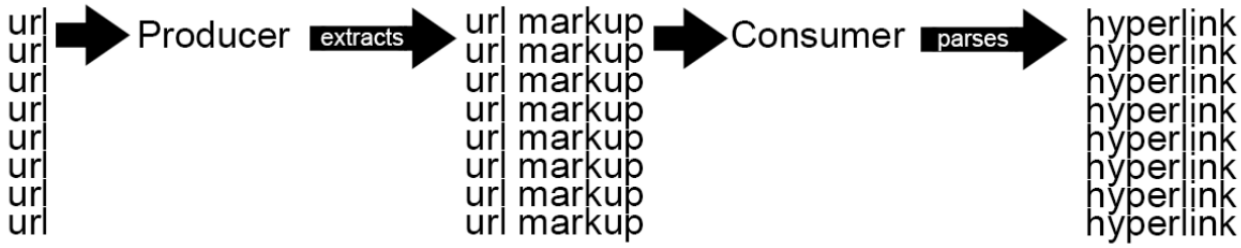
A Simple Producer/Consumer Web Link Extractor

Summary
The consumer receives a queue of markups from the producer, which gets those markups by extracting them from a list of URLs. The HTML is then parsed and the relevant information is extracted, be it certain email addresses or personal details. This new list is compiled into a file.

Description
The aim of the producer/consumer web link extractor is to collect hyperlinks from a list of URLs given to the producer. The producer will extract all useful information from this list of URLs, and if it encounters an error or a URL with no useful information it will discard it to a separate list to make sure it is not checked again. For these lists I will use a for loop to iterate through. I could use a .csv file to give the producer the list of URLs. The consumer will examine this new list and parse through it to find hyperlinks on this URL to add to the final list. The spider that does this will make sure to delete all completed URLs from the previous queue to make sure that they are not parsed again. Unit tests can be completed along the process to make sure that each individual process is running correctly, and problems with the code can be isolated and solved. An example of a unit test I could do is insert a known URL into the function and check that the output hyperlink is what I expect it to be.

Flowchart



Code

As I am not able to write my own code, I have researched code that could be used for this project. I have added my own annotations to this source code.

```
@staticmethod
    def add_links_to_queue(links): # the function that will take
a set of links and add them to the waiting list
        for url in links: # loops through the set
            if (url in Spider.queue) or (url in Spider.crawled):
# checks if links are already in the waiting or the crawled list
                continue
            if Spider.domain_name != get_domain_name(url): #
checks if the domain name is present in the URL.
#This ensures that the crawler will crawl only pages on the
targeted website, and not the external links present on the
website.
                continue
            Spider.queue.add(url) # adds link to the waiting
list

    @staticmethod
    def update_files(): # updates the files
        set_to_file(Spider.queue, Spider.queue_file)
        set_to_file(Spider.crawled, Spider.crawled_file)

https://geek-university.com/add-links-to-queue/
```

This section of code is adding the links to the queue, it does this by creating a for loop to iterate through the list then checking if the URL has already been crawled or not already. If it has been crawled it is ignored. With the links it finds it makes sure the domains are the same to make sure they are relevant URLs that are being found. The links are then added to the next queue.