Lecture Notes:

- **Files** and **HTTP** are protocols. A **file** means that it's a file on the user's home computer and not on the internet. So, say that a person has a file called index.html on their home computer. When they open index.html with a browser, the browser knows that index.html is a file and so it doesn't need to go on the internet to search for index.html.
- HTTP means that the browser needs to go onto the internet to find the HTTP file. The browser goes to a DNS to get the IP address of the given url. To make a url available on the internet, we need to use a web server.
- E.g.
  This is a file:
  

  This is HTTP:
  

- To use flask in Python, you need to import it using the statement: **from flask import Flask**
  flask is a module/package in Python that contains several related files, one of which is Flask.
- The difference between HTTP and HTTPS is that HTTPS is that HTTPS is more secure. With HTTPS, any information you send using the get request will be encrypted, where as with HTTP, the information will not be encrypted.
- When using Flask, make sure that all your method/function names are different.
- Flask looks at all the different routing names to determine which to choose. It doesn't do top to bottom.
  E.g. Consider the below code snippet:

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def func1():
        return "Hello World"

@app.route('/<name>')
def func2(name):
        return "Hello {0)".format(name)

@app.route('/rick')
def func3():
        return "Hello Rick"

if (__name__ == "__main__"):
        app.run(debug = True)
```

  If the user starts a web server, runs this python program and then types in http://localhost:5000/rick, the program will run func3 as opposed to func2. This is because there is a route called /rick.

- **Note:** __name__ is a built-in variable which evaluates to the name of the current module. Furthermore, because there is no main() function in Python, when the command to run a python program is given to the interpreter, the code that is at level 0 indentation is to be executed. However, before doing that, it will define a few special variables. __name__ is one such special variable. If the source file is executed as the main program, the interpreter sets the __name__ variable to have a value "__main__". If this file is being imported from another module, __name__ will be set to the module's name. I.e. __name__ == "__main__" acts as the main() function if you are running that program directly. If you are running a program via another program, __name__ == "__main__" will not work.
  E.g. Consider these 2 python programs:

File1.py File2.py

```
1  print("File1 __name__ = %s" %__name__)
2
3  if __name__ == "__main__":
4      print("File1 is being run directly")
5  else:
6      print("File1 is being imported")
```

File1.py File2.py

```
1  import File1
2
3  print("File2 __name__ = %s" %__name__)
4
5  if __name__ == "__main__":
6      print("File2 is being run directly")
7  else:
8      print("File2 is being imported")
```

If I run File1.py, I get this output:

```
File1 __name__ = __main__
File1 is being run directly
```

If I run File2.py, I get this output:

```
File1 __name__ = File1
File1 is being imported
File2 __name__ = __main__
File2 is being run directly
```

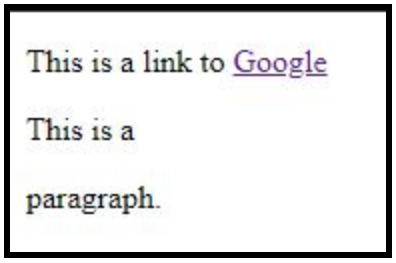
HTML:

- HTML stands for HyperText Markup Language. It is not a programming language. It is the markup language for creating web pages and is the building block of the web.
- All HTML files must end with the .html extension.
- index.html is the root or home page of a website.

- In general, HTML elements can be divided into two categories: block level and inline elements.
- **Inline elements** do not start a new line and take only the necessary width. Inline elements are those which only occupy the space bounded by the tags defining the element. They are usually within other HTML elements. Examples of inline elements are <img>, <a>.
- **Block-level elements** always start on a new line and take up the full width of a page, from left to right. A block-level element can take up one line or multiple lines and has a line break before and after the element. It can contain other block level elements as well as inline elements. Block level elements create larger structures than inline elements. Examples of block level elements are , <h1> - <h6> and .
- E.g. Consider the HTML code below:

```
<!DOCTYPE html>
<html>
<body>
    <p> This is a link to <a href="https://www.google.com"> Google </a> </p>
    <p> This is a <p> paragraph. </p> </p>
</body>
</html>
```

It looks like this on the browser:



Notice how for the line ** This is a link to Google ** the <a> tag doesn't start on a new line, but for the line ** This is a paragraph. ** the inner tag starts on a new line.

- All HTML tags can have **attributes** and attributes provide more information about an element. Attributes are always placed within the start tag. Some attributes are necessary, such as the href attribute for the <a> tag, while others are optional. Attributes are always formatted as key/value pairs. I.e. attribute="something"
  E.g. In the case of ** Link to Google **, href is the key and "https://www.google.com" is the value.
- All Web pages share a common structure:

```
<!DOCTYPE html>
<HTML>

  <HEAD>
    <TITLE> … </TITLE>
  </HEAD>

  <BODY>
  …
```

</BODY>

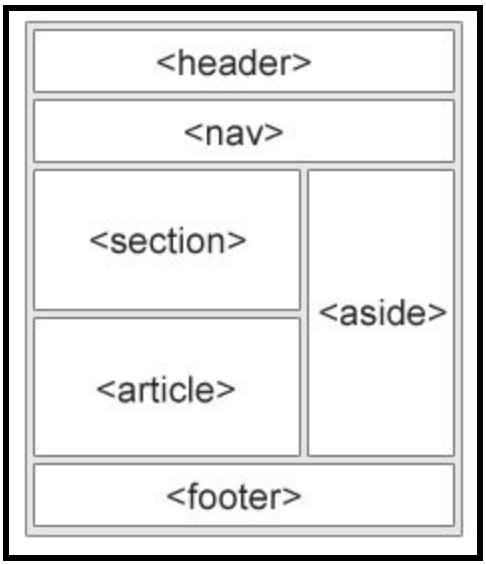
</HTML>

- All Web pages should contain a pair of <HTML>, <HEAD>, <TITLE>, and <BODY> tags.
- The **<TITLE>** tag is required in all HTML documents and it defines the title of the document. The <title> tag defines a title in the browser toolbar, provides a title for the page when it is added to favorites and displays a title for the page in search-engine results.
- The **class** attribute is used to define equal styles for elements with the same class name. This way, all HTML elements with the same class attribute will get the same style. The class attribute can be used on any HTML element. The class name is case sensitive. Different tags can have the same class name. A class cannot start with a number.
- The ** ** tag defines a division or a section in an HTML document. The element is often used as a container for other HTML elements to style them with CSS or to perform certain tasks with JavaScript.
- The **id** attribute specifies a unique id for an HTML element (the value must be unique within the HTML document). The id attribute can be used on any HTML element. The id value is case-sensitive. The id value must contain at least one character, and must not contain spaces. Furthermore, an id cannot start with a number. The difference between the id attribute and the class attribute is that an HTML element can only have one unique id that belongs to that single element, while a class name can be used by multiple elements.
  I.e. There can not be multiple of the same ids while there can be multiple of the same classes.
  HTML bookmarks are used to allow readers to jump to specific parts of a Web page. Bookmarks can be useful if your webpage is very long. To make a bookmark, you must first create the bookmark, and then add a link to it. When the link is clicked, the page will scroll to the location with the bookmark.
  E.g.
  1. First, create a bookmark with the id attribute:
     <h2 id="C4">Chapter 4</h2>
  2. Then, add a link to the bookmark ("Jump to Chapter 4"), from within the same page:
     Jump to Chapter 4
  3. Or, add a link to the bookmark ("Jump to Chapter 4"), from another page:
     Jump to Chapter 4

Semantic Elements:

- A **semantic element** clearly describes its meaning to both the browser and the developer.
- Examples of non-semantic elements: and . Tells nothing about its content.
- Examples of semantic elements: <form>, <table>, and <article>. Clearly defines its content.
- In HTML there are some semantic elements that can be used to define different parts of a web page:
  - <article>
  - <aside>
  - <details>

- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



- Here's a table that describes what each semantic element does:

Tag	**Description**
<article>	Defines an article
<aside>	Defines content aside from the page content
<details>	Defines additional details that the user can view or hide
<figcaption>	Defines a caption for a <figure> element
<figure>	Specifies self-contained content, like illustrations, diagrams, photos, code listings, etc.
<footer>	Defines a footer for a document or section
<header>	Specifies a header for a document or section
<main>	Specifies the main content of a document
<mark>	Defines marked/highlighted text

<nav>	Defines navigation links
<section>	Defines a section in a document
<summary>	Defines a visible heading for a <details> element
<time>	Defines a date/time

CSS:

- CSS stands for **Cascading Style Sheets**. It is a styling language that describes the presentation of HTML.
- In layman's terms, HTML provides the structure and the content of a website whereas CSS describes the look and feel of the site. One odd misconception that comes up once in a while is that some people think that HTML and CSS are the same thing. HTML and CSS are separate languages however they rely on one another and are thus often linked together in conjunction.
- **Note:** CSS properties must be spelled exactly the way they are listed. Like HTML, CSS will not complain or throw any errors if it is incorrect and your browser will not give any indication as to what is wrong with your code.
- A comment in CSS starts with /* and ends with */.
  E.g.

```
p {
  color: red;  /* Set text color to red */
}
```

- The main way of including CSS alongside an HTML file is by serving a separate CSS file to the HTML file of which your browser will read and serve accordingly. This is done by providing a <link> tag inside the <head> tag of your HTML code. This looks as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <link rel="stylesheet" href="style.css">       <!-- css file linked here -->
    <title>My Website</title>
</head>
<body>
    ...
</body>
```

  The <link> tag indicates that you are associating a separate file alongside the given HTML file. The rel attribute indicates that this file is a stylesheet or in other words a CSS file. The href attribute refers to the hyperlink reference and is the path to the stylesheet file so that the HTML file can find it. When normally using CSS, the location of the CSS file does not matter. However when using Flask, the stylesheet must be in a folder called **static** for Flask to use it properly.
- Using the style attribute inside an HTML tag is called an **inline style**. Although this functionality is supported by HTML5, it is not recommended as it makes your code less modular and less maintainable over time. Suppose you had many tags in your code with inline styles and one day you wished to change them. If all of your tags have inline styles, it would be a nightmare to go back through your code and change everything. As

a general rule of thumb, avoid inline styles wherever possible and also opt to include styles in an external CSS file. This is an example of inline style:

```
<div style="background-color: green">
   this is a div with green background
</div>
```

- A CSS rule-set consists of a selector and a declaration block. Each declaration block contains a property and a value such that the general form for a declaration block is property:value. The general syntax for css is:

```
selector{
declaration1;
declaration2;
…
declaration(n);
}
```

- E.g.

```
h1{
color:blue;
font-size:12px;
}
```

In this example, h1 is the selector, color:blue is the first declaration, and font-size:12px is the second declaration. In the first declaration, color is the property and blue is the value. In the second declaration, font-size is the property and 12px is the value. It means that all <h1> elements will be in the colour blue and have a font size of 12px.

- E.g.

```
p {
color:red;
text-align:center;
}
```

In this example, p is the selector, color:red is the first declaration with color being the property and red being the value and text-align:center is the second declaration with text-align being the property and center being the value. It means that all <p> elements will be center-aligned and with a red text color.

- CSS works by associating various style attributes to certain elements of HTML code. CSS can be bound to the following HTML elements:
    1. **Tags:**
        - The element selector selects HTML elements based on the element name.
        - E.g. Here, all <p> elements on the page will be center-aligned, with a red text color:

```
p {
  text-align: center;
  color: red;
}
```

    2. **Classes:**
        - The class selector selects HTML elements with a specific class attribute.
        - To select elements with a specific class, write a period (.) character, followed by the class name.

- E.g. In this example all HTML elements with class="center" will be red and center-aligned:

```
.center {
  text-align: center;
  color: red;
}
```

- You can also specify that only specific HTML elements should be affected by a class.
- E.g. In this example only <p> elements with class="center" will be center-aligned:

```
p.center {
  text-align: center;
  color: red;
}
```

3. **Ids:**
   - The id selector uses the id attribute of an HTML element to select a specific element.
   - The id of an element is unique within a page, so the id selector is used to select one unique element!
   - To select an element with a specific id, write a hash ,#, character, followed by the id of the element.
   - E.g. The CSS rule below will be applied to the HTML element with id="para1":

```
#para1 {
  text-align: center;
  color: red;
}
```

4. **Everything:**
   - The universal selector ,*, selects all HTML elements on the page.
   - This is usually used for global styles such as a font size, font family, font color.
   - E.g. The CSS rule below will affect every HTML element on the page:

```
* {
  text-align: center;
  color: blue;
}
```

5. **CSS Grouping Selector:**
   - The grouping selector selects all the HTML elements with the same style definitions. To group selectors, separate each selector with a comma.
   - E.g. In the following CSS code the h1, h2, and p elements have the same style definitions:

```
h1 {
  text-align: center;
  color: red;
}
h2 {
  text-align: center;
```

```
  color: red;
}
p {
  text-align: center;
  color: red;
}
```

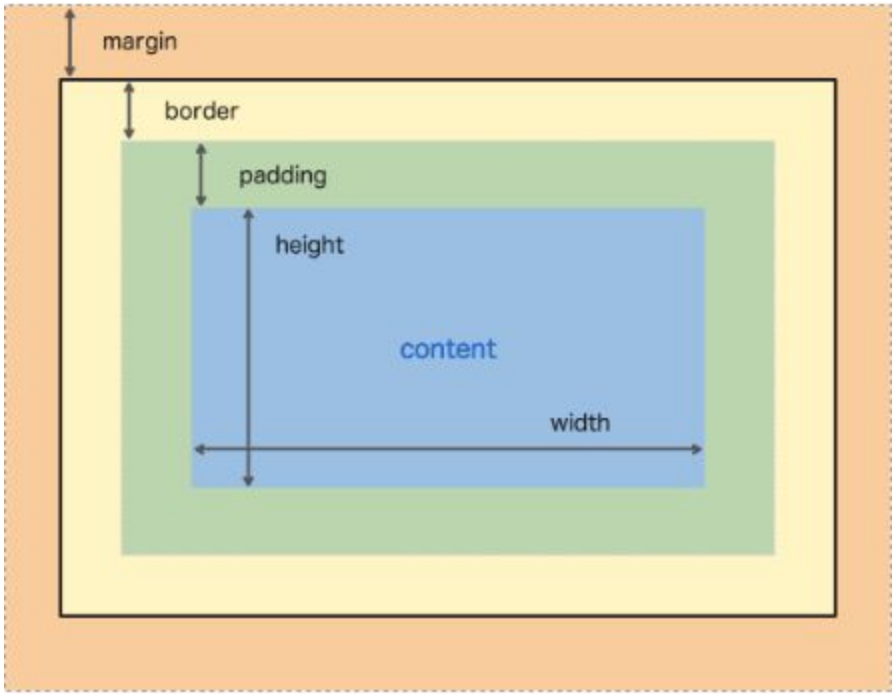
It will be better to group the selectors, to minimize the code.

We can group the selectors from the code above as such:

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

The CSS Box Model:

- Every element in HTML is made of a box that wraps around the element. Each box is made up of four properties as shown below:



- Properties:
  - **Margin:** The area outside of the border. This area is transparent.
  - **Border:** The area on the outskirts of the content.
  - **Padding:** The area between the border and the content; this area is transparent.
  - **Content:** The area where your actual content goes (e.g. text, images, other divs, etc).
- **Common CSS Properties:**
  Here is a table of common CSS properties

Property	Description	Values
background-color	Sets the background color of an element	color-rgb, color-hex, color-name, transparent

border	Sets the border around an element	border-width, border-style, border-color
visibility	Sets if an element should be visible or not	visible, hidden collapse
float	Sets where an image or a text will appear in another element	left, right, none
width, height	Sets dimensions of an element	none, length, %
font-family	Sets the font type of an element. In CSS, there are two types of font family names: 1. Generic family: A group of font families with a similar look (E.g. "Serif", "Monospace") 2. Font family: A specific font family (E.g. "Times New Roman", "Arial") You can have more than one font in a font-family property. The font-family property should hold several font names as a fallback system. If the browser does not support the first font, it tries the next font, and so on. Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available. If the name of a font family is more than one word, it must be in quotation marks, like "Times New Roman".	Arial, Times New Roman, Serif.
font-size	Sets the size of the font	%, px
font-weight	Sets how bold font should be	normal, bold, bolder, 100, 200
margin	Sets the space outside of an element’s borders	auto, length, %
padding	Sets the space inside of an element’s borders	auto, length, %
color	Sets the color of text (not the color of the background!)	color-rgb, color-hex, color-name, transparent
text-align	Sets the alignment of text	left, right, center, justify
text-decoration	Adds decorations to text (note: bold is not considered a decoration)	underline, strikethrough, blink
text-transform	Controls the letters of an element	uppercase, lowercase, capitalize, none
cursor	Sets the type of cursor to be displayed	default, pointer, crosshair, move