**Unit 1 – HTML & CSS**

**Lesson 1: How the Internet Works**

The Web is a collection of HTML (hyper text markup language) documents. It was started in the early 1990s, and now has over 30 billion pages! Anything can exist on the web, including Text, Music, PDFs, Videos and HTML. X The web can be broken down in to 3 major pieces - A computer with a **browser** (such as Safari, Internet Explorer, or Chrome), **servers**, remotely located computers that host the files that make up the web, and the **Internet**, the go between for the browser and servers.

X The browser makes requests to servers using HTTP protocols via the Internet, the browser then reads HTML, interpreting and displaying the page.

**Lesson 2: Coding Basics**

Coding is writing in a language that computers can understand. Computers are considered "dumb" because they simply follow directions. So it is very important to give them clear directions with proper formatting (syntax) or the computer will return errors.

Coding is made up of:

1. **Tags** Tags are always contained within a set of angled brackets, and usually will occur in a set- opening and closing. They look like this: <tag>content</tag>. So for example, to create bold text, you could say: <b>Title</b>
2. **Elements**An Element is the information within a set of tags. Elements can have either inline or block behavior.
3. **Attributes**Attributes are information assigned to an element.
4. **Anchors**Anchors are used to create links. An anchor tag looks like this <b>a</b> To create a link, you could say: < a href = "www...." > text that links >
5. **White Space**White Space does not naturally exist in coding, so you must force it with break and paragraph tags.

**Coding Structure**

HTML is considered "tree-like" because elements are nested inside of each other, with nested elements inheriting the traits of the parent element. Nested elements are indented - this is seen in written HTML in the wave shape on the left side of the page.

**Lesson 3: Adding Style**

HTML has underlying structure of boxes, CSS allows us to change the CSS look and feel of the boxes in a systematic way. This is important because it allows us to **avoid repetition** when coding - meaning changes are easier to make and we'll have fewer errors.

**CSS**

You style HTML using CSS (cascading style sheets). It gives programmers a way to control the style of related HTML elements. This is done by giving similar HTML elements the same **class** name and then specifying the style that should apply to that class.

**Unit 2 – Python**

**Lesson 1: Introduction to “Serious” Programming**

Unlike machines, which are typically physically designed to do a single task, computers are **programmable** to perform multiple tasks. Computers can pretty much do anything as long as we can write the sequence of instructions telling them what to do.

We call these sequences of instructions **programs.** Everything from web browsers to games and mobile apps are simply a precise sequence of steps, that a person wrote using one of the various *programming languages* for the computer to follow and execute.

**Grammar**

Like all languages, computer languages have **grammar**, rules specifying what is “correct” and “incorrect” - too. As mentioned in an earlier chapter computers are pretty stupid, so we have to write programs using correct grammar for them to understand what we are telling them to do.

**Python**

Python is one of the programming languages, and is read by a computer using a *Python Intrepreter*. We call python instructions *expressions.*

**Lesson 2: Variables and Strings**

**Variables** let programmers assign names to values, which we can then refer to our programs. This helps programs make more sense for humans, as we can assign names that help us understand what is happening within a program. Variables can change within a program when assigned a new value.

We give variables values with the following grammar:

name\_of\_variable = value\_to\_assign, for example x = 2.

**Strings** are a sequence of characters. Strings are denoted by single or double quotes. Python reads whatever is in the quotes as a single input. Characters in strings are numbered (starting with 0), and can be referenced in an expression to be manipulated.

In Python, **2** is a number while **"2"** is a string.

The code **2 + 2** would give **4**.

The code **"2" + "2"** would give **"2””2”**

**Lesson 3: Input, functions, and Outputs**

**Functions** take in i**nputs (**data that we want to alter in some way), perform an action on that data, creating an **output**. Functions help programmers avoid repetition because we can define it once and refer to it many times. The **return** keyword tells Python exactly what the function should produce as output. Functions are defined with the following syntax:

**Def** name\_of\_function(*function parameters*):

Action to be performed

Return output

For example:

Def full\_name(first string, second string):

name = first string + “ “ + second string

return name

Function parameters will be replaced by actual values when the function is called.

To call the function:

Print full\_name(Chauncey, Robinson)

>>>> Chauncey Robinson

**2.4 Control Flow & Loops: if/else and while**

**Control flows** help us execute different code based on certain conditions.

**If/else** - keywords that let us tell what we want to run in different situations

**While** – loop that lets us repeat a block of code many times until a certain condition is met. Code under loops is always indented.

**==, !=, >, <, >=, <=**: Python *comparison operators* that return True or False.

**2.5 Debugging**

Debugging in an important step of writing code. The last line of Python error messages are the best place to start, as it will tell you what line the problem occurs on.

**2.6 Structured Data: Lists & For Loops**

Lists are one way that we structure data. Lists are helpful because we can then run through all the elements of a list using a **for** loop.

**2.7 How to Solve Problems**

Problems can be very complicated, but become much more manageable once broken down in to single functions. The first step in solving large problems is to understand the problem. Next, figure out what are the inputs, and what are the outputs.

**Python Vocabulary**

Python vocabulary includes print, return, +, -, =, and ==.

String.find(‘substring’) – will return the position of the first instance of that string, to find a certain instance of an occurrence string.find(‘substring, #)

Def

Return

Select sub-sequences in string with [#:#]

If

Else

For

While