue

From the menu, if you go to “Remove from a queue” and then go to “Remove from action queue”, you will be given a list like this:



Typing in the position will remove the corresponding action from the action queue.

Creating the Website-action Queue

The main way to create the website-action queue is to apply all the actions from the action queue to the website queue. This is option “3” in the main menu. A complete website-action queue will look like this if you decide to print it:

```
Website: file:///Users/am058613/Documents/awn/example-files/example1.html
    Action: fill`/html/body/input[3]`Testing
    Action: click`/html/body/a/button
Website: file:///Users/am058613/Documents/awn/example-files/example2.html
    Action: fill`/html/body/input[3]`Testing
    Action: click`/html/body/a/button
Website: file:///Users/am058613/Documents/awn/example-files/example3.html
    Action: fill`/html/body/input[3]`Testing
    Action: click`/html/body/a/button
```

Insert to Website-action Queue

From the menu, if you go to “Insert into a queue” and then go to “Insert into website-action queue”, you will be given a list like this:

```
Website[1]: file:///Users/am058613/Documents/awn/example-files/example1.html
    Action[1]: fill`/html/body/input[3]`Testing
    Action[2]: click`/html/body/a/button
    Action[3]:

Website[2]: file:///Users/am058613/Documents/awn/example-files/example2.html
    Action[1]: fill`/html/body/input[3]`Testing
    Action[2]: click`/html/body/a/button
    Action[3]:

Website[3]: file:///Users/am058613/Documents/awn/example-files/example3.html
    Action[1]: fill`/html/body/input[3]`Testing
    Action[2]: click`/html/body/a/button
    Action[3]:

Website[4]:

Enter the position to insert:
```

From here, you can either add a new action to one of the websites in the queue OR add a new website to the queue. To add an action, you need to enter two numbers delimited by a space. For example, to insert a new action in position 2, to the second website, you would need to type “2 2”.

To insert a new website to the queue, you only need to enter one number. For example, to insert a new website to position 3, you need to type “3” and then specify the name of the website:

```
Website: file:///Users/am058613/Documents/awn/example-files/example1.html
    Action: fill`/html/body/input[3]`Testing
    Action: click`/html/body/a/button
Website: file:///Users/am058613/Documents/awn/example-files/example2.html
    Action: fill`/html/body/input[3]`Testing
    Action: click`/html/body/a/button
Website: https://www.google.com
Website: file:///Users/am058613/Documents/awn/example-files/example3.html
    Action: fill`/html/body/input[3]`Testing
    Action: click`/html/body/a/button
```

Remove from Website-action Queue

From the menu, if you go to “Remove from a queue” and then go to “Remove from website-action queue”, you will be given a list like this:

Removing works similarly to inserting. To remove an action from one of the websites, you need to enter two numbers specifying the site and then the action to remove from it. For example, to remove the second action from the second website in the queue, you need to type “2 2”.

To remove a website *and all actions underneath,* you only need to type the number of the website you wish to remove.

Running the Website-action Queue

Option “10” in the main menu will allow you to run the website-action queue. This will start the web driver specified in *config.ini* and execute all actions in the queue. You will see this happening live in the browser you specified.

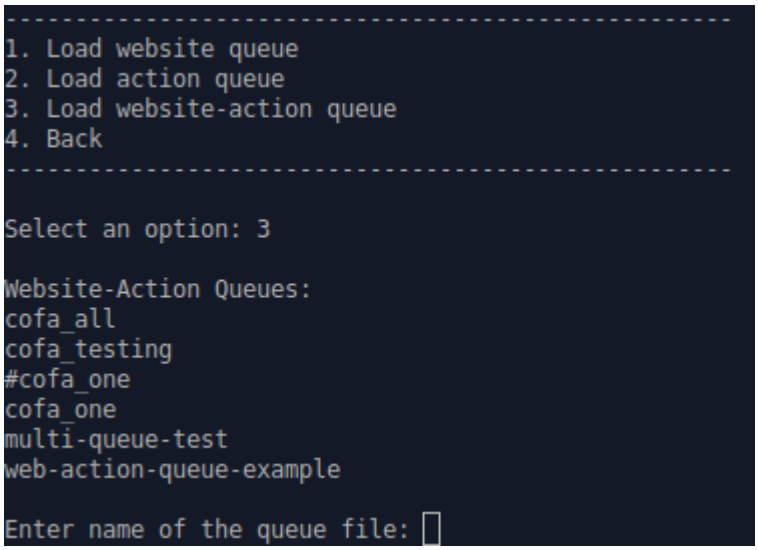
Saving a Queue

You are able to save any of the three queues by navigating to “Save a queue” in the main menu and specifying which queue you would like to save. You can then name the queue. The queue will be saved as *name_of_queue.wq* for a website queue, *name_of_queue.aq* for an action queue, or *name_of_queue.waq* for a website-action queue.

Website queues, action queues, website-action queue, are saved under the web-queues, action-queues, web-action-queues directories respectively.

Loading a Queue

You are able to load any of the three queues by navigating to “Load a queue” in the main menu and specifying which queue you would like to load:



navigator.py will provide a listing of the type of queues you specified to be loaded. Do not attempt to type the file name extensions *.wq, .aq, or .waq. navigator.py* will add the file extension behind when it goes to import the specified queue.

Printing/Clearing a Queue

The “Print a queue” and “Clear a queue” options on the main menu are pretty self explanatory. You can print any of the three queues. You can also clear *all* of the entries in any of the three queues.

Writing a Companion Script

From the “Definitions” section: “Adding a Python script allows *navigator.py* to import the script_main function of the specified Python script, pass the current web driver, and run the script_main function. This allows for extended functionality such as copying data from an Excel file and then pasting that data into a website.”

There are a few things to know when writing a Python companion script. Here is the code where *navigator.py* imports the script_main function and runs it:

```
func = importlib.import_module(import_module).__getattribute__("script_main")
driver.get(web_action_queue[index][0])
func(driver, web_action_queue[index][0], index)
```

The first line is importing the script_main function from your module (your companion script). The web driver *navigator.py* created then connects to the next page in the website-action queue, then calls your script_main function, passing it the web driver, the web page from the website-action queue, and then the current index in the website queue. This is indexed at zero.

When you go to create your companion script, you must save the companion script in the root of the AWN directory. *cofa_content_grab.py,* and *mod_page_creation.py* are both companion scripts for example. You will need to import the following modules before writing the script_main function:

```
from selenium import webdriver
from selenium.webdriver.support.ui import Select
```

Next you will need to write the script_main function. You **can not** change the name of the function and you **must** have three arguments like so:

```
def script_main(driver, url, pos):
```

The first parameter is for the web driver that will eventually be passed to the function. The second parameter is the URL we originally wanted to connect to. The reason for the second parameter is that when *navigator.py* connects to the page and then calls your script_main

function, there is a possibility the driver will redirect to another page. For example, if you tried to connect to a CMS, but got redirected to a login form, your script_main function will need to complete the form and then connect to the originally intended page. An example can be found in *mod_page_creation.py*:

```
def script_main(driver, url, pos):
    cas_username_xpath = "//*[@id='username']"
    cas_password_xpath = "//*[@id='password']"
    cas_login_button_xpath = "/html/body/div[1]/div[2]/div/form/section[3]/div/button[1]"

    if (re.findall(r"cas.sso.ohio.edu", str(driver.current_url))):
        username = raw_input("Enter OHIO username: ")
        password = getpass.getpass("Enter OHIO password: ")

        element = driver.find_element_by_xpath(cas_username_xpath)
        element.send_keys(username)
        element = driver.find_element_by_xpath(cas_password_xpath)
        element.send_keys(password)
        element = driver.find_element_by_xpath(cas_login_button_xpath)
        element.click()

    driver.get(url)
```

The URL is passed as a string and the position in queue (indexed at zero) is passed as an integer.

These are all the specifics of setting up a Python companion script. From here, you can use your Python and Selenium knowledge to automate anything across all the pages in the queue.

Good Resources for Python and Selenium

If you want to learn to make a companion script, I would recommend learning the basics of Python, then taking a look at regular expressions, and then moving on to learning Selenium from trial and error. Once you learn the basics of Python, you can pretty much just take glances at the Selenium bindings and do Google searches whenever you want to learn more about driving the browser.

Basic Python tutorials playlist from sentdex:

https://www.youtube.com/watch?v=oVp1vrfL_w4&list=PLQVvvaa0QuDe8XSftW-RAxdo6OmaeL85M

I would say you should learn up to at least the “Function Parameters” tutorial video. The first two videos of the playlist are up to you really. Note that everything in this project should be written as Python 2 code but pretty much everything in the Python 3 basic tutorial series is the same. I would say the main difference is that I will use the raw_input function from Python 2 instead of the input function from Python 3.

Regular expression tutorials from sentdex:

https://www.youtube.com/watch?v=sZyAn2TW7GY

https://www.youtube.com/watch?v=GEshegZzt3M

Selenium Python Bindings (IMPORTANT)

https://selenium-python.readthedocs.io/