## Python HTML table parser without dependencies

I wanted to parse a HTML page where I was interested in one particular table. It's quite a easy task if you can install dependencies like pandas html5lib lxml beautfulsoup or others. But according to my standards if I can dodge another dependency I do so and for one table to be dug out of HTML document … like if you cannot do that with included batteries you are a C developer or a lazy one.

### Finding the table

First step is to find the table inside HTML document so you can later process only the table HTML and don't bother with the whole document. This can be done quite easily with regular expressions if you know what are you looking for - if you can identify your table among others in the document or if there is only one.

In my case the document only had one table so I didn't have to bother with IDs or CSS classes to identify my target. Basically the regular expression should be like "cut out everything between `<table>` and `</table>` including those tags". This can be done with the following regular expression:

```
<table.*?>.*?</table>
```

and `s` as a flag/option/modifier.

### Digging out table data

Once you have the table HTML markup you can utilize built-in HTMLParser class. All it takes it to write hook methods for what you need and your parser is born.

My case requires one method and that's the one that does process data which means tag contents and thats `get_data()`. Once I can collect all data from the table into a list and then I can split the list into "rows" where the whole data ends up as a list of lists I won. Following code does exactly what I just described.

```
from html.parser import HTMLParser
from itertools import zip_longest
from pprint import pprint


class HtmlTableParser(HTMLParser):
    def __init__(self, columns, *args, **kwargs):

        self.data = []
```

```
        self.columns = columns
        super().__init__(*args, **kwargs)

    def handle_data(self, data):

        if data := data.strip():
            self.data.append(data)

    def get_data(self):

        return list(zip_longest(*[iter(self.data)] * self.columns,
         ↪  fillvalue=""))
```

Usage can be:

```
parser = HtmlTableParser(columns=4)
parser.feed(open("./table.html", "r").read())
print(parser.get_data())
```

and the output can look like this:

```
[('Announcement Date', 'Fiscal Quarter End', 'Estimated EPS', 'Actual EPS'),
 ('2022-05-03', '2022-03-31', '$0.83', '$1.02'),
 ('2022-02-01', '2021-12-31', '$0.69', '$0.83'),
 ('2021-10-26', '2021-09-30', '$0.61', '$0.66'),
 ('2021-07-27', '2021-06-30', '$0.48', '$0.58'),
 ('2021-04-27', '2021-03-31', '$0.38', '$0.47'),
 ('2021-01-26', '2020-12-31', '$0.41', '$0.45'),
 ('2020-10-27', '2020-09-30', '$0.31', '$0.35'),
 ('2020-07-28', '2020-06-30', '$0.12', '$0.13'),
 ('2020-04-28', '2020-03-31', '$0.14', '$0.14'),
 ('2020-01-28', '2019-12-31', '$0.26', '$0.27'),
 ('2019-10-29', '2019-09-30', '$0.14', '$0.14'),
 ('2019-07-30', '2019-06-30', '$0.05', '$0.04'),
 ('2019-04-30', '2019-03-31', '$0.02', '$0.03'),
 ('2019-01-29', '2018-12-31', '$0.06', '$0.05'),
 ('2018-10-24', '2018-09-30', '$0.11', '$0.10'),
 ('2018-07-25', '2018-06-30', '$0.10', '$0.12'),
 ('2018-04-25', '2018-03-31', '$0.06', '$0.08'),
 ('2018-01-30', '2017-12-31', '$0.02', '$0.06'),
 ('2017-10-24', '2017-09-30', '$0.06', '$0.08'),
 ('2017-07-25', '2017-06-30', '$-0.02', '$-0.01'),
 ('2017-05-01', '2017-03-31', '$-0.07', '$-0.07'),
 ('2017-01-31', '2016-12-31', '$-0.04', '$-0.04'),
```

```
('2016-10-20', '2016-09-30', '$-0.02', '$0.00'),
('2016-07-21', '2016-06-30', '$-0.11', '$-0.07'),
('2016-04-21', '2016-03-31', '$-0.15', '$-0.14'),
('2016-01-19', '2015-12-31', '$-0.12', '$-0.12'),
('2015-10-15', '2015-09-30', '$-0.12', '$-0.19'),
('2015-07-16', '2015-06-30', '$-0.17', '$-0.19'),
('2015-04-16', '2015-03-31', '$-0.06', '$-0.11'),
('2015-01-20', '2014-12-31', '$0.01', '$0.00')]
```

There you go - no dependencies for simple task - just a few lines of clean Python code.