CSS Tutorial Part 1:

Introduction:

CSS adds style to tags in your html page. With HTML you told the browser what things were (e.g., this is a paragraph). Now you are telling the browser how things look (e.g., I want my paragraphs to have blue text with double-spacing and a purple background). To make all paragraphs have the style just mentioned, you'd add the following CSS code (I'll explain what each line means in detail later):

```
p {     background-color: purple;
        color: blue;
        line-height: 200%;
}
```

A. Adding Style to a Web Page (3 options):

You can add a style to your web page in 3 different ways:

Method 1. By including the style inline (e.g., in the tag itself in your html document), e.g.,:

```
<body>
    <p  style = "background-color: purple;  color: blue; line-height: %200;" > This is a blue paragraph with a purple background color </p>
    …
```

Method 2. By including the style in the head section of the web page, e.g.,:

```
<head>
        <meta charset="utf-8" />
        <title> The web page </title>
        <style >
            p {     background-color: purple;
                    color: blue;
                    line-height: 200%;
            }
        </style>
</head>
```

****Method 3**** By attaching a "style sheet" to your web page, and then placing all your styles in that new style sheet.

To do this, you first create a brand new file (not a web page, a brand new completely blank file). You put in this brand new completely blank file they style(s) you want (e.g.,:

```
p {     background-color: purple;
        color: blue;
        line-height: 200%;
}
```

Once the file contains a style, save it as blablabla.css. The blablabla part can be some other name, if you prefer, but the .css must be just that.

Now you've got to attach the style file to your html file. If you skip this step, the styles you've created in the css file won't be applied to the web page you've created. To attach the css stylesheet to the web page, in the head section, add a link to the style as such:

```
<head>
   <meta charset="utf-8" />
   <title>The web page </title>

   <link rel="stylesheet" href="blablabla.css">
</head>
```

All 3 methods will work. Method 1 allows for quick and easy changes to style, and method 2 has the advantage of keeping your styles within the same file as your html code, but can lead to very big html files, and but can get hard to keep track of when making changes.

Method 3, however, is my preferred method. Method 3 allows you to create a style sheet separate from your web page(s). Once the style sheet is created, you can include it in as many web pages as you wish. Thus, if you have a web site with a large number of web pages, and you want to change the look of the entire web site, you just need to change one style sheet. In addition, you can create a number of different style sheets with very different styles. Then you can completely change the look of your web page simply by changing which style sheet you link to it. For an example of how completely you can change the look of a web page simply by changing its style sheet, visit:

http://www.csszengarden.com/

Thus, for our tutorials and class project, I will insist you use a separate style sheet as specified using **Method 3** and place most of your styles in the stylesheet.

B. Creating Styles:

Now you know how to include a style. But you don't know how to create a style. So we'd better learn that. You must apply a style to html tags, so first create or download an html page:

Step 1 (for method 3):

Create a separate CSS file.

- Open a new file in notepad++/wrangler
- Save it with a .css extension (I will call mine tutorial.css, but you can call it whatever you like)

Step 2:

(Note that this is still for **method 3** to attach our style to the html page)

Attach the CSS file to your html file. You must attach the css file to the html file or the styles won't show up.

1. Open the HTML file in notepad++/wrangler
2. In between the <head> and the </head> tag, add the following line (with your file name):

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="utf-8" />
      <title>All Things Giraffe</title>

      <link rel="stylesheet" href="tutorial.css"/>
   </head>
```

Note: You can attach the same CSS file to more than one html file. That way the style you create for one html file can be used on other html files as well.

B.1. Creating styles for Tags:

Note: you can add style to 3 different types of elements:

1. The pre-existing tags on your page (e.g., <p>, <h1>,<body>,<ol>, etc.)
2. ids you create on your web page
3. classes you create (we'll see these later)

We'll discuss 2 and 3 further later. For now, we shall just add styles to the pre-existing tags.

You add style to everything on the page by creating a style for the body:

Step 4: Open the tutorial.css file in textedit/notepad.

Let's start with a style for our paragraphs:

To style a specific tag, we use that tag, and then put the style in between { and }, e.g.,

```
p {  style goes here…
}
```

Styling with color:

There are 17 basic colors. These are:

aqua, black, blue, fuchsia, gray, grey, green, lime, maroon,
navy, olive, purple, red, silver, teal, white, yellow.

There are actually 147 other colors that html and css recognize as standard names of colors (e.g., BurlyWood, PapayaWhip, OliveDrab (I'm not making this up)). But if you really want unlimited color choices, you'll need to learn more about RGB values, and specifically hex values. We'll talk about colors later. For now, let's stick with the basics so we can see what's going on.

Changing text properties:

Step 5: Change font color:

To change the font color, modify the style as such:

```
p {       color:  olive;
}
```

Save the css file. Load the HTML file (the web page) into the browser. The color of the font in paragraphs (and paragraphs only) should have changed to olive color.

If it didn't either your style has been defined incorrectly (did you remember the semicolon at the end of the line?) or the style sheet has been attached to the html page incorrectly.

Step 6: Changing font family:

To change the font family (e.g., arial, Helvetica, times new-roman, comic, etc.), add the following the the paragraph's style:

```
p {       color:  olive;
          font-family: arial, helvetica, sans-serif;
}
```

Save the css file. Load the HTML file (the web page) into the browser. The font in the paragraphs should have changed to arial or Helvetica font. (If not, make sure you've got a semi-colon at the end of each line and a dash between font-family.)

A note about font-family:

You can specify font-family to be any font you want. However, you want to stick with common fonts because the font must exist on everyone's computer. In the above example, we specified the font to be:

```
font-family: arial, Helvetica, sans-serif;
```

In this case, the browser will try to display arial. If arial doesn't exist on the computer, then it will try Helvetica, and if that doesn't work, it will use a default sans-serif font.

THE FONT MUST EXIST ON THE PERSON WHO LOOKS AT YOUR WEB PAGE'S COMPUTER, not yours. Yes, there are ways around this, but for now be aware that once you put your web page on a web server somewhere (other than on your computer), other people's browsers will download your html and css code onto their computer. They are not downloading an image of the web page in your browser, just the html and css code you've written. Then the browser on their computer will try to display the page as you've specified. However, it only has access to the fonts that are on the other person's computer.

Example: So say you pick an unusual font, like maridregia. Maridgregia is a really cool font with kitten pics for letters. However, your computer is the only computer in the entire world that has maridgregia on it (since it's a made-up font name, that would pretty much be the case). Now you post your web page to a server and

someone downloads it. They, of course, don't have maridregia on their computer, so when the browser goes to display the text, it will be displayed with the browser's default font, which is usually Times New Roman.

If you do decide to go with an unusual font, make sure you include a few back-up fonts (like the above example with Helvetica and sans-serif as back-up fonts) so that if the user's computer doesn't have the unusual font you specified, it will use the more common back-up font you chose.

Step 6: Changing font size:

To change the font size, add the following the the paragraph's style:

```
p {     color:  olive;
        font-family: arial, helvetica, sans-serif;
        font-size: small;
}
```

Save the css file. Load the HTML file (the web page) into the browser. The font may (or may not) have changed in size – if the web page's default font size is small, you may not notice a difference.

To make sure you can actually see a difference, let's add a style for the h1 tags in your css style sheet:

```
h1 {    font-size: 625%;

}
```

Save the css file. Load the HTML file (the web page) into the browser. You should definitely see a difference in the font size of the h1 elements now!

A note about font-size:

You can specify font-size as:

- xx-small
- x-small
- small
- medium
- large
- x-large
- xx-large
- smaller
- larger
- px ***(avoid using!!)***
- %

% is percent of the default size. So, for instance, if the default size is small, setting:

```
font-size: 200%;
```

would result in the font being set to twice as big as the small font.

smaller and **larger** set the font size to be smaller than the default font size and larger than the default font size, respectively.

px is the font size in pixels. You probably want to avoid using this. People with low vision often increase the font size on their web pages or apps so they can read it more easily (they use control +. Try it some night when your eyes are getting tired). With px, the font size is set and won't increase, whereas when setting the font size using "small" or "200%", the font size will increase. In essence, px sets the font size absolutely, so those with low vision can't adjust it, whereas the other options set the font size relatively (relative to the default size), so if we change the default size, we change all the font sizes set relatively as well.

Step 7: Changing font style:

To change the font style, add the following to the h1 style:

```
h1 {    font-size: 625%;
        font-style: italic;
}
```

Save the css file. Load the HTML file (the web page) into the browser. The h1 element's font should be italic.

You can also set the font style to:

- normal
- oblique (no one uses this one – it looks just like italic)

Step 8: Changing font weight:

To change the font weight, add the following to the h1 style:

```
h1 {    font-size: 625%;
        font-style: italic;
        font-weight: bold;
}
```

Save the css file. Load the HTML file (the web page) into the browser. The h1 element's font should be bold.

You can also set the font weight to:

- normal
- bold
- bolder
- lighter
- 100
- 200
- 300
- 400 *(same weight as normal)*
- 500
- 600
- 700 *(same weight as bold)*
- 800
- 900

Step 9: Changing font famly (to Times new roman):

Let's make sure the font family of our h1 elements is Times new roman. To do this, we must put quotes around "Times New Roman" so that it is interpreted as one word. Add to the h1 style:

```
h1 {    font-size: 625%;
        font-style: italic;
        font-weight: bold;
        font-family: "times new roman", serif;
}
```

Save the css file. Load the HTML file (the web page) into the browser. You may not see a difference in the h1 elements because the default font is times new roman.

Changing text properties and line height:

Step 10: Change text alignment:

Let's change the text alignment of the h1 elements to the right:

```
h1 {    font-size: 625%;
        font-style: italic;
        font-weight: bold;
        font-family: "times new roman", serif;
        text-align: right;
}
```

And let's change the text alignment of our paragraphs to be lined up on the left:

```
p {     color:  olive;
        font-family: arial, helvetica, sans-serif;
        font-size: small;
        text-align: left;
}
```

Save the css file. Refresh the html file in the browser. You should see a different alignments – the h1 element should be aligned to the right, and the paragraphs should be left.

You can also set text-align to:

- center
- left
- right

Step 11: Change text indent

Text indent changes the indent of the first line of an element (e.g., think paragraphs – the first line is often indented a bit). Let's do that:

```
p {     color:  olive;
        font-family: arial, helvetica, sans-serif;
        font-size: small;
        text-align: left;
        text-indent: 25px;
}
```

Save the css file. Refresh the html file in the browser. You should see a different alignments – the h1 element should be aligned to the right, and the paragraphs should be left.

Aside: Text-decoration

If you make something on your web page blink, I will hurt you, but here's how it's done with css:

Add to a style:

```
text-decoration: underline;
```

You can also set the text decoration to be:

- blink (I think, meaning I really hope!!! this has been deprecated on most browsers now)
- overline
- line-through
- underline (this one can quickly become confusing, because links on your web page are usually underlined, so if you set other text to be underlined, people looking at your web page will assume that text is a link as well.

Text indent changes the indent of the first line of an element (e.g., think paragraphs – the first line is often indented a bit). Let's do that:

Step 12: Change line height:

This controls the height between two lines of text in a paragraph. In this case we're using 120% of the default height between two lines. To specify line-height for the paragraph style, add:

```
p {     color:  olive;
        font-family: arial, helvetica, sans-serif;
        font-size: small;
        text-align: left;
        text-indent: 25px;
        line-height:120%

}
```

Save the css file. Reload the html file in your browser and see the results of line-height. Feel free to play with it a bit.

Borders, Margins, Padding, Background Images

Adding a border and border styles:

Add a border:

You can modify a border's style, width, and color. You can also specify that a tag's style only has a border on one, two, or three sides. Try adding a border around your h1 elements:

```
h1 {    border-style: solid;
        border-width: 4px;
        border-color: green;
}
```

Save the css file and look at the html file in the browser. You should see a green border around your h1 elements that is 4 pixels wide and solid.

Other border **styles** include:

- solid,
- dotted,
- dashed,
- double,
- groove,
- ridge,
- inset,
- outset.

Border width is specified in pixels, and can be anything you like. You can also use preset widths, but I almost always use pixel widths (e.g., border-width: 3px; (note there's no space between the 3 and the px)):

- thin
- medium
- thick

Border color can be any of the standard colors, or you can specify the color using hex values or rgb values (which we will discuss later).

Adjusting a border on one side:

You can also add a border or set border properties on individual sides. So, for instance, youcould say:

```
h2 {
        border-bottom-style: dashed;
        border-bottom-width: 2px;
        border-bottom-color: purple;
}
```

You could also say:

```
h3 {
```

```
        border-top-style:dotted;
        border-right-style:solid;
        border-bottom-style:dotted;
        border-left-style:solid;
}
```

Feel free to try this and see how it looks.

Adding Padding and Margins:

Padding and margins are different. To clarify the difference:

- Padding is the space between the text and the border (inside the element).
- Margin is the space between the border and the other elements on the page (outside the element)

You can adjust both separately, and again, you can set the padding and margin for the top, left, bottom, and right separately.

```
h1 {    border-style: solid;
        border-width: 4px;
        border-color: green;

        padding-left: 10px;
        padding-right: 10px;
        padding-bottom: 10px;
        padding-top: 40px;
        margin: 0px;
}
```

With this style, you should see padding inside the border between the border and the text, and you should see no margins between the border and other elements.

Special Note: Margins: auto; for centering

Sometimes you want to center an element. For instance, you may create a header or a special paragraph that you want centered on the page. To center an element (not just text, but an entire element) you can set the margins to auto. This will automatically adjust the margins on both sides to be equal, so that the element stays centered even when you resize your browser. To try this, first set the width of an element to be a specific size (in pixels). To do this you'd use:

```
width: 500px;
```

Include this in a style you create for some element (I'll use h2):

```
h2 {    width: 500px;
        border-style: solid;
        border-width: 5px;
        border-color: green;

        padding: 15px;
        margin: auto;
}
```

Try including this in your css stylesheet. Load your html page into a browser and see what your h2 elements look like. Resize the browser. See how the h2 elements stay centered?

Adding background color and image:

Add a background color:

You can also add a background color to individual elements by using the style: background-color.

```
h1 {    border-style: solid;
        border-width: 4px;
        border-color: green;
        padding: 10px;
        margin: 0px;
        width: 800px;
        background-color: #F39642;
}
```

Now the whole h1 should have a background color

Add a background image:

Color is cool, but images are cooler. You can add a background image to the style of a tag. To do this, go find a good background image on the web. Make sure you place the image in the same folder as the css file you're working on (or follow the rules for getting the browser to locate the image. Now add it to your element:

```
h1 {    border-style: solid;
        border-width: 4px;
        border-color: green;
        padding: 10px;
        width: 500px;
        margin: auto;
        background-color:#F39642;
        background-image: url(savannahbg.jpg);
}
```

The h1 elements should now have a background image.

The default is for the background image to "tile" or repeat again and again across and down in the background (like tiles on your bathroom wall). If you use a small picture as your background image, you'll see this.

Background-repeat:

If you want a background image (that other elements float on top of), but you don't want it to tile, or you only want it to tile in one direction, you can set the background-repeat property.

If you only want the background image to show up once, you'd specify that you don't want it to repeat using no-repeat:

```
background-repeat: no-repeat;
```

If you specify a repeat, you might want to set the background image's position using background-position:

```
background-position: left top;
```

You can use the following to set the background image's position:

- left top
- left center
- left bottom
- right top
- right center
- right bottom
- center top

- center center
- center bottom

And you can set the background image's position using x% y% (the top left is 0% 0% and the bottom right is 100% 100%

```
background-position: 20% 40%
```

Or if you want to specify using pixels:

```
background-position: 50px 100px;
```

If you wish to have the background image tile (repeat), but only in one direction, you can use:

```
background-repeat: repeat-y;
```

or

```
background-repeat: repeat-x;
```

You can use background-position to specify the position for the repeat-x and repeat-y as well.

Adding default style to the entire page:

Modifying the body's style:

If you want your entire web page to have a default background color, specific font, font color, etc. you can style the body tag. For example you'd add to your css style sheet the following style:

```
body {
        background-color: brown;
        font-size: small;
        font-family: arial, helvetica, sans-serif;
}
```

Now the default for the entire web page is to have a brown background with a small font that's arial. You can override this default by styling specific elements.

Color using Hex Values:

RGB Values:

First, on the computer, colors are represented using RGB values. RGB stands for red-green-blue values. So when representing color as a series of numbers, the first number is the amount of red, the second is the amount of green, and the third is the amount of blue. Each amount is represented with 2 digits. So an RGB value might look like this:

#740199

74 represents the amount we want red to glow on the computer, 01 represents the amount we want green to glow, and 99 represents the amount we want blue to glow. The above color is purplish, with a bit more blue than red in the purple.

As another example:

#009911

00 represents the amount red should glow (not at all!), 99 represents the amount of green that should glow, and 11 represents the amount of blue that should glow (not much). Thus this color is pretty much a solid green.

And another example:

#000000

Has no red glowing, no green glowing, and no blue glowing, nothing is glowing, which means that there is no color whatsoever. With nothing glowing, the color we'd see would be black.

Hex numbes:

In the above example, I only used the digits 0 through 9 to represent each number. However, to represent color using RGB values, we use hex numbers. In our counting system, we count as follows:

0,1,2,3,4,5,6,7,8,9 and then we go to two digits: 10

In hex counting, we keep going past 9:

0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F and then we go to two digits.

Just as 8 is greater than 7 and 0 is the smallest single digit number and 9 is the largest single digit number in our counting system, 8 is greater than 7, A is greater than 9, and F is greater than E in the hex counting system. Equally, 0 is the smallest single digit number and F is the largest single digit number in the hex counting system,

So now to represent RGB colors using hex numbers (So red, green, and blue are still each represented with 2 digits):

#FF00FF

Has A LOT of red glowing, no green glowing, and A LOT of blue glowing, giving you a bright pure purple color.

Another example would be:

#CA210D

Has CA red glowing, which is a lot of red, 21 green glowing, which isn't much green, and 0D blue glowing, which isn't much blue (just like 09 wouldn't be a lot of blue). Thus this color is a pretty bright, but not completely pure, red color.

#21F3DA

Has very little red glowing (21) a lot of green (F3) and a lot, but not quite as much blue glowing (DA). Thus the color would be Bluish Green.

Final color note:

RGB colors are light-based colors, not pigment based colors. You spent kindergarten playing with pigment-based colors, and in doing so you learned that the primary colors are red, yellow, and blue, and if you mixed yellow and blue you got green. We are now dealing with light based colors. The primary colors are now red, green, and blue (didn't see that coming, did you?). Even more oddly, if you mix red and green you get yellow in light based colors (really didn't see that one coming.) Yep, if you've got:

#FFFF00

You've got all the red glowing, all the green glowing, and no blue glowing, so what you see is yellow. Try it. I'm not kidding.

Equally, if you have:

#FFFFFF

You have all the red glowing, all the green glowing, and all the blue glowing at full force. Everything glowing at its brightest in light based colors results in white.