## Files

- Why do we want to output into files?
  - because our data is lost when program ends
  - so we need to save results to disk
  - we want to generate *lots* of files
- Why do we want to input from files?
  - to read 'real world' data
  - to get access to all that 'big data'
- What is a disk file?
  - a sequence of characters (like a string)
  - **Warning**: 'newline' character is different on different systems. CR LF on Windows, CR on Mac, LF on Linux

## Writing to a file

```
myfile = open("test.txt", "w")
myfile.write("My first output file\n")
myfile.write("--------------------\n")
myfile.write("Hello, world!\n")
myfile.close()
```

test.txt

```
My first output file
--------------------
Hello, world!
```

# Reading from a file v1

test.txt

```
My first output file
--------------------
Hello, world!
```

```python
handle = open("test.txt", "r")
while True:
    line = handle.readline()
    if len(line) == 0:  # no more lines?
        break
    # Now process the line we just read
    print(line)
handle.close()
```

## Reading from a file v2

test.txt

```
My first output file
--------------------
Hello, world!
```

```
handle = open("test.txt", "r")
line = handle.readline() #read first line
while len(line) > 0:
    # Now process the line we just read
    print(line, end="")
    line = handle.readline() #read next line
handle.close()
```

# Reading from a file v3

test.txt

```
My first output file
--------------------
Hello, world!
```

```python
handle = open("test.txt", "r")
for line in handle:
    # Now process the line we just read
    print(line, end="")
handle.close()
```

## Summary

- General code pattern is always:

```
handle = open(filename, mode)
. . .
handle.close()
```

- Mode can be:
  - "r"  for reading a text file
  - "rb" for reading a binary file
  - "w" for writing (or create) a text file
  - "wb" for writing a binary file

## Demo

- Write a file
  - with a loop!
  - view it on disk
- Read the file back in again
  - sum the data
  - print the summary

## Some shortcuts

1. read all lines into a list?

```
f = open("friends.txt", "r")
xs = f.readlines()
f.close()
xs.sort()
...
```

2. read whole file into a string?

```
f = open("somefile.txt")
content = f.read()
f.close()
print("Contains", len(content.split()), "words")
```

## Difficult CSV files?

- Some CSV files are complex to read:

```
Name,Age,Income
Jane Smith,44,67143
"John Smith, Jr",  45,"23,456"
```

- If we read this, and row.split(",") each line:

```
['Name',          'Age',     'Income'   ]
['Jane Smith',    '44',      '67143'    ]
[' "John Smith',  'Jr" ',    '45',       ' "23',     '456" ']
```

- Oops!
  – CSV files from Excel are often like this.

# CSV reader

- Python has a fancy CSV reader for these:

```
import csv
file = open("incomes.csv", "r")
reader = csv.reader(file)
for row in reader:
    print(row)
```

- This gives better results:

```
['Name',            'Age',      'Income'   ]
['Jane Smith',      '44',       '67143'    ]
['John Smith, Jr', '  45',      '23,456'   ]
```

# CSV reader

- Python has a fancy CSV reader for these:

```
import csv
file = open("incomes.tsv", "r")
reader = csv.reader(file, 'excel-tab')
for row in reader:
    print(row)
```

- This gives better results:

```
['Name',              'Age',      'Income'   ]
['Jane Smith',        '44',       '67143'    ]
['John Smith, Jr', '  45',        '23,456'   ]
```

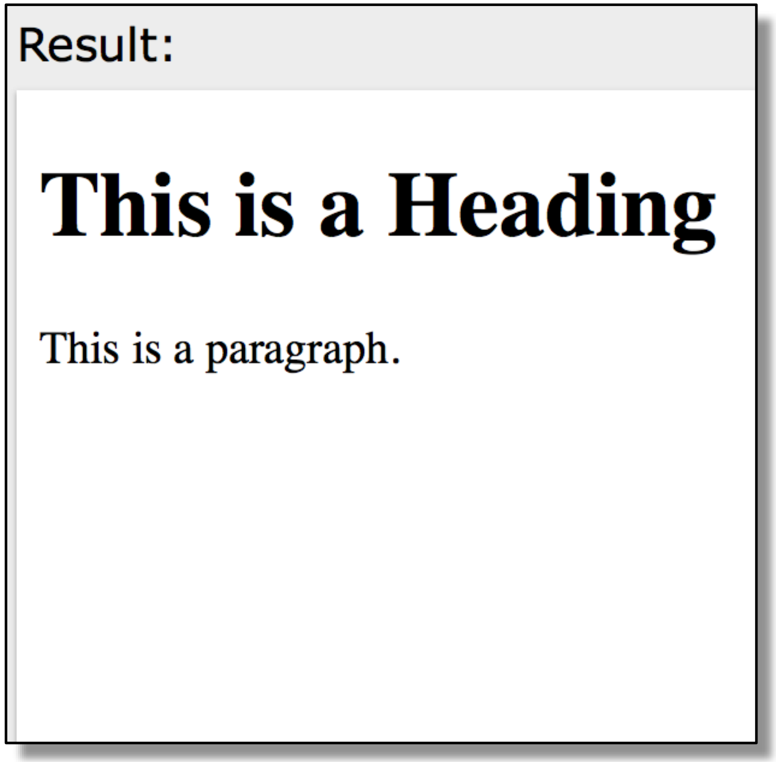
- Dialects: ['unix', 'excel', 'excel-tab']

# Crash Course: HTML

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Tutorial</title>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

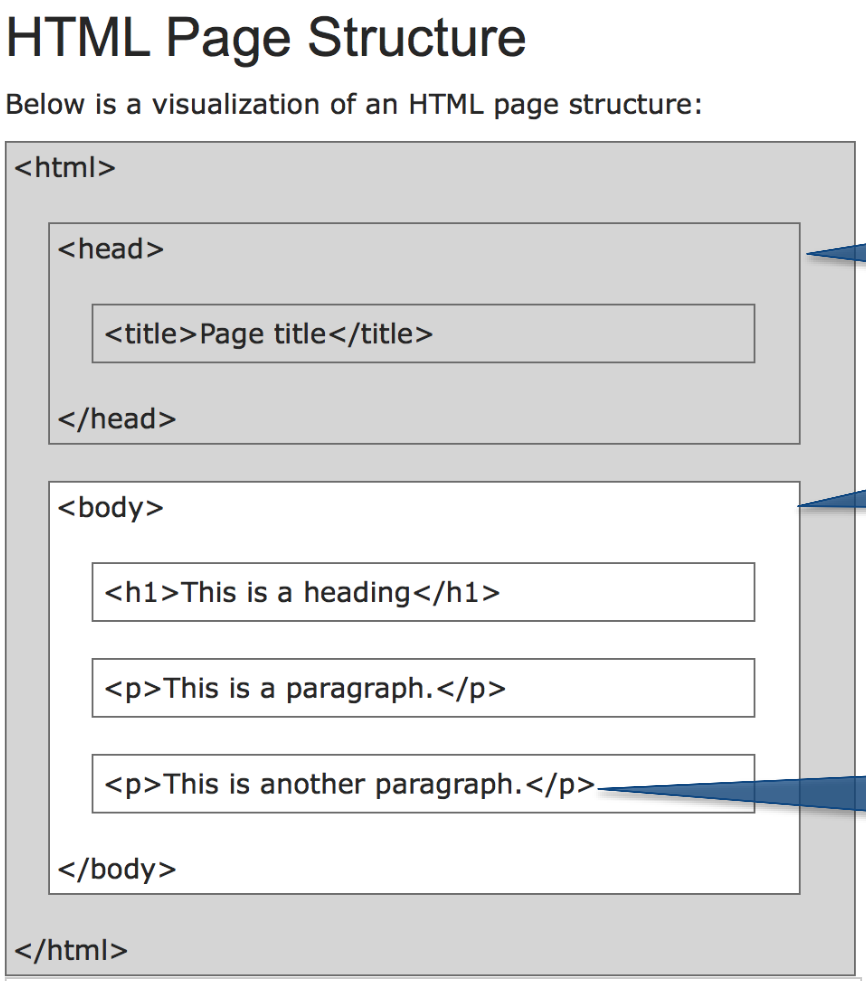

Result:

This is a Heading

This is a paragraph.

See [http://w3schools.com](http://w3schools.com)

# HTML page structure



## HTML Page Structure

Below is a visualization of an HTML page structure:

```
<html>
    <head>
        <title>Page title</title>
    </head>
    <body>
        <h1>This is a heading</h1>
        <p>This is a paragraph.</p>
        <p>This is another paragraph.</p>
    </body>
</html>
```

The <head> part gives information *about* the web page: author, title, keywords, styles...

The <body> part of the HTML page is displayed by the browser

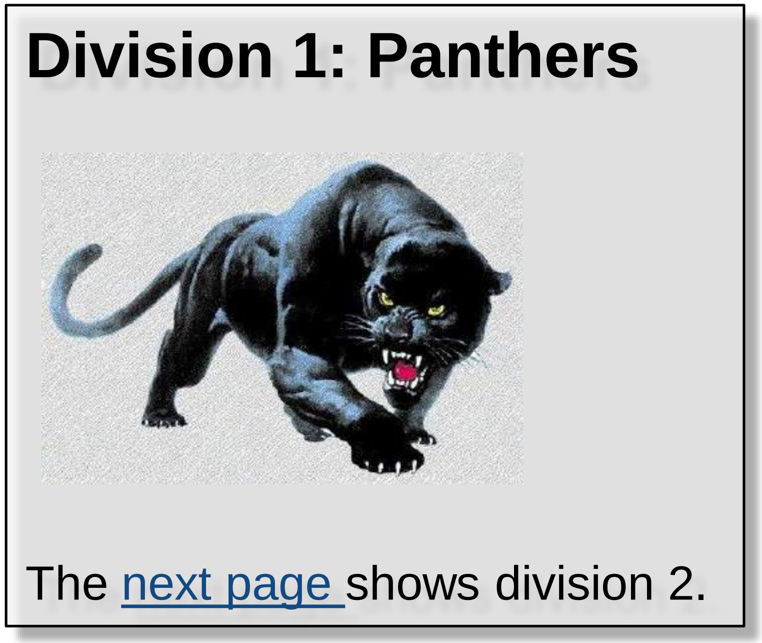
Most HTML constructs use begin/end tags: **<tag>**content...**</tag>**

# HTML links and images

**page1.html**

```
<!DOCTYPE html>
<html>
  <head> ... </head>
<body>
<h1>Division 1: Panthers</h1>
<img src="panthers.jpg"
     width="104" height="142">

<p>The <a href="page2.html">next
page</a> shows division 2.
</p>
</body>
</html>
```

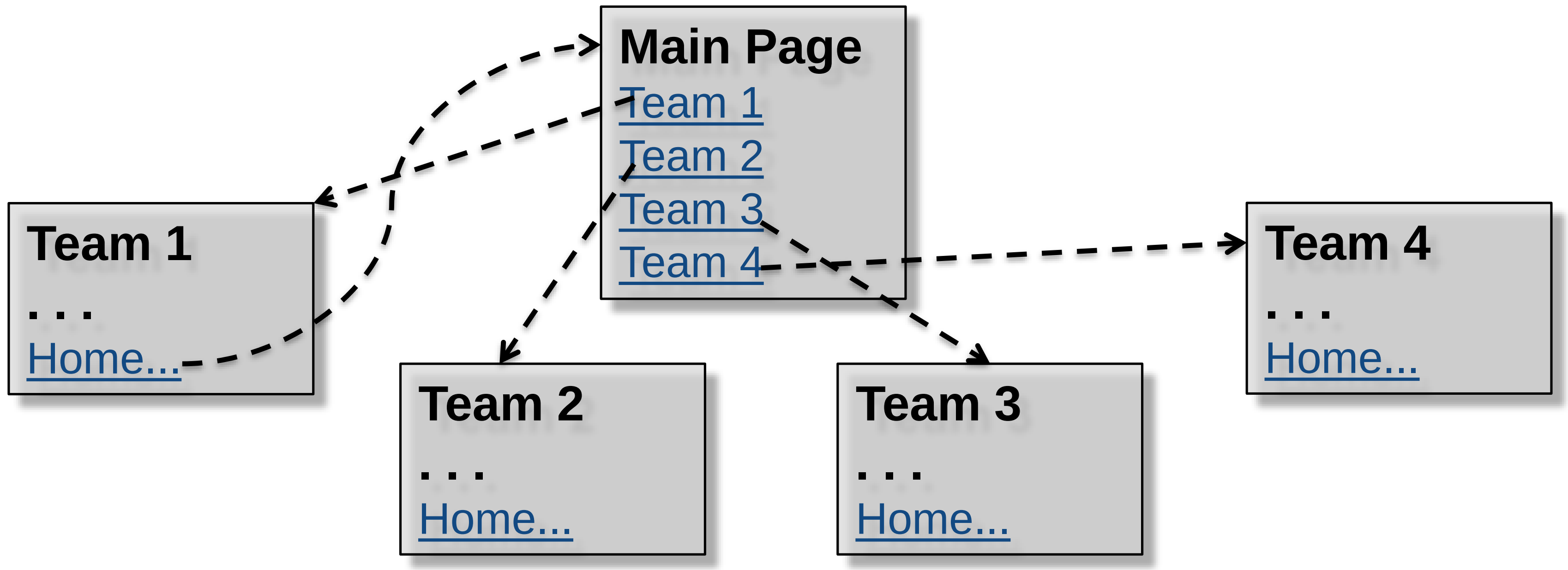


# Division 1: Panthers

The next page shows division 2.

See http://w3schools.com

## Your assignment



- Need to generate at least:
  - – one main page (with links to all children)
  - – 4-8 *child* web pages (with link back to parent)