

# HTML Parser - Part 2

\* This section assumes that you understand the basics discussed in **HTML Parser - Part 1**

## [.handle\\_comment\(data\)](#)

This method is called when a comment is encountered (e.g. `<!--comment-->`).  
The *data* argument is the content inside the comment tag:

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_comment(self, data):
        print("Comment :", data)
```

## [.handle\\_data\(data\)](#)

This method is called to process arbitrary data (e.g. text nodes and the content of `<script>...</script>` and `<style>...</style>`).  
The *data* argument is the text content of HTML.

```
from html.parser import HTMLParser

class MyHTMLParser(HTMLParser):
    def handle_data(self, data):
        print("Data :", data)
```

## Task

You are given an *HTML* code snippet of  $N$  lines.  
Your task is to print the *single-line comments*, *multi-line comments* and the *data*.

Print the result in the following format:

```
>>> Single-line Comment
Comment
>>> Data
My Data
>>> Multi-line Comment
Comment_multiline[0]
Comment_multiline[1]
>>> Data
My Data
>>> Single-line Comment:
```

**Note:** Do not print *data* if `data == '\n'`.

## Input Format

The first line contains integer  $N$ , the number of lines in the *HTML* code snippet.  
The next  $N$  lines contain *HTML* code.

## Constraints

$$0 < N < 100$$

## Output Format

Print the *single-line comments*, *multi-line comments* and the *data* in order of their occurrence from top to bottom in the snippet.

Format the answers as explained in the problem statement.

## Sample Input

```
4
<!--[if IE 9]>IE9-specific content
<![endif]-->
<div> Welcome to HackerRank</div>
<!--[if IE 9]>IE9-specific content<![endif]-->
```

## Sample Output

```
>>> Multi-line Comment
[if IE 9]>IE9-specific content
<![endif]
>>> Data
    Welcome to HackerRank
>>> Single-line Comment
[if IE 9]>IE9-specific content<![endif]
```