Changes: ssl error some people were experiencing and a solution in second paragraph.

Internet Problem
Before attempting to work on this problem, make sure you have gone through the corresponding part in module 8. In particular make sure you understand how to open a connection to read the file and how to load it in memory.

An error some people have reported is a rejection of the certificate the website is using. If that is your case, the following code might fix it by disabling the certificate requirement:

```
import urllib.request
import ssl  # library to handle security in IP

url = "http://www.gutenberg.org/cache/epub/28/pg28.txt"
print("Connecting...\r", end = "")
# Modifying some settings urllib.request will use.
ctx = ssl.create_default_context()
ctx.check_hostname = False  # Disables hostname checks.
ctx.verify_mode = ssl.CERT_NONE  # Disables certificate checks.

file = urllib.request.urlopen(url, context=ctx)
```

Parsing a file that is not designed to be parsed is usually tedious. However, this file has enough elements to make parsing manageable. There are many approaches to parsing. In this case, we will use one that processes all the lines in the list of read lines. (You can use a different one. I chose this one because it is intuitive enough so that I can explain it in a relatively easy manner.)

Make a `for` loop that iterates through the list of lines. We will move through the file identifying
- the beginning of the table of contents (and then)
- the end of the table of contents (and then)
- the beginning of the fables (and then)
- the title of the first fable (and then)
- the body of the first fable and
- the titles of the following fables (and then)
- the body of the fables (and then)
- the end of the each fable or the end of the whole fable section.

Notice that these elements are sequential in the file.

For this purpose, we will set up an `if-elif-else` statement that will move from condition to condition using flags. When working, add to this `if-elif-else` statement one condition at a time while testing to make sure you are identifying the appropriate places in the file. For example, before we reach the beginning of the table of contents, a flag `contents_found`, initialized to `False`, indicates that we have not reached the appropriate line. A line of text in the list of lines from the file containing the word "`Contents`" by itself marks the beginning of the table of contents. When we reach that line the flag `contents_found` would become `True`.

Let's illustrate this with the appropriate code. In what follows, the first part of the `if-elif-else` statement will catch all lines before the table of contents is found: make sure you understand why.

```
contents_found = False
for line in clean_data:
    if not contents_found:  # catches all lines before the Contents section
        if line == "Contents":  # Checks if we have reached it
            print(f"Table of contents found.")
            contents_found = True  # If so, we move on
    elif ...: pass
```

Notice how every line is captured by this condition until the right line is found. After that condition is satisfied, this section of the `if-elif-else` statement is never used again: `not contents_found` now evaluates to `True`. It will be what we put in the `elif` statement that gets evaluated from now on.

Let's move on. Importantly there are a few empty lines in the internet file after the "`Contents`" line. These lines do not correspond to titles and should not be added to the titles list. However, when the first title is identified something changes. After that moment all non-empty lines will be assumed to correspond to titles and should be appended to the list of titles. At this point a new condition is satisfied: the first title has been found. This can be addressed with a flag or by the fact that the list of titles ceases to be empty. Again this condition is handled by the `elif` statement above. Pause until you understand what you have accomplished. The strategies below are variations of this strategy.

Now, after the first title has been identified, all the non-empty lines that follow will be assumed to be titles. When a new empty line is found, that will indicate the end of the Contents section. The list of titles will be then complete. That is the time to move on to the next task. Notice that this can also be handled by yet another `elif` statement using a new flag.

Once the list is ready, we move on to find the beginning of the fables. This is marked by the line described in the assignment for that purpose: "`Aesop's Fables`". Once that line is found, we move on to find the bodies of the fables. Again this can be handled by another `elif` statement with an appropriate flag. Notice that we now switch to finding the first fable.

When searching for the first fable, we compare each line with the contents of the title list. We know that the titles are present in dedicated lines immediately preceding the body of the fable. These strings will be in the list of titles. This search can be handled with another `elif` statement. When a title is found, the text that follows will be assumed to be the body of the fable.

The moment the first fable is identified, we have to take measures so that it can be stored appropriately. At this point you can choose how to store the pairs of titles and fable bodies. We then move on to identifying the bodies for the fables, including the body for this first one. This is done again by means of an `elif` statement in our ever growing `if-elif-else` statement. You will also need to keep all the lines corresponding to the fable together. (It could be a list, a dictionary, a string, …) Now, though we don't need more flags. The flag for the first fable is the last flag.

After the first fable is identified, we have to collect the lines corresponding to the body. Again this can be handled with an `elif` statement. However, we have to be on the lookout for the next fable or even for the end of the fables. Either one of those occurrences will mark the end of the fable and the beginning of the next one in the former case or the end of the fables section in the latter. This requires the fable we were working on be saved appropriately. In my case I chose the `else` of the whole `if-elif-else` statement to be the case when we are adding to the body of the fable. Hence the other cases will be handled with their respective `elif` statements. Importantly, this section handles the titles and bodies of all the remaining fables. In my case I check whether the line of text is in the title list and if so I save my work before moving to the next fable. I also check where we are at the end of the fables section: see the assignment document for how to assess that. If so, I save my work and break from the for loop.

At this point parsing is done.

What is missing to complete the task should not present major difficulties.