Next, you have to cycle through the different pages to scrape news from them. You do this by using a `for` loop and adding a dynamic `changing_slug` whose value depends on the iterations of the loop. This helps to create the URL, which you call using `requests.get()` to capture the page's HTML to the `data` variable.

You then call the `BeautifulSoup` parser, which iterates through all the `<h2>` tags and `<p>` tags with class `intro` to create two separate lists for headlines and news. You can remove the two unwanted `<h2>` tags by slicing the headline list appropriately.

The next step involves editing the contents of both lists to remove parentheses and quotes and make the text lowercase. Then you create a dictionary that stores the headlines and news as key/values pairs. Finally, you build the `news_reader()` function, which iterates through the items in the dictionary, and have Melissa speak the headlines and news via the `tts()` function.

■ **Note** This approach is vulnerable to changes in the site's HTML. I used the data-mining approach to demonstrate how you can retrieve information if no other alternative is available. Using the official API or a web site's RSS feed will solve this problem.

The last step is adding the appropriate information to `brain.py`:

```
from GreyMatter import tell_time, general_conversations, weather,
define_subject, business_news_reader
```

Now add the corresponding code snippet to the `if/else` clause:

```
elif check_message(['business', 'news']):
    business_news_reader.news_reader()
```

Congratulations—you have just built a business news reader for Melissa! I am sure she is grateful. You can call this functionality by saying a command like, "Read me the business news!" or "Latest business news!" Let's revise the workflow you follow to obtain the news from the news web site and process it in such a way that it is appropriate to pass to `tts()`; see Figure 4-4.

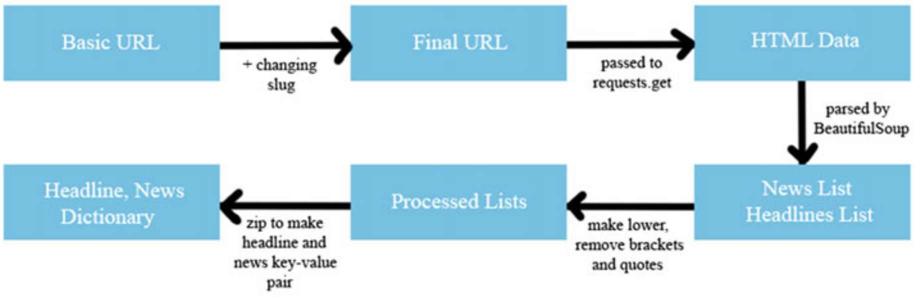

Figure 4-4. *News-retrieval workflow*