An example webpage

- For instance, here's the source code of a very basic webpage.

In [14]:
```
html_string = '''
<html>
    <body>
      <div id="content">
        <h1>Heading here</h1>
        <p>My First paragraph</p>
        <p>My <em>second</em> paragraph</p>
        <hr>
      </div>
      <div id="nav">
        <ul>
          <li>item 1</li>
          <li>item 2</li>
          <li>item 3</li>
        </ul>
      </div>
    </body>
</html>'''
```

- Here's what that webpage actually looks like:

In [15]:
```
HTML(html_string)
```

Out[15]:

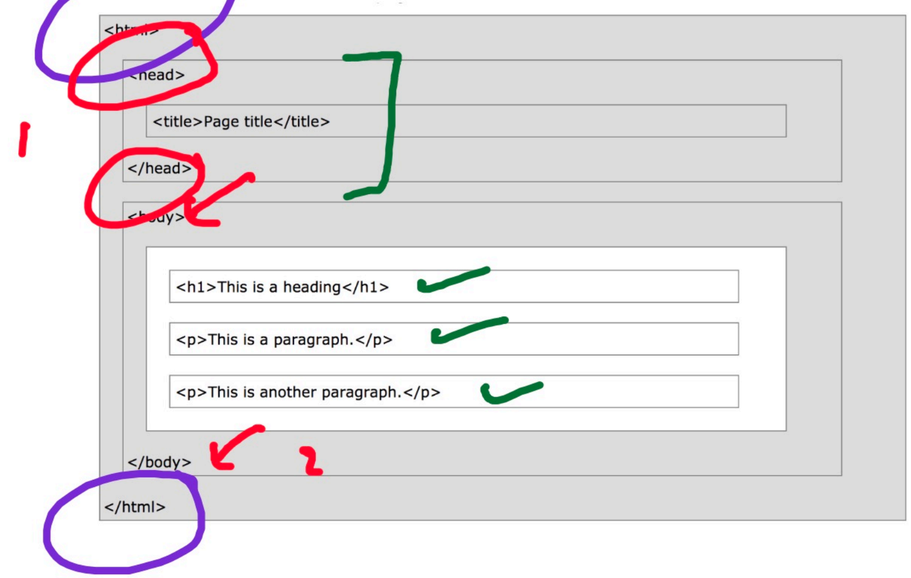
Heading here

My First paragraph

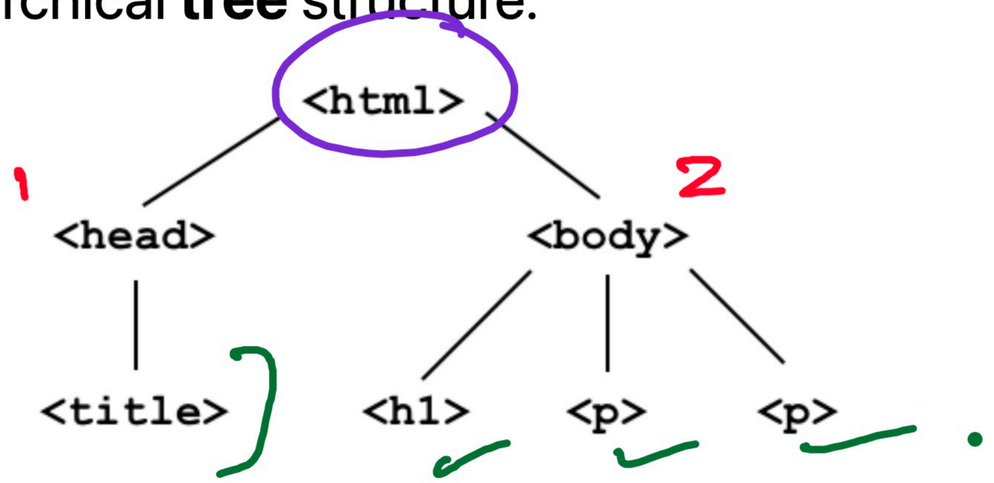
My *second* paragraph

- item 1
- item 2
- item 3

- **HTML document**: The totality of markup that makes up a webpage.



- **Document Object Model (DOM)**: The internal representation of an HTML document as a hierarchical **tree** structure.



Beautiful Soup 🍜

- [Beautiful Soup 4](#) is a Python HTML parser.

  Remember, **parse** means to "extract meaning from a sequence of symbols".

- **Warning**: Beautiful Soup 4 and Beautiful Soup 3 work differently, so make sure you are using and looking at documentation for Beautiful Soup 4.

  Rest assured, the `pds` conda environment already has Beautiful Soup 4 installed.

```html
<head>
    <title>3*Canada-2022-06-04</title>
</head>
<body>
    <h1>Spotify Top 3 - Canada</h1>
    <table>
        <tr class='heading'>
            <th>Rank</th>
            <th>Artist(s)</th>
            <th>Song</th>
        </tr>
        <tr class=1>
            <td>1</td>
            <td>Harry Styles</td>
            <td>As It Was</td>
        </tr>
        <tr class=2>
            <td>2</td>
            <td>Jack Harlow</td>
            <td>First Class</td>
        </tr>
        <tr class=3>
            <td>3</td>
            <td>Kendrick Lamar</td>
            <td>N95</td>
        </tr>
    </table>
</body>
```

14 leafs

- **Part 1**: How many leaf nodes are there in the DOM tree of the previous document — that is, how many nodes have no children?

```
    <title>3*Canada-2022-06-04</title>
</head>
<body>
    <h1>Spotify Top 3 - Canada</h1>
    <table>
        <tr class='heading'>
            <th>Rank</th>
            <th>Artist(s)</th>
            <th>Song</th>
        </tr>
        <tr class=1>
            <td>1</td>
            <td>Harry Styles</td>
            <td>As It Was</td>
        </tr>
        <tr class=2>
            <td>2</td>
            <td>Jack Harlow</td>
            <td>First Class</td>
        </tr>
        <tr class=3>
            <td>3</td>
            <td>Kendrick Lamar</td>
            <td>N95</td>
        </tr>
    </table>
</body>
```

- **Part 1**: How many leaf nodes are there in the DOM tree of the previous document — that is, how many nodes have no children?

9

- **Part 2**: What does the following line of code evaluate to?

```
len(soup.find_all("td"))
```

```
<div class="alert alert-success" markdown="1">
    <h3>Activity</h3>

Consider the following HTML document, which represents a webpage containing the top few songs with the most streams on Spotify today in Canada.

```html
<head>
    <title>3*Canada-2022-06-04</title>
</head>
<body>
    <h1>Spotify Top 3 - Canada</h1>
    <table>
        <tr class='heading'>
            <th>Rank</th>
            <th>Artist(s)</th>
            <th>Song</th>
        </tr>
        <tr class=1>
            <td>1</td>
            <td>Harry Styles</td>
            <td>As It Was</td>
        </tr>
        <tr class=2>
            <td>2</td>
            <td>Jack Harlow</td>
            <td>First Class</td>
        </tr>
        <tr class=3>
            <td>3</td>
            <td>Kendrick Lamar</td>
            <td>N95</td>
        </tr>
    </table>
</body>
```
```

- **Part 1**: How many leaf nodes are there in the DOM tree of the previous document — that is, how many nodes have no children?

- **Part 2**: What does the following line of code evaluate to?

  ```
  len(soup.find_all("td"))
  ```

- **Part 3**: What does the following line of code evaluate to?

  ```
  soup.find("tr").get("class")
  ```