



BREAKOUT ACTIVITIES

BREAKOUT 3.1: School Survey Web Page

Among the many applications of Python strings are file processing and HTML files generation. Consider the following scenario.

Your school loves to hear feedback from teachers and students on a wide variety of topics and issues – so much so, that the school website even has a dedicated page for hosting weekly surveys. This week's survey, shown below, relates to the menu in the school canteen.

School Survey

Do you think the menu in the school canteen should be changed?

First name:

Last name:

Thank you for taking part in the school survey".

The format of the web page is the same for every survey. Surveys differ from each other only in the actual question being asked, and the text of possible answers which always appear in two buttons displayed side-by-side on the page. Participants are always asked to enter their first name and last name.

The questions and answer options for the next three surveys approved by the school board are shown below.

- | | | |
|---|-------|-----------|
| 1. Destination for School Tour next year? | Rome | Barcelona |
| 2. Exam year students should be allowed to take part in <i>all</i> extra-curricular activities? | Agree | Disagree |

3. Computational Thinking should be taught as part of every subject Always Sometimes

The problem is that every week the school 'techie', Ms E. Fish, has to edit the HTML file with next week's survey question and answers, and upload it to the web server to make it live on the school web site.

The 'techie', who is also the school's Computer Science teacher understands HTML, but with the new term fast approaching is very busy developing content for the learning outcomes in the new LCCS specification as well as coming up with ideas for teaching the course through the lens of the four Applied Learning Tasks.

Even though the HTML code behind the survey page – shown below - is fairly straightforward, she needs come up with an efficient solution so that her colleague. Mr. Chips (the maths teacher who does not possess any knowledge of HTML) can upload the new survey.

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h2>School Survey</h2>
6 <p>Do you think the menu in the school canteen should be changed?</p>
7 <form action="">
8   First name:<br>
9   <input type="text" name="firstname" value="">
10  <br>
11  Last name:<br>
12  <input type="text" name="lastname" value="">
13  <br><br>
14  <input type="submit" value="Yes">
15  <input type="submit" value="No">
16 </form>
17
18 <p>Thank you for taking part in the school survey"</p>
19
20 </body>
21 </html>
```

survey.html

The solution Ms. E. Fish comes up with is as follows:

1. She will create a HTML survey template file with placeholders for the question and answer text to be used in the survey. The name of this file will be `survey_template.html`.
2. She will create a normal text file to store the question and answer text for the next survey. The name of this file will be `survey.txt`.

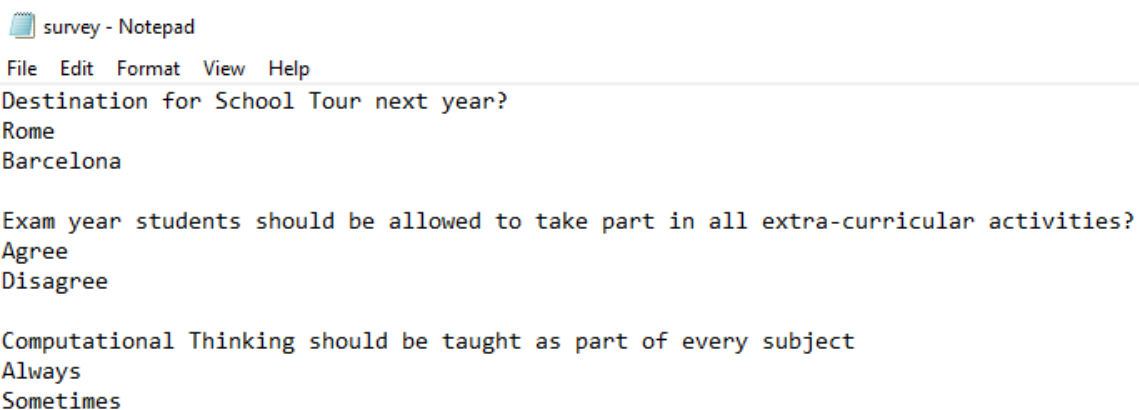
3. She will write a Python program that generates a HTML for the next survey using two files `survey_template.html` and `survey.txt`. The name of the generated file will be `survey.html`.

All Mr. Chips will have to do every week is edit the text file and run the program. No need to worry about HTML - easy!

Ms. E. Fish was delighted that her proof of concept worked and so commenced the job of implementing her solution. She created the two files - `survey_template.html` and `survey.txt` – as per design. These are shown below. Notice the three placeholders for the text on lines 6, 14 and 15 in the `survey_template.html`

```
1 <!DOCTYPE html>
2 <html>
3 <body>
4
5 <h2>School Survey</h2>
6 <p><place-holder-1></p>
7 <form action="">
8   First name:<br>
9   <input type="text" name="firstname" value="">
10  <br>
11  Last name:<br>
12  <input type="text" name="lastname" value="">
13  <br><br>
14  <input type="submit" value="<place-holder-2>">
15  <input type="submit" value="<place-holder-3>">
16 </form>
17
18 <p>Thank you for taking part in the school survey"</p>
19
20 </body>
21 </html>
22
```

survey_template.html



survey - Notepad

File Edit Format View Help

Destination for School Tour next year?

Rome

Barcelona

Exam year students should be allowed to take part in all extra-curricular activities?

Agree

Disagree

Computational Thinking should be taught as part of every subject

Always

Sometimes

survey.txt

However, soon after she started writing the Python program to generate the weekly survey, Ms. E. Fish realised that she needed to prioritise lesson planning for the fast approaching LCCS.

With the program, shown below, at unit test stage, unless someone steps in, the project is looking like it might have to be shelved and the destination for next year's school tour is still undecided.

```
# Open the survey template
htmlFile = open("survey_template.html","r", encoding="utf-8")
htmlStr = htmlFile.read()
htmlFile.close() # close the file

# Open the data file
surveyFile = open("survey.txt","r")
surveyQuestion = surveyFile.readline()

*** YOUR CODE GOES HERE ***

surveyFile.close() # close the file

# Now replace the 'markers' from the template file with the runtime data
htmlStr2 = htmlStr.replace("<place-holder-1>", surveyQuestion)

*** YOUR CODE GOES HERE ***

# Write the survey page
htmlFile = open("survey.html","w", encoding="utf-8") # Open the file for writing
htmlFile.write(htmlStr2)
htmlFile.close() # close the file
```

The desired output is shown below.

School Survey

Destination for School Tour next year?

First name:

Last name:

Rome

Barcelona

Thank you for taking part in the school survey.

Suggested Activities

1. Restate the problem in your own words.



2. Complete the program by adding the necessary code inside the red rectangles to get it displaying the desired output. You will need to make sure the two files - `survey_template.html` and `survey.txt` – are in the runtime directory.



3. Outline how you might test your system.



4. Describe in detail the steps you would have to take in order to introduce a third answer button into the survey



BREAKOUT 3.2: RSS Feed Analysis

Really Simple Syndication (RSS, aka Rich Site Summary) is a technology that allows an end-user to have information delivered automatically from selected website(s) to a device. The information is referred to as a *feed*. User can subscribe to receive RSS feeds from specific sites, and in this way, keep up-to-date with the information they are interested in.

Typically, these sites contain information relating to business, jobs, blogs, news, entertainment etc. Once a user has subscribed, feeds containing any new information on these sites are automatically 'sent' to that user who can then view them using a RSS reader (typically a web browser).



Use your favourite browser to locate and record the names and URLs of some RSS sites that interest you.

1. _____
2. _____
3. _____
4. _____
5. _____

In this session we will modify and write code that analyses data from a live RSS feed of your choice.

A working program that connects to a URL and delivers a live RSS from that URL to a variable called `feed` is provided as a starting point. The program listing is shown on the next page.

Notice that a large section of the code is enclosed in a black box. The black box is used to indicate code that is needed for the program to run, but not necessary to understand. In this example, the code inside the black box tells Python to read a RSS feed from the URL specified as a string on line number 24.

Program to read a RSS feed

The listing below pulls live headline news publically available from Apple's RSS feed. The data is stored in the string variable, `feed`.

Line 24 is an assignment statement. The right hand side tells Python to used code (inside the black box) to read the contents of a RSS feed. The full URL of the site from which to read the feed is *hard-coded* as a string in this line. The feed itself is returned as a string and stored in the variable `feed`.

```
1. from urllib.request import urlopen
2. from xml.dom import minidom
3. import collections
4.
5. # extract the headlines from the feed
6. def extractString(doc):
7.     str = ""
8.     for node in doc.getElementsByTagName('channel'):
9.         for title in node.getElementsByTagName('title'):
10.            str = str + title.firstChild.data + "\n"
11.     return str
12.
13. # extract the feed from the url
14. def getRSSString(url):
15.     results = []
16.     rssString = ""
17.     results.append(minidom.parse(urlopen(url)))
18.     for webDoc in results:
19.         rssString = rssString + extractString(webDoc)
20.     return rssString
21.
22. # PROGRAM START FROM HERE ...
23. # Read the RSS feed from the URL provided
24. feed = getRSSString("https://www.apple.com/main/rss/hotnews/hotnews.rss")
25. # Display the results
26. print(feed)
27. # Display the number of lines
28. print("There were %d lines in this feed" %feed.count("\n"))
```

Line 26 displays the contents of the `feed` on the output console.

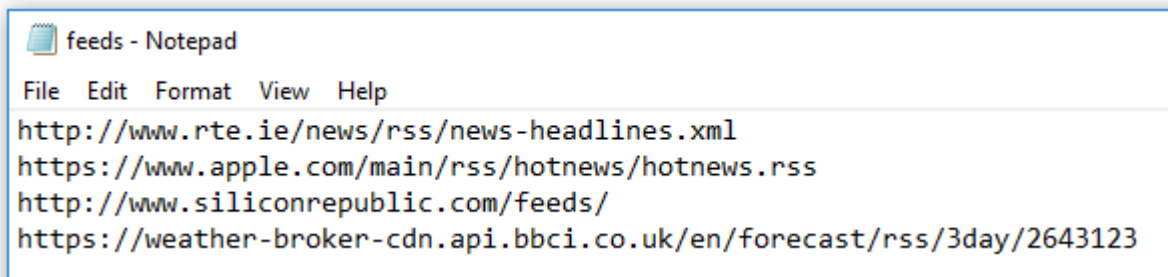
Line 28 displays the number of lines in `feed`. It does this by using the `count` command (method) to count the number of newline characters in `feed`. (A standard technique used to count the number of lines in a piece of text is to count the number of occurrences of the newline character, `'\n'`.)

We are now ready to make changes to the code.

Suggested Activities

1. Key in the full program and make sure it runs properly.
2. The feed URL is hard-coded into the program (line 24). This means that a programmer must change the code every time a different feed needs to be read.
Modify the program so that it reads the URL from the first line of a text/data file.

Hint #1: You will need to create a file first (e.g. `feeds.txt`), and save it in your runtime directory. The file might look something like this:



```
feeds - Notepad
File Edit Format View Help
http://www.rte.ie/news/rss/news-headlines.xml
https://www.apple.com/main/rss/hotnews/hotnews.rss
http://www.siliconrepublic.com/feeds/
https://weather-broker-cdn.api.bbci.co.uk/en/forecast/rss/3day/2643123
```

Hint #2: A solution can be obtained by using four of following lines (in a different order).

```
        feedFile.close()
feedFile.write("https://www.apple.com/main/rss/hotnews/hotnews.rss")
        feedURL = feedFile.readline()
        feed = getRSSString(feedFile)

        feedFile = open("feeds.txt", "w")
        feedFile = open("feeds.txt", "r")
        feed = getRSSString(feedURL)
```

3. Extend the code so that it displays the entire feed a) capitalised, b) in upper case and c) in lower case.
4. Write a line of code that replaces every occurrence of a specific word in the feed with another word of your choosing e.g. replace the word 'Apple' with the word 'Microsoft'. Display the new string.
5. Write a code snippet to count and add the number of vowels in the feed. (Later we will use the `plotly` library to display these data on a bar chart.)

6. Add the following code snippet to the end of your program and run it.

```
words = feed.split()
print(collections.Counter(words).most_common(5))
```



What happened when you added the above code?

What do these two lines do?

Can you condense these two lines into one line that does the same thing?

-
-
-
7. Write code to replace the first **two** occurrence of a specific word/phrase in the feed with another word/phrase of your choosing. (You will need to identify some word/phrase that occurs at least twice yourself.) The output should display the entire feed with the new word/phrase in place of the original two.

Hint: You will need to use the techniques of indexing/slicing and concatenation to construct the output string.

8. Same as previous activity except, instead of replacing the two words, your program should apply the Caesar cipher algorithm implemented earlier in this Section.
9. Browse to the official Python site documentation on string methods (see link below) and identify some method(s) that you have not already used. Now, use the feed from this activity to test drive your new string method.

<https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str>

10. Read the listing below and predict what it will do.

Now key it in and run it. Was your prediction correct?

```
1. # A program to read and display a web page
2. import urllib.request
3.
4. # Open the URL
5. page = urllib.request.urlopen("https://www.compsci.ie/")
6. # Read and decode
7. text = page.read().decode("utf8")
8. # Display
9. print(text)
```



Experiment!

What happens if you change the URL on line 5?



Discuss in groups ways an activity (or sequence of activities) might be developed around the above code. (You may find it useful to reflect on the types of activities carried out in this breakout session on RSS feeds.)
