Tinesha N. Conway

July , 2020

**Module 1 Objective**:

To read and reflect on the nature of code and creation through digitized data and software.

**Module 1**



Answer the following questions in a .txt file saved as: **“What\_is\_code\_reflection**”.

**In your own words what is a computer really doing whenever you type a letter on a keyboard?**

Computer data is made up code; for instance, if I wanted to type the letter “Z”—yes it would type it, but there are several things going on in the background **BEFORE** I typed the letter “Z”.

First, the computer is waiting; and if we used the OSI (Open System Interconnection) Model you would know the computer stops at the first layer out of seven, which is the Application Level—you have to have a program open in order to type.

Next you type the letter Z and in mille-seconds the Z is formatted on your screen. But that is not all it does—in the mille-second that Z was produced, is what is called the second layer of the OSI, which is Presentation Layer. This layer formats the letters so **YOU** can read them—remember this is machine, it doesn’t see the words I am typing the same way. It sees them as bits, bytes and nibbles. As stated from Google:

**“A bit is a single binary digit - a '1' or a '0'. A byte is 8 bits…so just like you can make a number like 1635 from four decimal digits - you can make a number like 10110110 out of eight binary digits. A nibble (also spelled 'nybble') is 4 bits… half of a byte.”**

The letter Z has 122 bytes. Which is **01011010** in binary. All of the “0’s” and “1’s” is how the computer see the letter Z and we see the actual letter.

The other five layers come into play once we hook up the internet and start sending and receiving things.

**In your own words, how do computers use input and output to create experiences?**

That is a very wide and varied question to throw out there in my opinion. Everyone is very different in their experiences when it comes to the computer. But I will use one example.

I’ll use video games, played on the computer. As you know, the computer games today are not like the first video game called “Tennis” originated back in October 18th, 1958. Later on, a video game called “Pong” in 1972 was made by Atari. Basically it was a one dimensional court with two sticks on either side which you moved back and forth to hit the little dot which represented the ball. Hours of fun!! Today, kids and adults alike would be bored with it in less than five minutes.

Now, we have controllers with many buttons and rules that far exceed moving the sticks up and down. The ‘input’ is using the controllers and ‘output’ is what we see our controllers do on the screen. Some games are so vivid and realistic, they had to put “**Parental Warnings**” on the packages because children literally thought if they stabbed/shot someone in real life they would just get up and walk away. An example is 18-year-old Devin Moore after playing **Grand Theft Auto** killed three (3) police officers because he did not want to go to jail.

Now, he is just one incident the gaming industry has had; there are many others. When we see and play these video games hours on end, staying up past bed times—I’ve heard of 10 gamers who literally died from playing video games.

Now all of these examples are horrific but then make you stand and ask the question: “**What are we doing to ourselves?**” I’m 100% sure the makers of these gaming systems had no idea you would have to put these warnings on them—but you do. And now it is highly recommended to everyone who is an avid gamer they must get up every couple of hours to walk around and stretch because of hours of lack of motion and the torture on the brain from lack of sleep.

What experiences are these creating—I always play the devil’s advocate, where I look at both the good and the bad side of what is being discussed; here, we are discussing the experience of what is happening between a person’s mind and the computer.

While it is good fun to play games, when you are too young, you don’t have the mental acuity to know **ALL** that is right/wrong, good/bad and most importantly what is real/imaginary. These games while giving a great experience can also cause great harm if not used correctly.

**Explain why software products are essentially “infinitely reproducible nothings”.**

From the Bloomberg, entitled: “**What is Coding**, Business Week, June 11, 2015, by Paul Ford:

**“Data management is the problem that programming is supposed to solve. But of course now that we have computers everywhere, we keep generating more data, which requires more programming, and so forth.**

**It’s a hell of a problem with no end in sight. This is why people in technology make so much money. Not only do they sell infinitely reproducible nothings, but they sell so many of them that they actually have to come up with new categories of infinitely reproducible nothings just to handle what happened with the last batch.”**

I believe they are this way for two (2) reasons:

* Computer programmers have to keep changing/re-writing code on these programs because they “keep changing”—being made better, faster more competent to read by the user. This i because people need bigger and better.
* They are just that—nothing…computers are machines. The sooner some people not in the industry realize that the better off everyone will be.

These are and always will be machines. They cannot think or do anything without **FIRST** a human behind a keyboard giving it instructions.

A machine is not now or in the near future going to be “**alive**”; it does not think for itself and thereby does not know what you want it to do unless you tell it. It is there by ignorant of why it was made unless it is given instructions.

**Explain in your own words how code becomes software**.

As I explained earlier, computers don’t write like us—where we see letters, the computer sees millions and billions of binary code.

Coders don’t write in binary code, they write the instructions on what the software will do in a language—such as C++, Java, COBOL, etc. These languages are written in a way that gives the computer specific instructions on what we want it to do. Whereas the language looks foreign to the average onlooker, the computer understands it perfectly.

The languages are then translated into computer code [binary] known as software.

**Explain “algorithm” in your own words**.

An algorithm is a set (or multiple sets) of instructions that have decisive instructions to solve a series of problems, create some/any thing and/or perform a computation.

An example would be asking Siri/Alexa a question and having them respond—how? do you ask. Because it takes hundreds of millions to algorithms to change your words into text, which then gets converted into symbols and then those symbols are checked against a large database to give you an answer. All of this is done with algorithms.

**Explain the “DRY” principle.**

**DRY** stands for “**Don’t Repeat Yourself**”. This is a major principle of coding but many don’t use it. If we had to repeat and/or retype each and every algorithm that is placed into the computer—nothing would ever get done. This is why we need the computer to learn and repeat.

Since reading the article, I feel, once a code is learned, we should find new ways of writing code that will add to the ease and quickness of using the program without really changing the code.

This way when we name, do, and create things we can teach the computer to take the saved information that was put in and reuse it—I believe some programs have this function, but I couldn’t really tell you which ones right now.

**In your own words, what is** **object-oriented programming**?

Object Oriented Programming also known as OOP, first known as: **using the computer to write programs “using the idea of objects” to “act for” data and methods—in other words.**

From the definition it seems OOP uses classes, these are “pre-objects” which contain attributes when explained become an object.

An object simply defines one certain instance of a “thing belonging to a class”.

The first OOP language was [generally] known as Simula. It introduced the “idea of “objects”—these were collections of info that are treated as a singular existence. OOP languages, like Java, helps to make it easier to organize data/code that is more adaptable for any projects including larger projects.

**In your own words explain data and how it relates to code.**

**Data:** facts and information collected for examination/research.

**Code:** used in communication/information to process a system of rules that convert info to sound/image/letters/words or gestures into another form for communication or storage.

**Data relating to codes** means the process of moving gathered info and/or observations to a set of close-knit classification. Codes are ideas that link data with assumptions/theory. They can either be preconceived by the researcher or emerge initially from the coding process.

In short-short terms, data relating to code is gathering info and mining it and seeing what works, what doesn’t, making hypotheses and taking what works—I think! (Smile)

**What is a framework?**

**Framework:** a basic structure underlying a system, concept or text.

A framework is a broad and general term that can be applied to almost anything—as the article says, from Web Framework to Network Framework.

Frameworks have been used since the internet has begun. It is “reusable” code that provides specific functionality that you can use in your programs. Frameworks are there to keep you from “reinventing” the wheel, so to speak.

The more you use it the more you become proficient in it. There are different programs that allow you to use frameworks such as Java Collections Framework, Modular Testing Framework, Hybrid Testing Framework and Django.

**What is debugging?**

In basic terms, it is removing bugs from a program. You have to analyze the code to see where you made a mistake or whether or not you have a bug either in the computer’s OS (Operating System) or just in the program you are using.

The best programs to use would be Visual Studio Debugger, Eclipse or the GDB which stands for the GNU Debugger (or **G**NU’s **N**ot **U**nix Debugger).

**Why do you want to learn coding?**

I have fallen in love with computers since the later 80’s and the early 90’s—by the mid 90’s I had my first computer system, printer, scanner and Wacom Tablet. I attended Fashion Institute of Technology—but not for fashion, for Exhibit & Design. I learned the intricate knowledge of designing and installing store windows, rooms, exhibits, events, etc. From sketching, to lighting, designing and creating things; mostly from scratch.

So my love for computers was always there along with how to use them—I always thought learning how and why they worked was above my head; until I took a Networking Class within the past year and learned a lot about how the internet worked and then a new passion arose in me. This is why I want to learn, use and teach Web Design. It is a way to not only share what I know with future enthusiasts but also learn more about the subject.

There are so many projects I would like to learn how to create; I will start small at first, such as show off some of my personal projects in a blog; my projects are small and simple, but after creating a few of them myself I have found they are a lot of work and can become complicated the more you want to add or do to them.

I then would like to start a website—about what? I have no idea, YET! I would like to get my feet wet in working with a fashion company learning step by step what I need to do in order to create a live and working web site as well as become a TA (Teacher’s Assistant) to learn all that is needed to allow students to grow not only in themselves and their knowledge, but in the fact they made it as far as they did.

All in all, I look forward to being in the class and meeting everyone.