Considering Computer Code as Foreign Language

Kelsey Grinnan

Candidate for the degree

Bachelor of Sciences

Submitted in partial fulfilment of the requirements for

College Honors

[illegible]
Daniel Falabella, Ph. D

[illegible]
Wil Lindsay, M.F.A

[illegible]
Paul Clark, Chaplain

F. Wilbur Gingrich Library
Special Collections Department
Albright College

Release of Senior Thesis

I hereby grant to the Special Collections Department of the F. Wilbur Gingrich Library at Albright College the nonexclusive right and privilege to reproduce, disseminate, or otherwise preserve the Senior Honors Thesis described below in any noncommercial manner that furthers the educational, research or public service purposes of Albright College. Copyright privileges remain with me, the author.

Title: Considering Computer Code as Foreign Language

Signature of Author: [illegible] Date: May 9, 2018

Printed Name of Author: Kelsey Grinnan

Street Address: 55 Patricia Lane

City, State, Zip Code: Mt. Laurel, NJ 08054

Albright College

Considering Computer Code as Foreign Language

An Honors Thesis

Kelsey Grinnan

Candidate for Bachelor of Sciences

April 30, 2018

Considering Computer Code as Foreign Language

Introduction

Computer code, and the technological advancement that comes with it, has been a huge impact on society. An alternative name for computer code is computer language which can bring to mind a peculiar subject: Could these various computer languages be considered foreign languages? With such a mindset, one would have to look at how code fits into language itself. Foreign languages, typically, are considered to be natural languages while computer languages are artificial which have kept some for looking at this controversial topic more closely. Thinkers like Noam Chomsky have created views of language that contributes to such a question, adding mathematical elements to all languages, be they artificial or natural.

In addition, if one views computer language as a foreign language, there are several implications that are created. A second question arises as part of these implications: If computer code is a foreign language, then can it be considered for a foreign language requirement for an educational purpose? One must closely analyze both coding language and foreign language, comparing the two in the light of formal languages to get to this question. Then one must discuss the aspects of a foreign language class, including spoken word, teaching styles, and culture. Finally, it is important to take into consideration the thoughts of the ordinary person on this highly controversial topic. Both of these questions formulate into two halves of one whole: coding as a foreign language and computer coding as meeting the requirements for a modern foreign language. There are those that believe not only that computer codes are not a language, but that if it were to count as a credit, it would be for mathematics or the science (Turkle 132).

However, the opposite—that it is a foreign language and could be taught as such—could very well be true.

A Brief Summary of Coding Terms

Computer code is used in a great number of devices that have become commonplace in the everyday lives of people. The issue that arises is many would not know the intricacies of computer jargon. In order to alleviate and provide some understanding of it. To begin, some common terms used in coding are strings and variables. Strings are lengths of texts that can be used in coding. Often, these strings are used to print out to a user or as an indicator of other data within the program. Meanwhile, variables are values that can be changed; often they are declared in a unique type. Such types are text, which is used for strings, integers, decimals, and many more. Sometimes these types can even vary by language. These types cannot be changed, but the specific value of the variable can. Computer languages are highly mathematical and algorithmic. These algorithms are used to run programs in what are called loops until an answer removes them. If no answers are reached, the program will crash.

There are also a number of programming languages such as C, C++, Java, and Python which are object-orient languages. These languages focus on objects and data rather than actions and logic, meaning that it will take in data to process and then output more data. Another common language that is highly unique is HTML or Hypertext Markup Language which is used to code websites sometimes with the addition of Java to add unique effects. All of these are artificial languages that are commonly used today, Java in particular.

What is Language?

Noam Chomsky

One way to look at language is through the eyes of linguist Noam Chomsky. Chomsky has completely re-evaluated the way that people look at language, especially in terms of grammar. During the 1950s and 60s, Chomsky was the dominant force in linguistics and he, along with some colleagues, “actually proved theorems about certain pieces of algebra connected with their models of language” (Donnelly 76). According to Chomsky, children “have an innate knowledge of certain principles that guide them in developing the grammar of their language” (Dubuc). This is what some refer to as insight. Insight helps people gain a deep understanding of concepts; mimicking the human brain’s insight is how some people are beginning to further the development of AI ("Community Research and Development Information Service - CORDIS"). In addition, a major part of Noam Chomsky’s theories on linguistics was to simplify and abstract what was once an empirical domain with an approach that “ignores meaning entirely” (Jäger 1956). Chomsky’s theories are mathematical in nature and, despite that, were successful and influential in linguistics. This was in addition to its influences on theoretical computer science (Jäger 1956). It is this linguistic effect that helped support the idea that computer code is a language and not a simple tool. People seem to focus on the humanity of language, where Chomsky has focused on viewing these languages through mathematical lenses.

Coding relies heavily on mathematics to do its functions, which displays the important correlation between Chomsky’s hierarchy of language and computer coding. While some believed that Chomsky’s connection to algebra must be a movement to overcome, being seen as an “analysis of semantics and common sense” (Donnelly 76), it is actually useful to viewing code as a foreign language. If a programmer wants to run their program on a loop, be it printing out a message or having a system that runs through possible moves in chess until it comes to the best option, it must be set up in math. There has to be a specific number set up to keep it from being continuous. When speaking to a computer, one must remember that their language relies heavily on order and context. If forgotten, a computer will not know what to reply in return and would not print out or it would not go through the directory of moves—in other words, something got lost in translation. If a programmer forgets a certain piece of context, the entire program can fall apart.

By some accounts, time has been "wasted" in computer science trying to put "grammar into these machines" because grammar "is what we were taught to think about" (Donnelly 44). One could easily get lost discussing the semantics of equating periods to semi-colons and other such minor mechanics. Instead, one should look at the rules of grammar. When some argue against this, they point out that computer code contains different grammatical rules than typically spoken tongue. These differences, which makes coding as a foreign language hard to accept, but it is not so different when explored. For instance, just like translating between two languages, there are issues translating between English and code as well. To further elaborate, the word 'but' is meant to "negate the instance that first seems appropriate, which is precisely how a computer 'understands' that word" (Donnelly 50). Assumptions that humans make naturally do need to be programmed into the code for a computer to get the full translation (Donnelly 51). This is where the specific grammars of coding come into place. While a program is given English, it has to be translated to them, even if they give English—or any other language—in its response.

According to Chomsky, "grammar is a set of rules—preferably a finite set [...] that specify the grammatical strings of symbols" (Chomsky 92). It is this mathematical solution, using terms and ideas similar to coding like strings, to grammar and language that gives us a better idea of the grammar in code. This is looking at if the code does in fact already have grammar which are the finite, shorthand commands and styles that must be put into each code. Chomsky's Hierarchy of languages is a greatly important work for such matters. The Chomsky Hierarchy added mathematical elements to what was traditionally seen as part of the humanities. Even to this day, this hierarchy influences computer coding even going as far as to be used to help properly code artificial intelligence. These are working under the assumption that language is defined by grammar, but this grammar is not the typical grammar that is taught in school. When we look at these language hierarchies, we are thinking in terms of ∑ to NT. ∑ represents a "finite vocabulary" (Jäger 1956) and NT is "a second finite vocabulary of extra symbols called non-terminals" (Jäger 1957). This is the grammar that these languages follow which Chomsky defined is untraditional because "the rules as stated in traditional grammar books do not lend themselves to logical analysis" (Chomsky 92). To determine computer code as a foreign language, one must first analyze if there are the four necessary elements present to define a grammar. Computer code must have a finite vocabulary of symbols called terminals, a second finite vocabulary of extra symbols called non-terminals, a start symbol, and a finite set of rules (Jäger 1957); the last of which, coding most definitely has.

Entire codes will break if one does not follow the very specific rules; the rest of hierarchies are a bit different. Coding varies from language to language; for example, HTML code cannot even be written without "<!DOCTYPE html>" at the beginning of the file. This means that that document type the computer will read is HTML and not another language. Meanwhile, Java cannot start without having some identifier that stated to be a "public class". Both of these appear to be starting symbols, which is why it is fortunate that starting symbols can be strings of words. Therefore, Java and HTML can fall into the class of grammar. Codes like Python and C++ may be challenging to define as they do not have a definite starting symbol like

Java and HTML. In general, codes often include open brackets ("{") to designate where a program starts. If that is included in code, perhaps a form of backtracking can be used to read code properly for grammatical purposes. That means that different codes could be classified into different parts of the Chomsky Hierarchy. As for Java and HTML, those two languages would fit nicely into computably enumerable languages because they are code and they are runnable by a Turing Machine, which is an "abstract machine" (Korb). This means that the machine is not necessarily physical, but grammars can be put theoretically put through it.

To further elaborate on Chomsky's Hierarchy which these grammars work upon, there are four classes within it: "the class of recursively enumerable languages", "the class of context-sensitive languages", "the class of context-free languages", and "the class of regular languages" (Okubo 43). According to the Formal Language Theory, the class of recursively enumerable languages is defined as "any kind of formal, algorithmic procedure that can be precisely defined can also be expressed by some grammar" (Jäger 1957). All recursively enumerable languages can have every one of their strings returned by a Turing Machine result due to the fact that languages that are computably enumerable can be run through a Turing Machine (Jäger 1957). In comparison, "context-sensitive grammars are those grammars where the left-hand side of each rule is never longer than the right-hand side. In the context-free language class, "all rules take the form of $A \rightarrow \beta$" where the arrow stands for when the variable A may be replaced by the variable β where A and β are strings of symbols from $\sum$ to NT. This leads to a complex web of replacements to reach ultimate goals. Typically, when not a context-free language, one can assume the rule of "$\alpha \rightarrow \beta$" (Jäger 1957). Finally, the class of "regular languages are those languages that are defined by regular grammars" meaning the rules will either take the form of "$A \rightarrow a$" or "$A \rightarrow aB$" with A and B representing nonterminal symbols and a as a terminal symbol

(Jäger 1958). Regular grammars are also considered to be context-free (Jäger 1958). When speaking of what are called natural languages, "the vocabulary is usually identified with words, morphemes or sounds" (Jäger 1956). When creating these, Chomsky added mathematical rules to language, adding an algorithmic element to it. Computer code relies heavily on similar algorithmic methods to run.

In addition, when talking about these hierarchies, *w* represents a string and *G* represents a grammar that refers to "a quadruple <∑,*NT*,*S*,*R*>" (Jäger 1957). In such a quadruple, "*S* is the start symbol and R is the rules" (Jäger 1957). When using a Turing Machine, a grammar is analyzed for being decidable. Decidable means that the Turing Machine—or an equivalent—that answers if "string *w* is generated by a given grammar *G*" (Jäger 1957). To be considered decidable, the question must be answered either negatively or positively in a limited amount of time (Jäger 1957). It gives a time limit for it to complete. Computably enumerable languages are actually semi-decidable, meaning that a positive result outputs only when *w* is generated by *G* (Jäger 1957). Otherwise, *G* will either output a different result or run forever, which is not quite decidable, hence semi-decidable. The example used in one text even references computer programs, stating that a program will output 'yes' upon termination (Jäger 1957). As this implies, computer languages do have a place in Chomsky's language hierarchy.

In truth, the best way to test if coding is a language would be to utilize a Turing machine. A Turing Machine, being an abstract machine, is not meant to execute programs, but to "define or describe algorithms precisely" (Korb). Truthfully, it is meant to be used for coding itself and a Turing Machine can read code without question in order to account for its algorithmic capability. However, looking at it from the perspective of the Chomsky Hierarchy, one can see similarities that may have been ignored due to the idea that the hierarchy is simply for natural language. If a Turing Machine could be run utilizing a computer code as a natural language, what would the result be? An experiment would have to be run, but to some extent, this is already known. According to one source, it is not computably enumerable languages, but regular grammars that coding falls into (O'Regan). Truthfully, there could be a greater exploration of this idea. Research agrees that coding should fall into one of the levels of Chomsky's hierarchy, be it regular grammars or what has been discussed above with Java and HTML.

Bernard Block

Noam Chomsky is not the only view on language that can be referenced, however. For instance, Bernard Bloch in his "Outline of Linguistic Analysis" cites language as a "system of arbitrary vocal symbols by means of which a social group cooperates". Bernard Bloch was a linguist with a long number of articles that studied languages, from colloquialism to how the pause relates to English grammar. Bloch, like Chomsky, has a more mathematical view of language. When analyzing phonemics, which is the measurement of any distinct units of sound that can separate one word from another, he casts asides all "semantic and psychological criteria" as they need not be involved in the "theoretical foundation of phonemics" (Bloch 5). Bloch takes phonemes and language down to a mathematical science, breaking it down to its most basic components. Bloch's view of language does heavily focus on the spoken word which can conflict with the idea of computer coding as a language.

In the Classroom

Spoken Word

A major part of a foreign language course is the spoken language itself. There are some who would argue that coding language is different from the spoken word. When teaching languages, it serves the purpose of "communication capability" or "the ability to transfer information successfully" (Milková 176). This does, of course, resemble the transfer of information between programmer and machine. Programming is simply communication to computers and learning code is learning to communicate with the computer so it does as asked. Still, some may argue that for computer code to count as a foreign language it must be spoken, however, sign language is not spoken but communicated through hand gestures. There is also another language that is meant to be written and not traditionally spoken; this language is Latin. Considered a dead language, Latin is widely taught throughout many school districts, offered in high schools, colleges, universities, and even select middle schools. While one can speak Latin, it is traditionally written. In fact, Albright College's own Latin classes do not require an oral exam while every other language class at Albright does. The same exception can be made for computer code especially when writing code and testing programs is an alternative to an oral exam that are already a core part of every coding class.

Another argument is that coders do not speak to each other in code; that does not mean that they cannot. In order to speak a language, "one must have control of close to 100 percent of the phonology and 75 percent or more of the grammar, but a mere 1 percent of the vocabulary" for most situations (Hook). When discussing projects, those who know code can easily state what they want to happen through code. Usually, however, pseudocode is used for shorthand writing and speaking. Pseudocode helps to more easily convey a message that will be given to a computer to a person—akin to slang or even the combination of English and other languages like Spanish. However, code can most easily be transmitted through writing as it is not naturally a spoken tongue. Binary, what the computer actually translates code into, would sound more like Morse code when spoken than something translatable using math as it is a series of ones and zeroes. In contrast, humans cannot process binary as fast as a computer reads its native tongue. Despite all this, these are things that still can be taught and processed as language.

ASL

ASL is considered a “natural language with structural properties akin to those of spoken languages” (Corina 1258), but this was not always the case. Despite this, some felt that ASL was closer to vague nonlinguistic gesturing than language. Those in the deaf community viewed themselves as their own unique subculture (Regan 241), of which ASL was a large portion as their “vernacular language” (Regan 243). To determine if ASL was more aligned with spoken word, the left hemisphere of the brain was studied as the left hemisphere is believed to be used for language (Corina 1258). Through studying different stimuli on both the left and right brain hemisphere, the results suggested that ASL is part of the “linguistic nature of movement” (Corina 126). In addition, when a deaf signer received left hemisphere brain damage, he had no issues doing pantomime gestures, but could not comprehend sign language as well afterwards (Corina 1260). This shows that comprehension of the gestures used for language were not activating the same area of the brain as using pantomime. These research results solidified ASL as a language.

The pathological way to view deafness is as a disability, putting deaf people as lesser than hearing people (Reagan 242). The Deaf community define themselves as a linguistic minority (Reagan 241). Due to the views on deafness as a disability and feeling that ASL was closer to pantomime than language, this community had to fight for their acknowledgement of their culture. Their culture does also heavily rely on ASL to not just be their common language, but to pass down their history “orally”, meaning they tell their history using sign language. Due to these views, the Deaf community often rejects such objects like cochlear implants as the use of them would “[reject] ASL as a legitimate language” which, in addition, delegitimizes their culture (Reagan 248).The study done was highly influential in legitimizing the case for Deaf culture. While there has not been a similar study done for computer coding, a study could look deeper into this using coding stimuli instead of gestures and signs. To align computer code in the left hemisphere would help aid in the idea of it as language. If coding showed similar results to what ASL did when it was evaluated then it could solidify it as a language. That would require that samples of people communicating through code, both in its common form and perhaps spoken pseudocode, though the actual spoken nature of it may sway the results, would have to align in the left hemisphere.

Testing

For classroom settings, there are those looking into studying foreign languages to learn more about coding, and vice versa. In fact, computer code can be taught in the style of foreign language instruction. This is built off the idea that both foreign language and computer language deal in syntax and semantics. Understanding “syntax and semantics” of programming languages “is the cornerstone” of developing algorithms (Milková 172). For coding, syntax is the combination of symbols as defined by a strict set of rules which is unique to each programming language. Usually, though, there are similarities between each language. Similarly, the syntax of languages are rules for how a sentence can be structured. For example, at the University of Hradec Králové in the Czech Republic, to explore the concept of using algorithms to teach foreign languages, researchers taught students in an Algorithms and Data Structure class (Milková 173). Students are taught to use “Czech pseudo-language (based on Pascal programming language)” to write papers (Milková 173). This clearly takes a basis in using pseudo-coding language to solve coding problems and communicate them to others in addition to a basic idea of how to write a problem. These students go on to use programs to evaluate their algorithms as well as answer their foreign language questions (Milková 176). Tests have also been given the foreign language treatment; included in this course were five typical foreign language tests. There is first the Gapped Text, commonly known as Fill-In-The-Blank, where "students are presented with a question containing blanks and must provide the missing word or text" (Milková 176). Next, there is Jumbled Sentences where "students are present with a question containing blanks for which they must provide the missing text by selecting word or phrase from a list" (Milková 176). Thirdly, there is Multiple Matching which involves students being "presented with two lists and must match terms in one list with terms or definitions in the other list" (Milková 176). Fourthly, there is the traditional Multiple Choice, which goes without explanation. Finally, there is Open Close where "students are required to fill the gaps in text" without a word bank and must try to fill in the gap correctly (Milková 176). These are all different styles of testing that are used for foreign language testing that can also be used for testing coding. In each of these, students are given a coding sample, sometimes with an answer bank, in order to figure out where to place the correct answer in the code.

All of these can be useful in supplementing traditional teaching methods in computer coding classes. It is quite likely that there already are professors using some of these foreign language test styles in coding classes. Through Chomsky's words, one can assume there is a distinct reliance on the algorithmic aspects of coding when it comes to language. Teachers have used the syntax and semantics of foreign language to further the teaching of programming. Coding becomes a sibling to foreign language in this way, legitimizing its linguistic status. It shows that it can be taught in ways similar and inspired by foreign language teaching, which helps show that it could be an ample comparison. This teaching style could be applied to future coding classes in order to make them more resemble foreign language courses. Currently, many people are leaving colleges with degrees in coding feeling like they do not know how to code; perhaps the application of foreign language teaching will help students feel like they truly know what they are doing.

CULA

Chomsky's universal language and insight play a role in one learning system called the CULA, or "Computer User Learning Aptitude". What is unique about the CULA is that it involves its own belief in "computerse" which is defined as "a language used to communicate or interact with a computer" (Warner 995). The CULA is for people trying to understand all aspects of "computerese" before "determining whether the technology is useful or easy to use" (Warner 993). It was also created with the observation that language is used as a middleman to the computer and user which makes learning language crucial to technological dependence (Warner 991) in order to help technology acceptance. The CULA deals heavily with anxiety and fear around dealing with the computer and what it entails, including code, but also in dealing with code as a language in itself. One cannot argue that computer code is a natural language, but it does not illegitimatize it as a language. The counterargument is certainly there, but communication is key to language which programming languages support between man and machine. It may be that the line between natural language and artificial is smaller than one may think. In addition, going by CULA standards, learning computer languages *does* compare to the learning of second languages.

Chomsky views language as universal, and when one learns their first language as a child, they are utilizing their innate ability to learn language at that age as well as learning about the world around them simultaneously (Warner 996). Second language acquisition involves a certain level of cognitive maturity where “learners already have knowledge of their world before trying to learn to communicate with a different set of symbols and vocabulary” (Warner 996). Code and other computer technologies include not just syntactical rules, but vocabularies that are unique to each piece of technology. Computer codes are “theoretically processed in a similar manner as any human language” by the people who use them speak with various technology (Warner 999). CULA is specifically meant to measure the aptitude of someone to “computerese”. A language barrier will form with a low CULA score (Warner 999). In fact, computer technologies are making it easier to communicate with machines in ways that greater resemble natural language than even code, such as the “extensive use of graphical user interfaces” (Warner 999), or GUIs for short. Such GUIs use command prompts that resemble code to speak with a machine directly or can be used for code. The CULA—it should be notes—is focused on all facets of computer systems provide such as “hardware, software, programming, and applications like the Internet” (Warner 996). To the CULA creators, it is not just programming that is considered a foreign language, but the whole of “computerese”, which more people are becoming well versed in. All of it has a hand in “computer literacy, an extension of traditional literacy” which is becoming a crucial skill for people to learn (Warner 995). A computer literacy course is featured as a requirement for some majors at Albright; it is “approached as a humanized communication process exemplified by the terminology used in the literature from the earliest use of computers” (Warner 995). The world continues to make advancements and computer literacy is crucial to even the workforce (Warner 995). Technology is booming and newer generations learn how to use it from birth, but programming can still remain a blind spot despite some efforts from schools and online games.

Culture

Examples

In addition to spoken language portions and testing, many foreign language classes include units on culture, be that watching movies made in the subject language, discussing topics that include specific dialects, or learning about traditions. As seen with ASL, language is highly important to recognizing a culture and, additionally, it is a matter that must be addressed for a classroom setting. As of now, there is no question that there is a computer culture. While that may suffice, in truth one must look at if there is a computer *coding* culture. However, as the idea of computer code as language is not yet widely accepted some things of this nature are harder to find. These things would have to be built up if computer code was to be accepted as a foreign language. Currently, there are only a few cases of something that could be considered acceptable to a foreign language course. One group, called WaveLab is planning to include "all the code to reproduce all the figures in [their] public wavelet articles" (Buckheit 1). This is meant to include coding as an official part of their research, not just a mean to the results. Even the article where they spoke about their plans included a large section of code. They believe in fully sharing computational research, though they compare it more to a mathematician publishing a theorem "without giving the proof" (Buckheit 6). While what WaveLab wants is not exact to the ideas of publishing academic works fully in computer language, they want more inclusion of code in their own works. The issue with this is their argument is for something computational and not communicative.

Projects that more closely relate come from such groups like the *School for Poetic Computation* which works "to explore the intersections of code, design, hardware and theory — focusing especially on artistic intervention", according to their website ("SFPC"). Artists like Ben Fry combine AS code with design in order to convey information. He also is developing

Processing which is an "open source programming environment" to teach computational design (*About | Ben Fry*). In addition, there is the affect that programming language has on design as a whole. In essence, coding languages allow for a great adaptability when designing (Burry 8). Scripting, a lesser form of coding, has been used in coding and now design to allow for this (Burry 246). Assets of using scripting include "productivity", finding a final result, to a "voyage of discovery" (Burry 32). Each of these are causing culture to form in their own ways, becoming physical in a world where many think that online culture is simply an extension of other cultures (Burry 32). While it is a work in progress, this is understandable as many would not see code as necessary to culture. Latin is not spoken, but is believed to serve a culture in linguistics and sciences, but coding is still the language of computers. One college student could easily have both a desktop PC and a laptop, an e-reader, a television, and often a gaming console. The digital world constantly surrounds those who do not know how to speak it.

Artificial Intelligence

Any strong opinions against believing in code as foreign language may come from the perception that there is a lack of emotion in programmed code. Even Chomsky's research on language has relied on removing emotion from it. To most, though, language is seen not just as a matter of syntax and semantics, but as an ethical, emotional issue. For example, artificial intelligence does play a role in supporting computer code as a foreign language, though it may not seem like it at first and taps into such emotional issues. In particular, strong AI and its resemblance to humans can help people look towards this. Strong AI, studied by Carnegie-Mellon, is defined as dealing with "biological plausibility" by attempting to get as close to actual human systems (Lucci). Programmed technology can be built based on human brain function which means such technology is getting increasingly human. It is coding that runs these AIs that help them recognize words of other languages, like English and what to do in response. A machine may understand English, just as someone may understand Spanish, but they will process it in their code; it is the machine's native tongue.

One of the reasons people have a hard time accepting coding as foreign language is the way academic linguists convey language. AI work, according to one expert, "flies in the face of linguistics" because they "are trying to formalize language in the absence of knowledge" (Donnelly 76). Linguists often refuse to support AI work because what they want to do is in "opposition to a system that relies on plans and goals and knowledge and context" (Donnelly 76). Due to reasoning like this, when thinking of Artificial Intelligence, some may see it as a moral issue. Much like how CULA addresses anxiety about technology, AI can cause anxiety in people due to their own moral beliefs. In reaction to the rising amount of AI and regularly used technology, it should be no feat to see that it is necessary to know code in the same way we have adapted to introducing Spanish on signs or how other non-English speaking countries often require English courses throughout school. It is especially important because systems like AI is starting to be programmed increasingly more human.

General Opinion

Florida Bill

There are clear connections between computer code and foreign language, but the opinions of people can also affect such a matter. It is important to look not just at peer-reviewed voices on the matter, but that of ordinary people. In 2016, a Florida bill was proposed "that would have authorized high schools to offer coding classes to fulfill foreign language credits" (Tate). The initial Florida bill passed in the Senate, but not have the same success in the Florida

House of Representatives (Postal). The bill entailed school districts provide computer programming classes of "'sufficient rigor' that they could be swapped for foreign language courses" (Postal). Meanwhile, Texas did pass a bill that allowed the substitution. This is only taken into consideration, however, under the circumstance that a student had performed poorly in a prior foreign language requirement (Tate). This may seem insulting at first—that code is an option seen as easier or an alternative to "real" language—but it is a step in the right direction. A mother described the benefits of this program as that the "majority of kids will never use the language they learn, nor will they ever become fluent in it" (Tate). Computer language, in her opinion, "could possibly open the door to a career in computer programming that might not have otherwise been an option for them" (Tate). Many fields need the expertise of coders: from information technicians to web developers, people versed in various computer languages are desired. In comparison, it is valuable when a person is bilingual when applying for a job.

Another parent believes that coding is simply a vocational skill (Tate), but many companies are "begging people with coding knowledge to work for them" (Tate). Coding knowledge and computer literacy are essential in a job market where technically illiterate managers are no longer meeting their worth for employers (Warner 995). Thus, coding knowledge and bilingual skill are both held in high regard when it comes to resumes. In fact, the current Microsoft chairman John Thompson said that he believed allowing computer code as a substitute for foreign language was a "great idea" (Hatter). Even in teaching, where information technology, or IT, is becoming a portion of a teacher's training, some are considering including computer code as part of their training program as well (Amiri). While the Florida bill did not pass, change is being discussed, but there are clearly people against it.

There is still an aggression towards seeing coding as a foreign language; instead, people want to see it as a trade skill. It is not an invalid argument, but as aforementioned, it is an aspect of computer literacy. While it may be an ethical debate, feeling as though communication must be between people, for some this aggression may be a manifested anxiety. There were some "advocates of foreign language" who were against the Florida bill. One retired Florida university professor used Twitter as a platform to say, "Safeguard communication w[ith] the rest of the world" at the height of the bill reboot (Postal). There is a computer anxiety stemming from such a tweet of loaded words, brimming with ethical and social worry about the impact of the bill. Both language anxiety and computer anxiety have extensive studies and "share similar characteristics" (Warner 997). When it comes to language anxiety, most will have a lack of appreciation for the language and will find the language "difficult to use" which is similar to the "adverse effects" had on a computer system's possible use by someone with computer anxiety. It is typical for computer anxiety, which ought to be viewed as a "state-based anxiety", to occur before "any knowledge of a specific technology is attain or processed" (Warner 997). While someone may have some basic understanding of Word, it is not the same as knowing a computer code. As humans communicate increasingly with machines, it becomes more useful to speak to them rather than not knowing how it functions outside of turning it on and off again. Computer anxiety can be a steady stream from such fear to outright fear and the justification of those emotions.

Survey

Results

An anonymous online survey was put out to students and faculty of Albright as well as several people outside the campus through email in an effort to determine some values about how people view code. The goal of the survey was to get responses from people who are tangible, not just the authors of educated articles. Their responses help shape how people feel in regard to the matter at hand. A mix of about eighty people above the ages of eighteen were asked if they believed code was a foreign language, given samples of code, and then asked if their opinion had changed. Two kinds of coding were given in the survey. There were two samples of Java code and one sample of HTML. Many did not recognize what exactly the code did, but some did have at least an understanding of it. Others did know what the HTML would command the machine to display, but not what the Java would run and vice versa and some did know both. When asked the second time, they were asked if their opinion on computer code as a foreign language had changed and why. No one changed their opinion upon being given the code, despite its intent to get people to view code in a new light. Most survey participants agree that code could be considered a foreign language and, in fact, believe as much already from what garnered from their responses. The majority of people who responded that computer code was not a foreign language were actually faculty members, with a wide age range but mainly falling into the category of 45-54-year-olds. It does seem that there is a correlation between those who are older and may have a graduate school education. Of the twenty-one faculty members who answered, seventeen have postgraduate degrees. It does not represent the entirety of the faculty, as there are a number of people who said yes as well and this is worth analyzing.

As for graduate school, it was a fairly even split, with nearly 53% siding for computer code being counted as a foreign language, and almost 47% against. Meanwhile, from those with the highest education being a high school, there were only 20% who were against the idea. Most were college students, while about 20% were forty-five years of age or older. It shows that many of the negative responses are coming from those who might have been further educated by liberal arts, though the survey did not go into specific majors. One may see this as ignorance to the subject, but one cannot ignore the fact that of graduate students there was a majority in favor of considering code a foreign language. In addition, a number of those that said yes were college students in the midst of their undergrad education, but still being given an enriching education. Many of the ones who answered that coding was not a language both before and after code was given, argued that language was for humans to communicate with humans.

There are those unwilling to accept the idea of speaking to a machine, despite AI being coded more human, as discussed. Younger generations seem to reflect a better openness to accepting code as a foreign language. Of twenty-eight 18-24-year-olds, only one thought that code should not be counted as a foreign language meaning 96.86% of that age group agree on the matter. Perhaps having been raised around technology, this younger generation seems to have some acceptance of the idea of computer code as a unique linguistic entity. Coded machines are a part of their everyday life and have, albeit stereotypically, a greater computer literacy than prior generations. Having grown up with the technology, anxiety towards such concepts tend to be lessened as they are familiar with it. Even those who did not understand the code were ready to accept it as a foreign language.

Responses

Each person in the survey was asked why they answered the way they did. One response that was received when asked about if their opinion before and after the code had changed said, "I agreed that it should be considered a foreign language. Computers operate in ways I do not understand, it is truly foreign to me, and would take considerable effort to do so, even to a 'conversational' level. Computer code should for certain be considered a foreign language". People are open to viewing coding as a foreign language, from all ages and walks of life. Still, another said that they didn't think their feelings on the matter could be explained away by a simple statement of "It certainly seems to have the rules needed to be counted as its own language". In truth, this is what this paper was written for. To describe the minute details and workings of how computer code is a foreign language through both rules and natural relationship to foreign language. It comes from a hierarchal perspective and adaptation to foreign language classes. These arguments were presented not only to affirm those who agreed with the idea of coding as a foreign language but those who may not yet be convinced. As shown with the survey, giving samplings of code is not much to convince a person otherwise of what they already believe.

Conclusion

Often, if a person did not recognize code, but had responded that they did consider it a different language, it served their own evidence as to why computer code could be considered a foreign language. While this is the initial idea some people may have, the truth goes far deeper than that. It is an argument that would be easy to break down. However, another argument propositioned from the responses was that computer code was too mathematical and was a physical science while linguists belong to the humanities. Opposition to the Florida bill that was proposed offered the idea of Arkansas's "computer science flex credit" where a mandatory math or science credit could be substituted by a computer science credit (Postal). While the latter of the survey results deals with what has been discussed above, Noam Chomsky has shown that language is far more algorithmic than some people may be led to believe.

All of this points to computer code being a foreign language. A computer culture that could be taught in a class is present in addition to a history that can also be taught. There is a readily available comparison between computer codes to the hierarchy of language. In addition, there are those who agree that computer coding should count towards foreign language credits. Currently, this is still a controversial topic, but in order to help humanity continue to move forward with the ever-evolving computer culture, it is necessary for people to learn code. There are already attempts to educate the current generation through games that resemble coding. Computer literacy seems to be at a high with younger generations, but people are still avoidant of accepting coding as a language despite cultural signs that give a nod to computer code being considered a language. In the hit Netflix show *Stranger Things*, which is set in the 80s, a character says, “Why don’t I teach you French while we’re at it?” in reference to another character asking to learn Basic—an early computer language—in a few minutes. This speaks of the volume of difficulty in learning code, comparable to learning the complex language of French that programmers face when learning their new language. Clearly, the show’s writers have made the connection between foreign language and computer language if, perhaps, unintentionally. Accepting code as a foreign language is a way to broaden horizons and give students greater access to a great skill to have in the future. Just as having some knowledge of Spanish is useful, so too could having some knowledge of coding be useful. Adjusting classes to accommodate a computer science language credit could go a variety of different ways. Using Arkansas’ flex credit idea, but keeping it to Florida’s core idea of placing it within language would work. While every school could and should implement coding as a foreign language, Albright College can now take this chance to adopt this different way of thinking. There is a real chance to forget traditional boxes, borders, and limits of language set an example for other schools, and be uniquely Albright.

Works Cited

About | Ben Fry, benfry.com/about/.

Bloch, Bernard. “A Set of Postulates for Phonemic Analysis.” Language, vol. 24, no. 1, 1948, p. 3., doi:10.2307/410284.

Burry, Mark Prof, and Mark Burry. Scripting Cultures: Architectural Design and Programming. Wiley, 2011.

Buckheit, Jonathan B., and David L. Donoho. “WaveLab and Reproducible Research.” Wavelets and Statistics Lecture Notes in Statistics, 1995, pp. 55–81., doi:10.1007/978-1-4612-2544-7_5.

Chomsky, Noam, and George A. Miller. “Finite State Languages.” Information and Control, vol. 1, no. 2, 1958, pp. 91–112., doi:10.1016/s0019-9958(58)90082-2.

“Community Research and Development Information Service - CORDIS.” European Commission : CORDIS : News and Events : Why human brains hold the key to smarter artificial intelligence, CORDIS, cordis.europa.eu/news/rcn/126345_en.html.

Corina, D., et al. “The Linguistic Basis of Left Hemisphere Specialization.” Science, vol. 255, no. 5049, June 1992, pp. 1258–1260., doi:10.1126/science.1546327.

Donnelly, Denis P. The Computer Culture: a Symposium to Explore the Computer's Impact on Society. Fairleigh Dickinson University Press, 1985.

Dubuc, Bruno. Translated by Al Daigen, Tool Module: Chomsky’s Universal Grammar, The Brain from Top to Bottom, thebrain.mcgill.ca/flash/capsules/outil_rouge06.html.

Hatter, Lynn. “French, Spanish, German ... Java? Making Coding Count As A Foreign Language.” NPR, NPR, 1 Mar. 2016, www.npr.org/sections/ed/2016/03/01/468695376/french-spanish-german-java-making-coding-count-as-a-foreign-language.

Hook, Donald D. "Language and Linguistics." Salem Press Encyclopedia of Literature, 2017. EBSCOhost, felix.albright.edu/login?url=http://search.ebscohost.com.felix.albright.edu/login.aspx?direct=true&db=ers&AN=125599178&site=eds-live.

Jäger, Gerhard, and James Rogers. “Formal Language Theory: Refining the Chomsky Hierarchy.” *Philosophical Transactions: Biological Sciences*, vol. 367, no. 1598, 2012, pp. 1956–1970. *JSTOR*, JSTOR, www.jstor.org/stable/23250426.

Korb, Kevin B. "Turing Invents the Universal Turing Machine." Salem Press Encyclopedia, 2013. EBSCOhost, felix.albright.edu/login?url=http://search.ebscohost.com.felix.albright.edu/login.aspx?direct=true&db=ers&AN=89316433&site=eds-live

Milková, Eva. “Development of Programming Capabilities Inspired by Foreign Language Teaching.” Procedia - Social and Behavioral Sciences, vol. 171, 16 Jan. 2015, pp. 172–177., doi:10.1016/j.sbspro.2015.01.104.

Okubo, Fumiya and Takashi Yokomori. "Finite Automata with Multiset Memory: A New Characterization of Chomsky Hierarchy." Fundamenta Informaticae, vol. 138, no. 1/2, May 2015, pp. 31-44. EBSCOhost, doi:10.3233/FI-2015-1196.

O'Regan, Gerard. Giants of Computing : A Compendium of Select, Pivotal Pioneers. London : Springer, [2013], 2013. EBSCOhost, felix.albright.edu/login?url=http://search.ebscohost.com.felix.albright.edu/login.aspx?direct=true&db=cat04927a&AN=alc.447486&site=eds-live.

Postal, Leslie. “Computer Coding Instead of Foreign Language? Swap Back in Florida Senate.” OrlandoSentinel.com, Orlando Sentinel, 6 Feb. 2017, www.orlandosentinel.com/features/education/school-zone/os-computer-coding-foreign-language-florida-senate-20170206-story.html.

Reagan, Timothy. “A Sociocultural Understanding of Deafness: American Sign Language and the Culture of Deaf People.” International Journal of Intercultural Relations, vol. 19, no. 2, 1995, pp. 239–251., doi:10.1016/0147-1767(95)00007-x.

“SFPC.” School for Poetic Computation, sfpc.io/.

Tate, Allison Slater. “Should Computer Coding Be Considered A Foreign Language in School? Some Say Yes.” NBCNews.com, NBCUniversal News Group, 23 Mar. 2016, www.nbcnews.com/feature/college-game-plan/should-computer-coding-be-considered-foreign-language-school-some-say-n543656.

Turkle, Sherry, and Seymour Papert. “Epistemological Pluralism: Styles and Voices within the Computer Culture.” *Signs*, vol. 16, no. 1, 1990, pp. 128–157. *JSTOR*, JSTOR, www.jstor.org/stable/3174610.

Warner, Janis A., et al. "Learning Computerese: The Role of Second Language Learning Aptitude in Technology Acceptance." Educational and Psychological Measurement, vol. 74, no. 6, Dec. 2014, pp. 991-1017. EBSCOhost, doi:10.1177/0013164414520629.