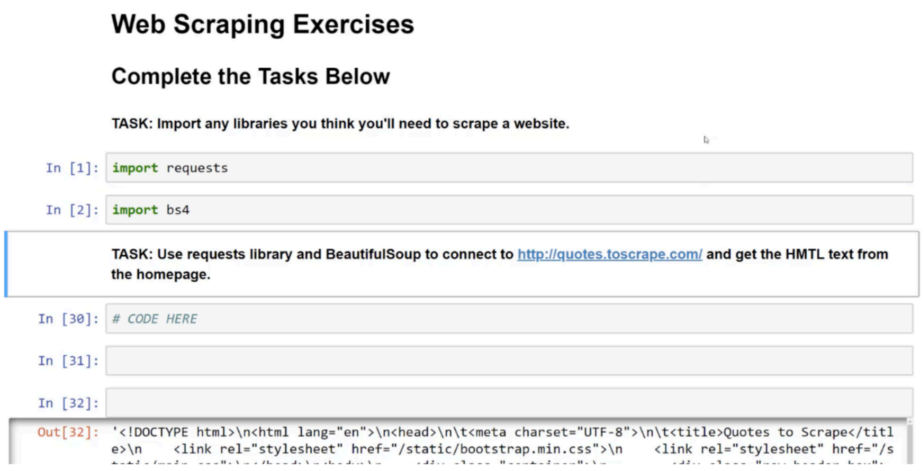
SECTION 15: WEB SCRAPING WITH PYTHON - 1 hour 40 minutes, 9 parts

9/9 Python Web Scraping - Exercise Solutions

- **web scraping solutions**
  - -> there are multiple different approaches to doing this
  - -> importing libraries we need
  - -> these are requests and bs4

  - **-> then connecting to the request website to scrape the information from**
    - -> this is done using the requests.get() method
    - -> the argument of this is the URL of the scraping website
    - -> he's then printed the information we've scraped
    - -> it's the HTML for the quote site from the previous video

  - **-> processing this information**
    - **-> checking we have the right information**
      - -> we have lists of quotes in the scraped information
      - -> it's quotes, and some of the information is repeated (some isn't)
      - -> we are figuring out what the correct class call is
      - -> he's converted the information to a beautiful soup object
      - **-> on the webpage which we've scraped from, he's inspected the html for the webpage**
      - -> then he's printed the scraped information from the webpage -> which is the html in the ipynb file, and this matches the inspected html on the webpage



Web Scraping Exercises

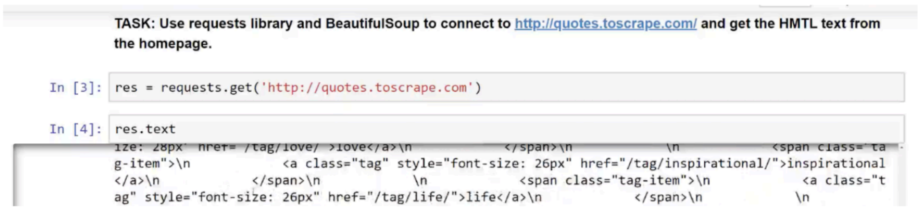
Complete the Tasks Below

TASK: Import any libraries you think you'll need to scrape a website.

```
In [1]: import requests

In [2]: import bs4
```

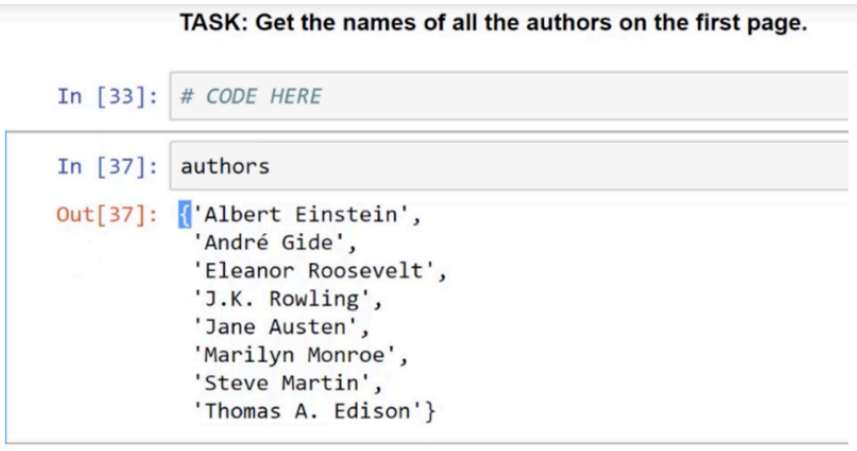
TASK: Use requests library and BeautifulSoup to connect to http://quotes.toscrape.com/ and get the HMTL text from the homepage.

```
In [30]: # CODE HERE

In [31]:

In [32]:

Out[32]: '<!DOCTYPE html>\n<html lang="en">\n<head>\n\t<meta charset="UTF-8">\n\t<title>Quotes to Scrape</titl
         e>\n    <link rel="stylesheet" href="/static/bootstrap.min.css">\n    <link rel="stylesheet" href="/s
```



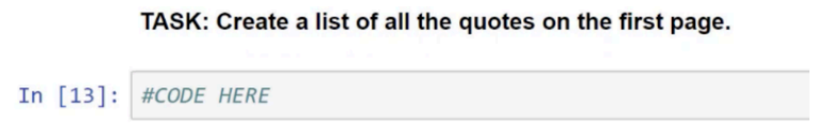
TASK: Use requests library and BeautifulSoup to connect to http://quotes.toscrape.com/ and get the HMTL text from the homepage.

```
In [3]: res = requests.get('http://quotes.toscrape.com')

In [4]: res.text
        ize: 28px" href="/tag/love/">love</a>\n            </span>\n            \n            <span class="ta
        g-item">\n            <a class="tag" style="font-size: 26px" href="/tag/inspirational/">inspirational
        </a>\n            </span>\n            \n            <span class="tag-item">\n            <a class="t
        ag" style="font-size: 26px" href="/tag/life/">life</a>\n            </span>\n            \n
```



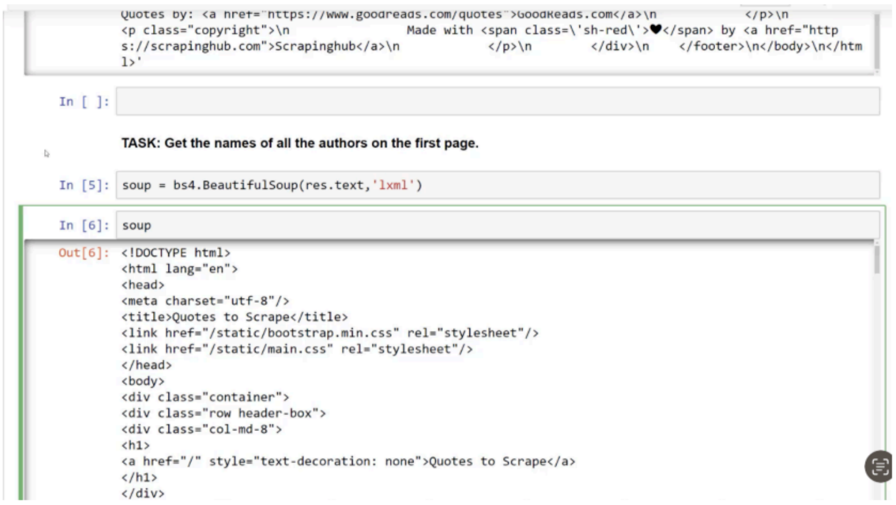
TASK: Get the names of all the authors on the first page.

```
In [33]: # CODE HERE

In [37]: authors

Out[37]: {'Albert Einstein',
          'André Gide',
          'Eleanor Roosevelt',
          'J.K. Rowling',
          'Jane Austen',
          'Marilyn Monroe',
          'Steve Martin',
          'Thomas A. Edison'}
```



TASK: Create a list of all the quotes on the first page.

```
In [13]: #CODE HERE
```

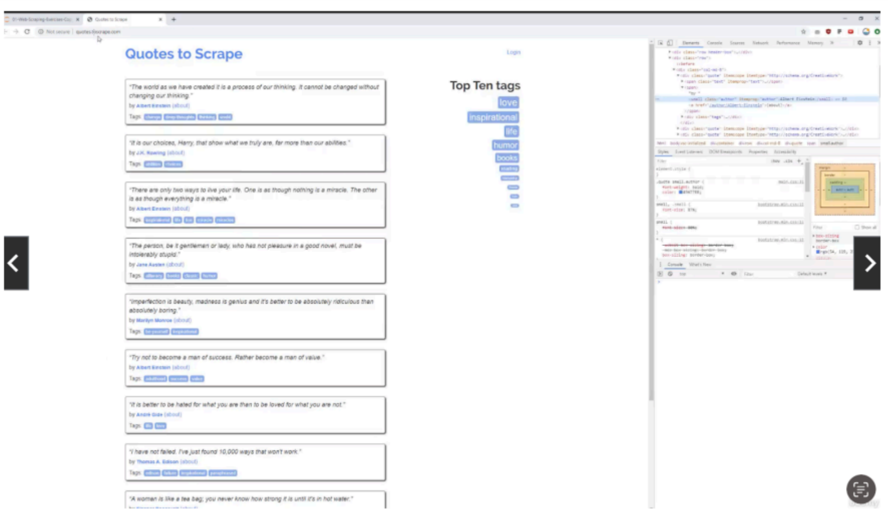


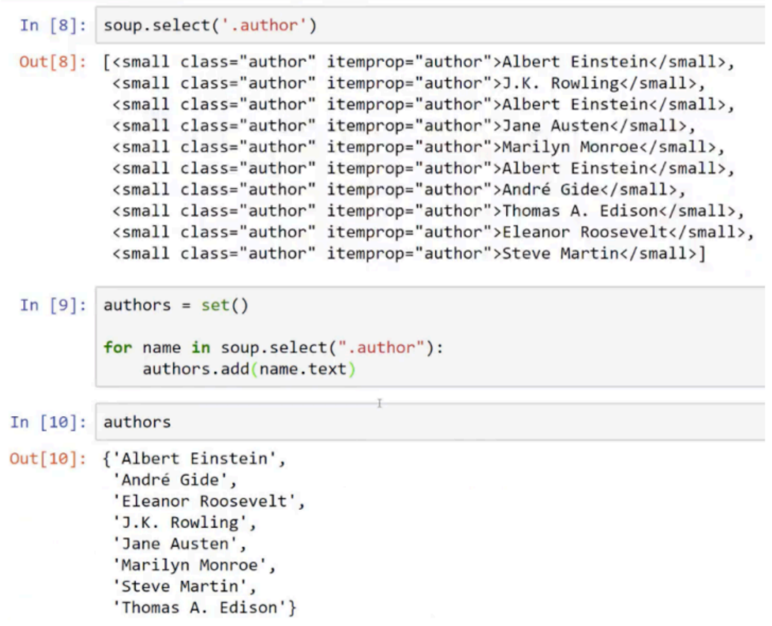
```
        Quotes by: <a href="https://www.goodreads.com/quotes">GoodReads.com</a>\n            </p>\n
        <p class="copyright">\n            Made with <span class=\'sh-red\'>❤</span> by <a href="http
        s://scrapinghub.com">Scrapinghub</a>\n            </p>\n        </div>\n    </footer>\n</body>\n</htm
        l>'

In [ ]:
```

TASK: Get the names of all the authors on the first page.

```
In [5]: soup = bs4.BeautifulSoup(res.text,'lxml')

In [6]: soup

Out[6]: <!DOCTYPE html>
        <html lang="en">
        <head>
        <meta charset="utf-8"/>
        <title>Quotes to Scrape</title>
        <link href="/static/bootstrap.min.css" rel="stylesheet"/>
        <link href="/static/main.css" rel="stylesheet"/>
        </head>
        <body>
        <div class="container">
        <div class="row header-box">
        <div class="col-md-8">
        <h1>
        <a href="/" style="text-decoration: none">Quotes to Scrape</a>
        </h1>
        </div>
```

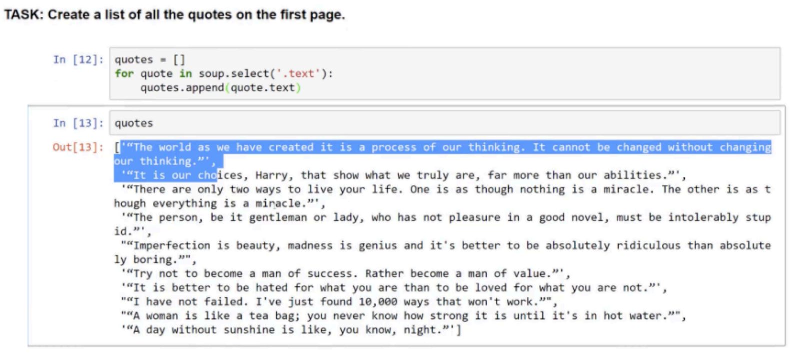
- **-> iterating through the information**
  - -> we are getting all of the names of the authors on the first page
  - -> we have a set
  - -> we want to take advantage of the set, to avoid adding duplicates to it
  - -> he has created a soup using beautiful soup and passed in res.text as one of the arguments
  - -> he's inspecting the elements on the webpage whose html we just imported, so that we know it's been correctly imported into the ipynb file
  - -> then the .select method can be used to extract information from the html
  - **-> a set is where the information isn't repeated <- he's implementing this into the code to avoid repeats**
    - -> he's iterating through the different names which have been extracted from the page, and making a set out of the authors





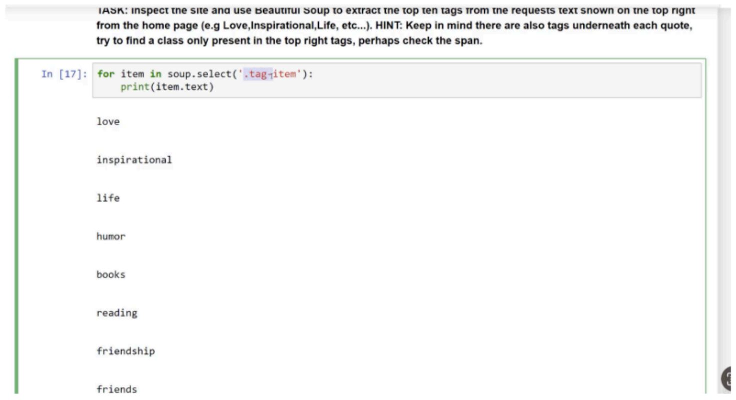
```
In [8]:  soup.select('.author')

Out[8]:  [<small class="author" itemprop="author">Albert Einstein</small>,
          <small class="author" itemprop="author">J.K. Rowling</small>,
          <small class="author" itemprop="author">Albert Einstein</small>,
          <small class="author" itemprop="author">Jane Austen</small>,
          <small class="author" itemprop="author">Marilyn Monroe</small>,
          <small class="author" itemprop="author">Albert Einstein</small>,
          <small class="author" itemprop="author">André Gide</small>,
          <small class="author" itemprop="author">Thomas A. Edison</small>,
          <small class="author" itemprop="author">Eleanor Roosevelt</small>,
          <small class="author" itemprop="author">Steve Martin</small>]

In [9]:  authors = set()

         for name in soup.select(".author"):
             authors.add(name.text)

In [10]: authors

Out[10]: {'Albert Einstein',
          'André Gide',
          'Eleanor Roosevelt',
          'J.K. Rowling',
          'Jane Austen',
          'Marilyn Monroe',
          'Steve Martin',
          'Thomas A. Edison'}
```

- **-> then making a list out of the quotes on the page**
  - -> he has inspected the html for the elements on the page which we want to extract
  - -> then using the .select() method, we have a list of the quotes on the page <- the html for them has been extracted
  - -> then he is putting them into an empty list
  - -> we already have a soup made out of the information
  - -> then soup.select() with an argument (a css class in this example), return the html which matches it
  - -> then he's iterating through it



TASK: Create a list of all the quotes on the first page.

```
In [12]: quotes = []
         for quote in soup.select('.text'):
             quotes.append(quote.text)

In [13]: quotes

Out[13]: ['"The world as we have created it is a process of our thinking. It cannot be changed without changing
         our thinking."',
          '"It is our choices, Harry, that show what we truly are, far more than our abilities."',
          '"There are only two ways to live your life. One is as though nothing is a miracle. The other is as t
         hough everything is a miracle."',
          '"The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stup
         id."',
          ""Imperfection is beauty, madness is genius and it's better to be absolutely ridiculous than absolute
         ly boring."",
          '"Try not to become a man of success. Rather become a man of value."',
          '"It is better to be hated for what you are than to be loved for what you are not."',
          ""I have not failed. I've just found 10,000 ways that won't work."",
          ""A woman is like a tea bag; you never know how strong it is until it's in hot water."",
          '"A day without sunshine is like, you know, night."']
```



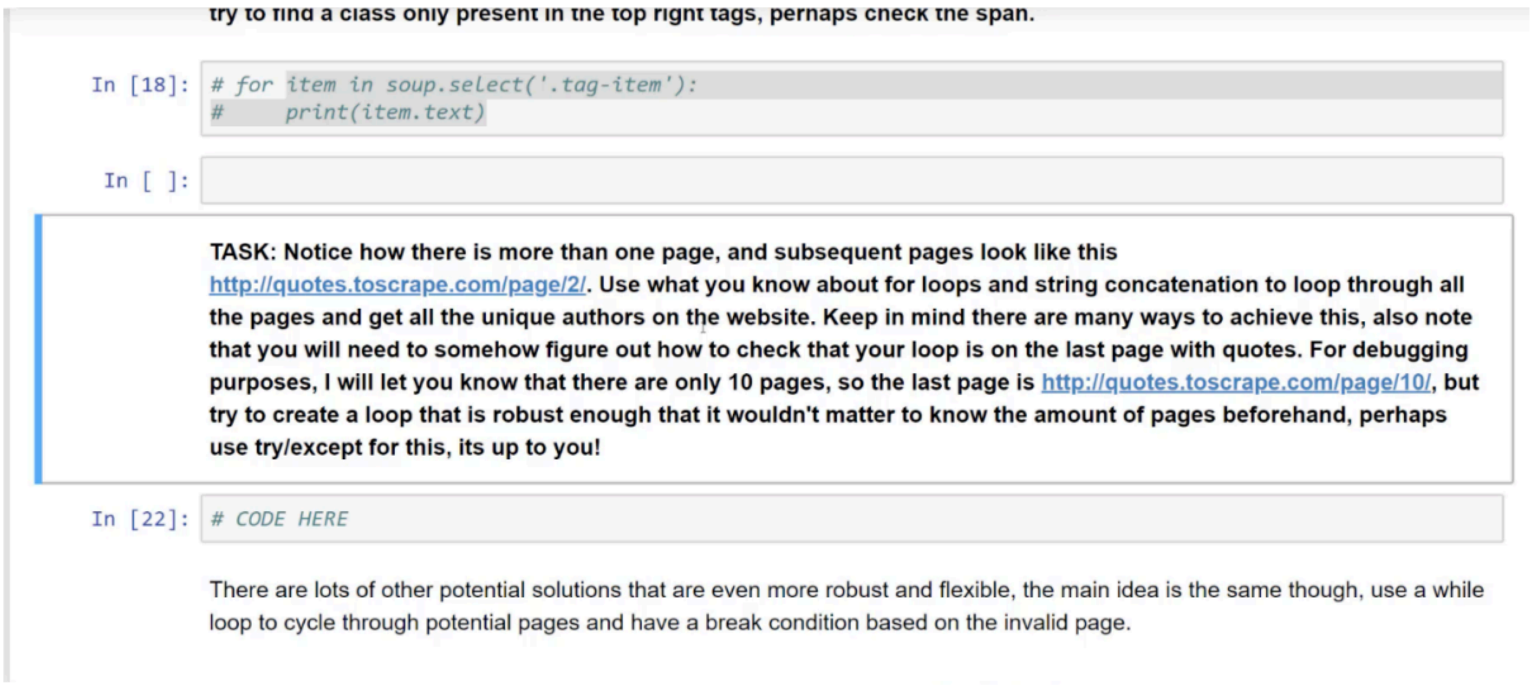
TASK: Inspect the site and use Beautiful Soup to extract the top ten tags from the requests text shown on the top right from the home page (e.g Love,Inspirational,Life, etc...). HINT: Keep in mind there are also tags underneath each quote, try to find a class only present in the top right tags, perhaps check the span.

```
In [17]: for item in soup.select('.tag-item'):
             print(item.text)

         love

         inspirational

         life

         humor

         books

         reading

         friendship

         friends
```

- **-> extracting the top 10 tags on the webpage**
  - **-> this is a process which can be generalised to do web scraping**
    - **-> the first stage is write down what you want <- he's inspecting the element on the webpage which we want**
    - **-> the element which we want to target was made with a span**
    - **-> then using the soup.select method to target that class**
  - -> this was tag-item (not just tag)
  - -> we want the unique authors on the page
  - -> he's stored the webpage URL in a variable

  - **-> we iterate through all of the webpages on the site in Python with a for loop**
    - -> he's generalised the URL for each of the webpages to do this
    - -> then in one of the variables, he has used the requests.get method on the URL to extract its information
    - -> then he converts this to a beautiful soup object in another variable
    - -> we are iterating through 10 pages on the website
    - -> he first does it for one page, and then the 10 pages
    - -> once this is running, he prints the results
    - -> this returns a list of all the authors on all 10 pages of the book
    - -> we won't know ahead of time how many pages there are
    - -> when we reach a page that doesn't have any quotes on there

    - **-> to test this, you can request a page whose number is > 10**
      - -> by testing, check to see if there are any quotes on the page
      - -> he returns the soup -> and in this case there are no quotes found on this page
      - -> he's returned the html for one of the webpages using res.text
      - **-> he is running the loop and checking for a boolean condition to see if the condition is satisfied**
      - -> then changing it to False, depending on the results which we have
      - -> this is done using a while loop
      - -> iterating through the different page numbers on the site again <- using a counter for this
      - -> to see if it's worked, he makes requests to a page with an extremely high page number (a page URL)
      - -> when iterating through page numbers in a while loop, we need break
      - -> with each iteration, we are checking if it's a valid stage or not
      - -> to code web scraping applications
      - -> running the code will take time to scrape the information
      - -> res.text is an entire string
      - **-> he is defining another while loop to iterate through the different webpages, which means having a counter what increases by 1 each time**
      - -> we need to first check if it's a valid page <- this is done using an if block
      - -> then iterating through the information on the page
      - -> extracting the information we want
      - -> he then runs authors
      - -> coding web scraping applications <- the second method used here is more robust because it's independent of the number of pages on the webpage
      - **-> you want to generalise it**



try to find a class only present in the top right tags, perhaps check the span.

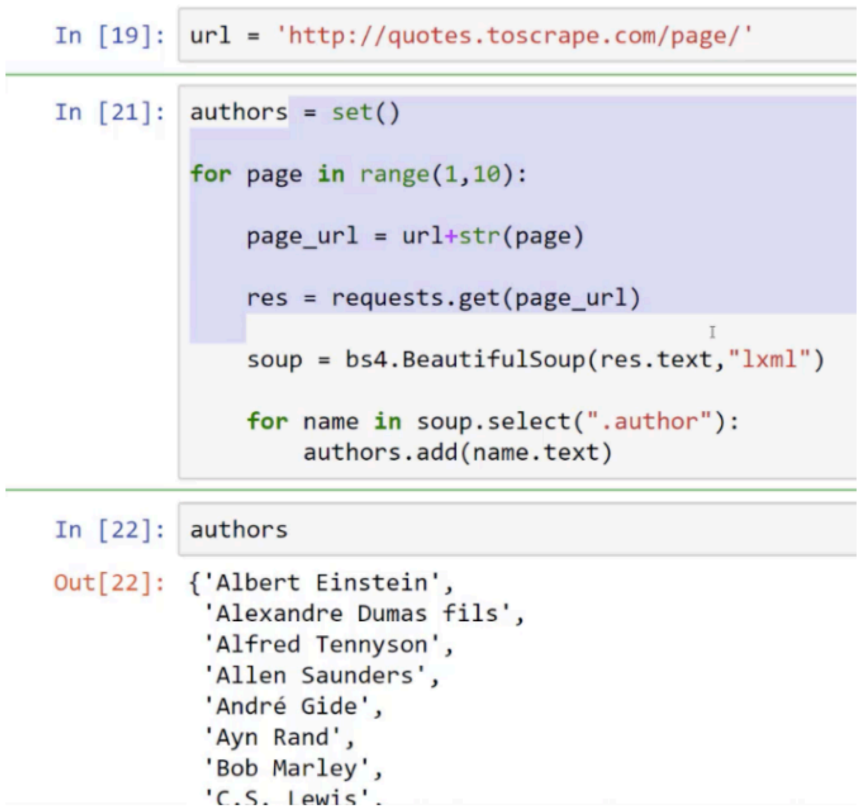
```
In [18]: # for item in soup.select('.tag-item'):
         #     print(item.text)
```

```
In [ ]:
```

TASK: Notice how there is more than one page, and subsequent pages look like this http://quotes.toscrape.com/page/2/. Use what you know about for loops and string concatenation to loop through all the pages and get all the unique authors on the website. Keep in mind there are many ways to achieve this, also note that you will need to somehow figure out how to check that your loop is on the last page with quotes. For debugging purposes, I will let you know that there are only 10 pages, so the last page is http://quotes.toscrape.com/page/10/, but try to create a loop that is robust enough that it wouldn't matter to know the amount of pages beforehand, perhaps use try/except for this, its up to you!

```
In [22]: # CODE HERE
```

There are lots of other potential solutions that are even more robust and flexible, the main idea is the same though, use a while loop to cycle through potential pages and have a break condition based on the invalid page.

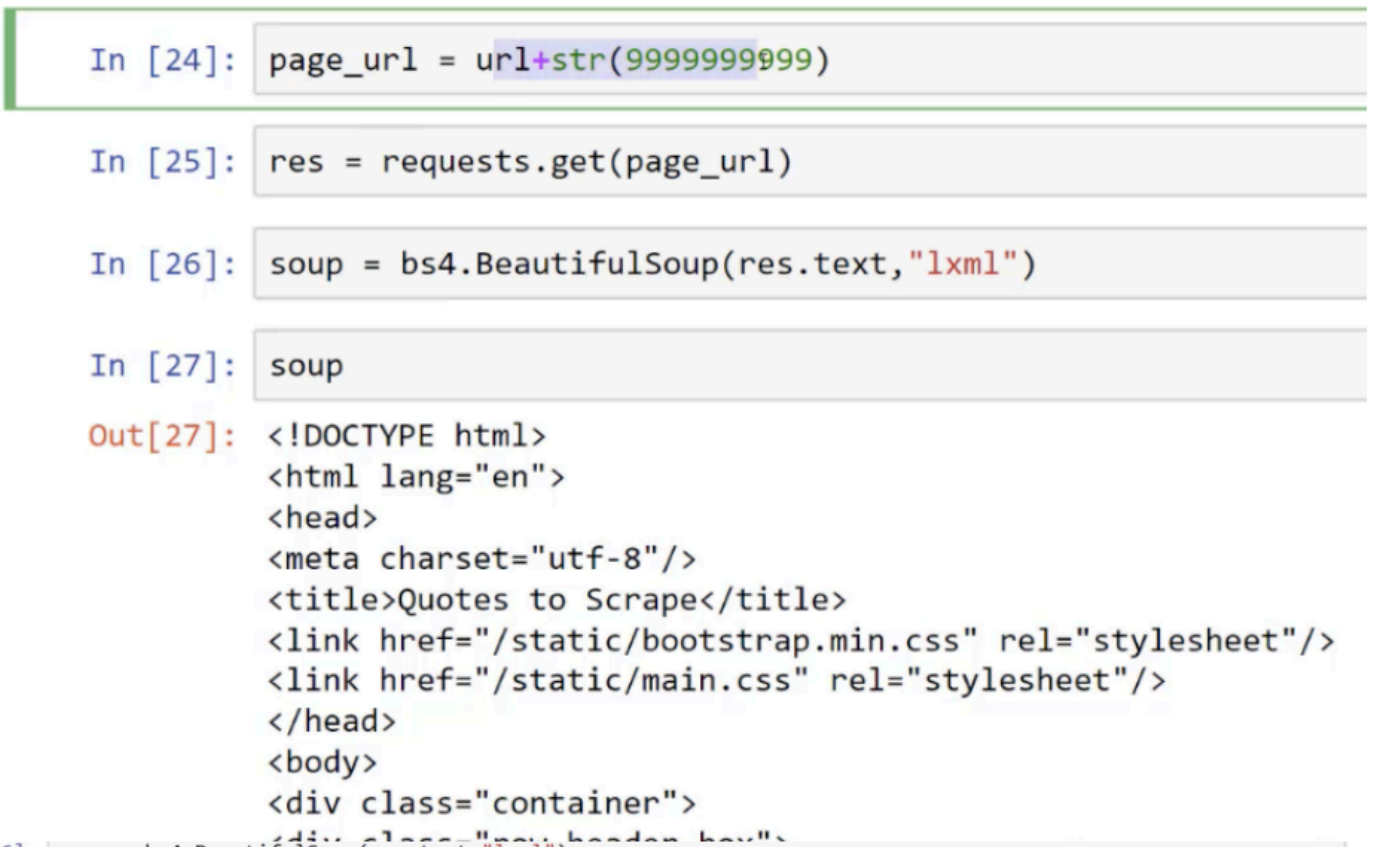


```
In [19]: url = 'http://quotes.toscrape.com/page/'
```

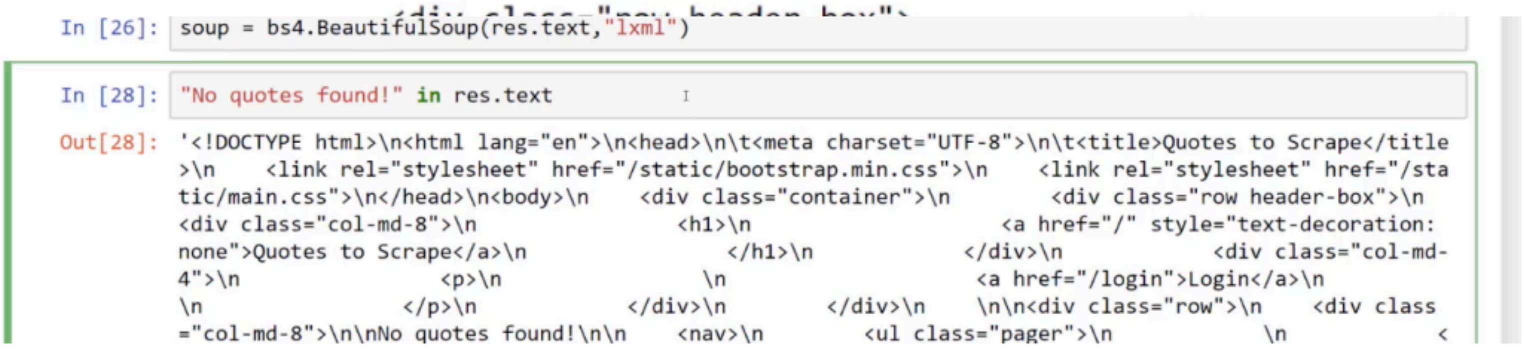
```
In [21]: authors = set()

         for page in range(1,10):

             page_url = url+str(page)

             res = requests.get(page_url)

             soup = bs4.BeautifulSoup(res.text,"lxml")

             for name in soup.select(".author"):
                 authors.add(name.text)
```

```
In [22]: authors
```

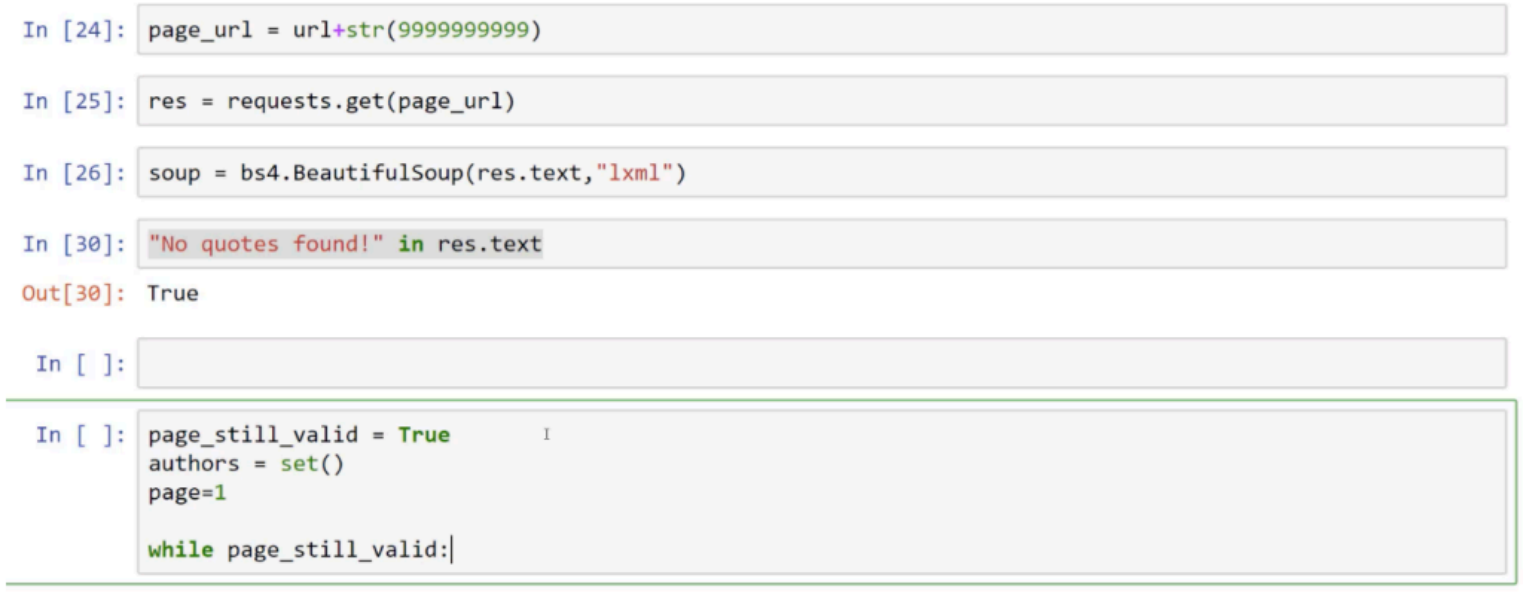
```
Out[22]: {'Albert Einstein',
          'Alexandre Dumas fils',
          'Alfred Tennyson',
          'Allen Saunders',
          'André Gide',
          'Ayn Rand',
          'Bob Marley',
          'C.S. Lewis',
```



```
In [24]: page_url = url+str(9999999999)

In [25]: res = requests.get(page_url)

In [26]: soup = bs4.BeautifulSoup(res.text,"lxml")

In [27]: soup

Out[27]: <!DOCTYPE html>
         <html lang="en">
         <head>
         <meta charset="utf-8"/>
         <title>Quotes to Scrape</title>
         <link href="/static/bootstrap.min.css" rel="stylesheet"/>
         <link href="/static/main.css" rel="stylesheet"/>
         </head>
         <body>
         <div class="container">
         <div class="row header-box">
```

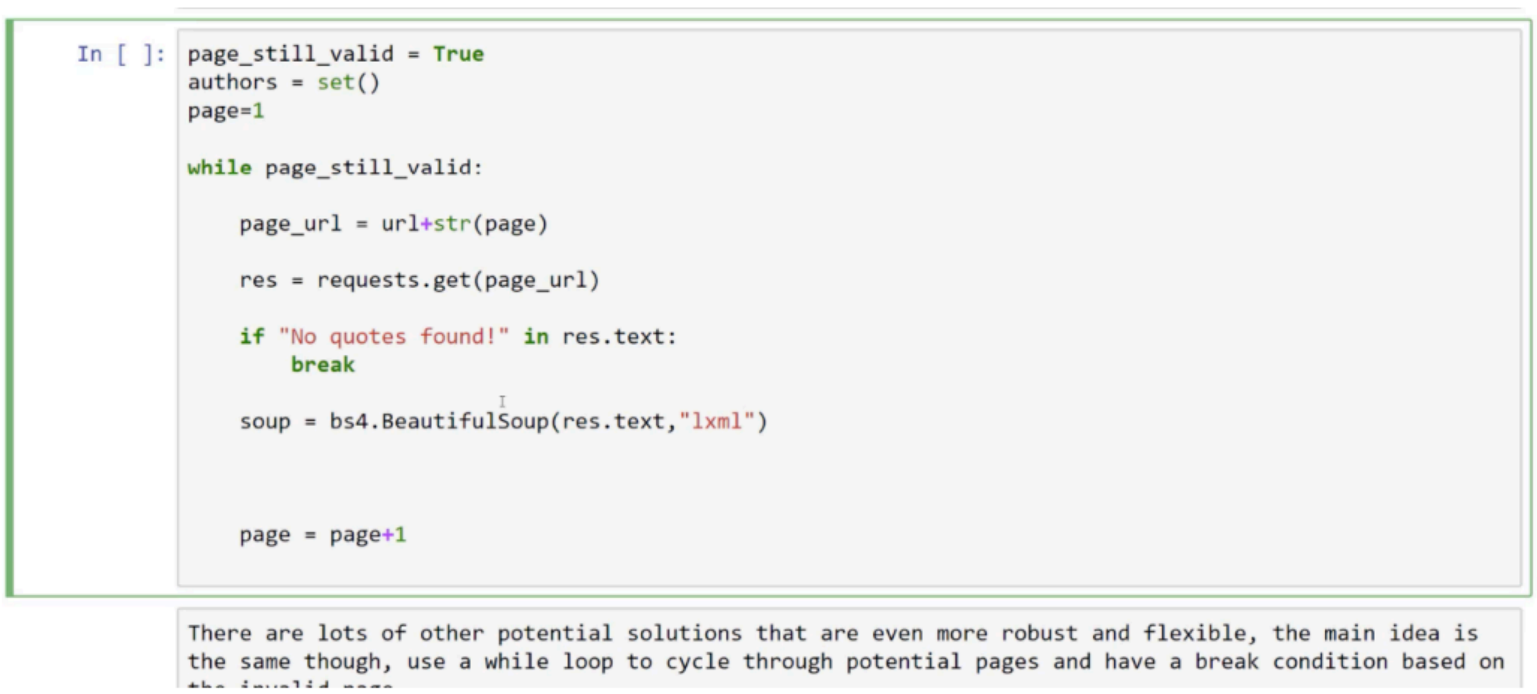


```
In [26]: soup = bs4.BeautifulSoup(res.text,"lxml")

In [28]: "No quotes found!" in res.text

Out[28]: '<!DOCTYPE html>\n<html lang="en">\n<head>\n\t<meta charset="UTF-8">\n\t<title>Quotes to Scrape</title
         >\n    <link rel="stylesheet" href="/static/bootstrap.min.css">\n    <link rel="stylesheet" href="/sta
         tic/main.css">\n</head>\n<body>\n    <div class="container">\n        <div class="row header-box">\n
         <div class="col-md-8">\n             <h1>\n                 <a href="/" style="text-decoration:
         none">Quotes to Scrape</a>\n             </h1>\n            </div>\n          <div class="col-md-
         4">\n             <p>\n              \n              <a href="/login">Login</a>\n
         \n             </p>\n          </div>\n        </div>\n    \n\n<div class="row">\n    <div class
         ="col-md-8">\n\nNo quotes found!\n\n     <nav>\n        <ul class="pager">\n              \n          <
```



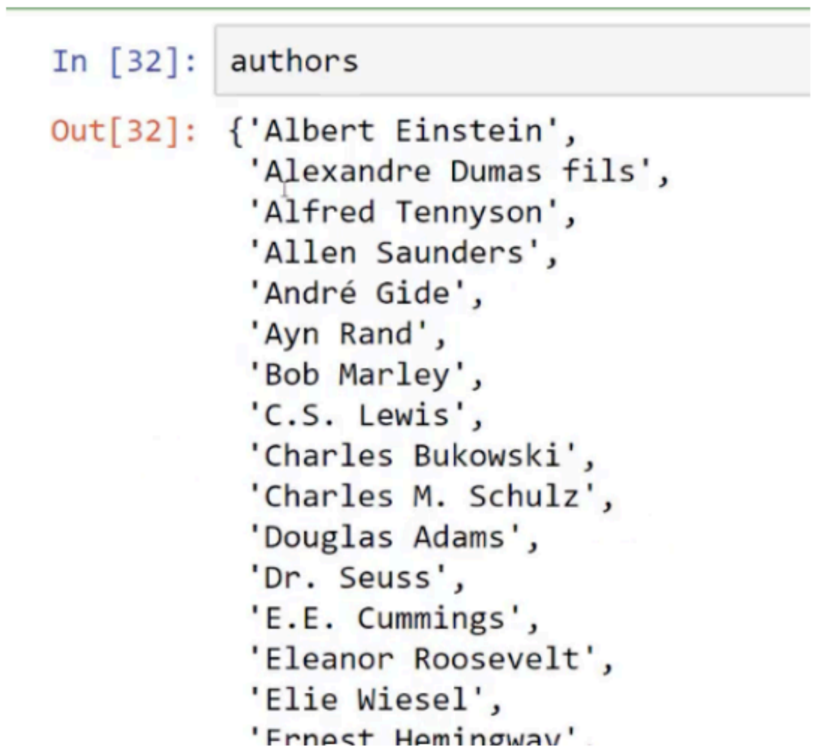
```
In [24]: page_url = url+str(9999999999)

In [25]: res = requests.get(page_url)

In [26]: soup = bs4.BeautifulSoup(res.text,"lxml")

In [30]: "No quotes found!" in res.text

Out[30]: True

In [ ]:

In [ ]: page_still_valid = True
        authors = set()
        page=1

        while page_still_valid:
```

There are lots of other potential solutions that are even more robust and flexible, the main idea is the same though, use a while loop to cycle through potential pages and have a break condition based on the invalid page.



```
In [ ]: page_still_valid = True
        authors = set()
        page=1

        while page_still_valid:

            page_url = url+str(page)

            res = requests.get(page_url)

            if "No quotes found!" in res.text:
                break

            soup = bs4.BeautifulSoup(res.text,"lxml")



            page = page+1
```

There are lots of other potential solutions that are even more robust and flexible, the main idea is the same though, use a while loop to cycle through potential pages and have a break condition based on



```
In [32]: authors

Out[32]: {'Albert Einstein',
          'Alexandre Dumas fils',
          'Alfred Tennyson',
          'Allen Saunders',
          'André Gide',
          'Ayn Rand',
          'Bob Marley',
          'C.S. Lewis',
          'Charles Bukowski',
          'Charles M. Schulz',
          'Douglas Adams',
          'Dr. Seuss',
          'E.E. Cummings',
          'Eleanor Roosevelt',
          'Elie Wiesel',
          'Ernest Hemingway',
```