The Imbalance in Latin American Technology Production and Non-Native Language Computer Programming

An honors thesis for the Department of Romance Languages

Adam Kercheval

Advisor: Pedro Palou, Department of Romance Languages

Tufts University, 2018

Table of Contents

1. Introduction……1
2. Cultural Background……. 3
    2.1. Internet and Technology Use in Latin America Today…….. 3
    2.2. History and Economics…….. 6
    2.3. Interpersonal Trust……. 12
3. Non-Native Language Computer Programming……... 16
    3.1. Rationale behind Investigation……..... 16
    3.2. Espy Overview and Testing Procedure…….. 21
    3.3. Notes on Data Collection……... 26
    3.4. Figures……... 28
    3.5. Results and Discussion…….. 34
4. Conclusion…… 42

Appendix
    1. Translation Tables between Python and Espy……. 45
    2. Testing Functions…….. 52
    3. Participant Data Collection Form……. 57
    4. Raw Participant Data……..... 59
    5. Participant Qualitative Feedback Transcriptions…….. 61
    6. Espy Codebase…….. 71
Bibliography……. 72

1. Introduction

For someone in the United States, technology is and has been a major part of life for almost half a century. HP, Apple, IBM, and Microsoft have been making personal computers since the 1970s. One of the first programming languages, Fortran, was written by IBM in the 1950s, and another, C, by Bell in the 1960s. Today, computers, phones, and the internet have evolved so far that it is hard to imagine a world without technology and the impact it has had on society. American technology in particular has spread worldwide: there are 4 billion internet users on earth today,[1] virtually every personal computer uses Xerox's graphical user interface layout, and 99% of all smartphones run Google's Android OS or Apple's iOS.[2]

Obviously, countries and regions other than the United States have made their own technology, with some having quite a bit of success: Huawei phones and the WeChat app are Chinese, the Spotify music streaming service is Swedish, and Samsung phones are from South Korea, to name a few. One region where global-scale technological production has historically been lacking, though, is Latin America. While companies like Lanix, Telmex, and Despegar.com have enjoyed success in the region, no Latin American technology brand today has the global name recognition enjoyed by Apple, Samsung, or Google.

In this thesis, I will first explore the reasons for this disparity in the world of technology, examining the cultural, historical, and economic reasons why Latin America is behind its peers when it comes to technological innovation. Then, I will propose and investigate the effectiveness of a programming language based in Spanish as a possible tool for helping native Spanish

[1] "Internet Usage Statistics," Internet World Stats, 31 December 2017, https://www.internetworldstats.com/stats.htm.
[2] "Global mobile OS market share in sales to end users from 1st quarter 2009 to 2nd quarter 2017", Statistica, accessed April 2, 2018, https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/.

speakers feel more comfortable entering into a programming-based discipline. In the long term, this tool would create a larger in-house workforce for the Latin American technology sector, boosting the region's technological production as a whole.

2. Cultural Background

2.1. Internet and Technology Use in Latin America Today

In order to effectively discuss the relationship between technology and Latin America as well as how to strengthen it, demographics are pivotal. Who and where the individuals who use technology in Latin America are - that is, the users of the software that I will propose - is especially important to recognize. Given that Latin America is a quite diverse region socially and economically where having access to technology can never be taken for granted, penetration rates of internet and computer usage will serve as effective indicators as to how technology, and subsequently programming and the field of software development, have impacted Latin America geographically and culturally.

Access to computers and the internet has steadily increased in Latin America over the last decade. According to the most recent poll taken by the *Comisión Económica para América Latina y el Caribe* (CEPAL) in 2016, 46.3% of homes in Latin America and the Caribbean have computers, a notable increase since 2009 when the data was first collected and that number was 27.3%.[3] CEPAL's study also indicates that 45.5% of homes in Latin America have access to the internet, up from 28.0% in 2009.[4] With both of those figures growing steadily and Latin America and the Caribbean's population around 650 million today,[5] there are at least 300 million computer users in the region, with that number sure to increase in the future.

Those figures, though, represent generalizations over the entire region - on a more specific basis, certain countries stand out as outliers in rates of computer and internet access in

[3] "Estadísticas e indicadores," Comisión Económica para América Latina y el Caribe, 2018, http://estadisticas.cepal.org/cepalstat/WEB_CEPALSTAT/estadisticasIndicadores.asp?string_busqueda=internet.
[4] Ibid.
[5] "Latin America Population," World Population Review, November 2 2017, http://worldpopulationreview.com/continents/latin-america-population/.

the home. Only five countries in Latin America and the Caribbean have in-home computer ownership rates of greater than 50%: Argentina, Brazil, Chile, Costa Rica, and Uruguay,[6] and those same five countries also have the highest rates of in-home internet access,[7] though Puerto Rico and Panama have in-home internet access rates greater than 50% as well, presumably from smartphone- as opposed to computer-based internet access.[8] Uruguay stands out as particularly technologically involved, with 69.5% of homes in the country equipped with computers, and 61.8% of homes having access to the internet,[9] which is about the level the US was at in 2011.[10] In the United States in 2015, for comparison, 78.2% of American homes had computers and 77.2% of homes had internet access.[11]

On the other end of the spectrum, some countries in Latin America are notably behind the regional average for computer and internet use. Only 13.1% of households in Nicaragua and 15.1% of households in Cuba have a computer,[12] and, respectively, only 16.2% and 7.5% of households in those countries have access to the internet.[13] This indicates that governmental and financial barriers toward accessing technology do exist in Latin America, and that nations with less economic success, like Guatemala, Honduras, and Nicaragua,[14] and nations like Cuba with a

[6] "Estadísticas e indicadores," Comisión Económica para América Latina y el Caribe, 2018, http://estadisticas.cepal.org/cepalstat/WEB_CEPALSTAT/estadisticasIndicadores.asp?string_busqueda=internet.
[7] Ibid.
[8] Ibid.
[9] Ibid.
[10] "Internet/Broadband Factsheet," Pew Research Center Internet and Technology, February 5 2018, http://www.pewinternet.org/fact-sheet/internet-broadband/.
[11] Camille Ryan and Jamie M. Lewis, "Computer and Internet Use in the United States: 2015," *American Community Survey Reports*, September 2017, https://www.census.gov/content/dam/Census/library/publications/2017/acs/acs-37.pdf.
[12] "Estadísticas e indicadores," Comisión Económica para América Latina y el Caribe, 2018, http://estadisticas.cepal.org/cepalstat/WEB_CEPALSTAT/estadisticasIndicadores.asp?string_busqueda=internet.
[13] Ibid.
[14] Ibid., with respective in-home computer ownership rates of 23.4%, 23.5%, and 13.1%.

history of restricting individual access to technology[15] may be slower than other Latin American nations when it comes to adopting and creating their own technology. That said, in-home computer and internet access is not the only way to interact with technology, and CEPAL's data on internet use in general shows that, especially in poorer countries in Latin America, levels of in-home internet access are not necessarily indicative of internet access as a whole. In El Salvador, for example, only 15% of homes had access to the internet in 2015, but in the same year, 27.5% of people indicated that they had used the internet in one way or another,[16] and Honduras, likewise, indicated in 2015 that 22.8% of homes had internet access, but 52.8% of people had used the internet in one way or another.[17]

While this means that certain parts of Latin America may be the first to benefit from new technology simply because they have higher penetration rates of personal computer and internet use than their counterparts, CEPAL's data also indicates that computer and internet use is growing in each and every country it examined.[18] In other words, while some countries in Latin America have been delayed in their processes of integration of technology into citizens' everyday lives, no country is stagnant in that regard, and the need for technology in the region will only continue to grow.

[15] Emilio San Pedro, "Cuba internet access still severely restricted," *BBC*, March 21 2016, http://www.bbc.com/news/world-latin-america-35865283.
[16] "Estadísticas e indicadores," Comisión Económica para América Latina y el Caribe, 2018, http://estadisticas.cepal.org/cepalstat/WEB_CEPALSTAT/estadisticasIndicadores.asp?string_busqueda=internet.
[17] Ibid.
[18] Ibid.

2.2. History and Economics

As the previous section indicates, though it is true that technology like computers and the internet are used today in Latin America by a significant, and in some places quite high, portion of the population, global-scale technological innovation is still lacking in the region. Consider the field of video game design, for example: as Phillip Penix-Tadsen explains in his essay "Latin American Game Design and the Narrative Tradition," game design on a global production scale (as opposed to small, independent games) was not present in Latin America or for Latin American markets until the 2000s.[19] In fact, the first large-scale, multi-console game to be designed and built entirely in Latin America, called *Lucha Libre AAA: Héroes del Ring,* was not made until 2010.[20] Of course, game design only represents a small part of the overarching field of technological innovation, but the information that Penix-Tadsen presents holds true for much of the world of technology in Latin America: while Latin America and its inhabitants may use technology, few tech-based products designed for a global consumer audience have yet to come from the region.

One of the reasons that this may be the case are the lingering effects of neoliberalism and their impact on Latin American culture. Technology, in the all-encompassing way it is thought about today, did not come into conversation until the mid-to-late 20th century, which is precisely the time the United States and other Western nations began neoliberal campaigns in the economic and political systems of many of Latin American countries. R. A. Dello Buono, in "Technology and Development in Latin America: Urgent Challenges for the 21st Century," presents an argument that, in the age of neoliberalism, "Washington needed to ensure its access

[19] Phillip Penix-Tadsen, "Latin American Game Design and the Narrative Tradition," (2015): 205-230.
[20] Ibid., 211.

to cheap raw materials and natural resources from the Americas, particularly energy resources, while taking full advantage over its higher level of technological development relative to the region."[21] In other words, the United States realized that, in order to maintain the boom of technological production it was undergoing at the time, it needed a significant amount of natural resources, which were plentiful in Latin America. This easy access to resources that Latin American territory offered was a blessing to technology companies in the United States, but those companies also realized that its availability was contingent on there not being any other competition for it. The United States, therefore, was incentivized to purposely inhibit any technological production in Latin America, which they did by supplying all the technology the region needed directly to the individuals there: "As early as the 1960s, however, it became abundantly clear that modernization approaches served mostly to reinforce dependencies upon the more developed countries. . . . Modernization essentially amounted to the 'development of underdevelopment.'"[22] This "development of underdevelopment" was fairly successful in Latin America, and it instilled in the region a dependence on technology from the United States, as well as an aversion, via cultural shifts as well as economic overpowering, toward the act of being technologically innovative and self-sufficient.

It is important to note that, in reality, Latin America had been working on its own technological advances for some time before the neoliberal era when the United States entered and intentionally stopped them. The *Fábrica Argentina de Aviones,* for example, Argentina's main producer of commercial and military aircraft, was founded in 1927 and produced more than

[21] R. A. Dello. Buono, "Technology and Development in Latin America: Urgent Challenges for the 21st Century." Perspectives On Global Development & Technology 11, no. 3 (June 2012): 341-351. Academic Search Premier, EBSCOhost (accessed April 2, 2018), 344.
[22] Ibid., 342.

a dozen different types of aircraft before it was acquired by Lockheed Martin, an American company, in 1990.[23] This acquisition was a very intentional one, and, as described in the documentary *La argentina latente*,[24] it resulted in *Fábrica Argentina de Aviones*'s almost total halt of productivity. As Dello Buono puts it, "Despite the respectable level of industrialization that Argentina had accumulated over the course of 20th century developmentalism, it proved to be no match for the disintegrative effect of 'strong' neoliberalism . . . The resulting damages wrought by the [1998-2002 Argentine economic] crisis managed to set that highly developed South American country back at least a couple of decades and this was particularly observable in the areas of science and technology."[25] Control of *Fábrica Argentina de Aviones* was transferred from Lockheed Martin back to the Argentine government in 2009, but the acquisition's effects are, unfortunately, still easily observed: it's not hard at all to find an American plane in Argentine air, but to find an Argentine one in American air is almost impossible.

A particularly interesting case to look at with regard to technological development is Cuba. American neoliberalism did not reach Cuba with anywhere near the strength that it hit other Latin American countries, and as a result there was a strong push in Cuban society to engineer everything, from technology to medicine, internally. As Dello Buono explains, instead of using American technology, Cubans created what were known as "Scientific Poles," where research and production of new technologies were carried out entirely using Cuban labor and resources. "The centers were specifically designed to enhance the creation and development of [Cuban] technologies rather than being consumers and distributors of imported technologies. It

---

[23] "Nuestra Empresa," FAdeA - Fábrica Argentina de Aviones "Brig. San Martín" S. A., accessed April 2, 2018, https://www.fadeasa.com.ar/fadea/?page_id=212.

[24] *Argentina latente*, directed by Fernando E. Solanas (2007; Argentina; Cinesur), Web (https://www.youtube.com/watch?v=lXxDMzjVPng).

[25] Dello Buono, "Technology and Development in Latin America: Urgent Challenges for the 21st Century," 345.

was precisely this approach that led to spectacular results in the creation of new products, including medications, vaccines, sophisticated cosmetics, and even medical equipment."[26] While an isolationist approach to scientific development harbors its own imperfections, the results of Cuba's scientific ideology are hard to deny. The Cuban medical system, for example, to this day has a good reputation: despite spending almost a third of what the United States spends per capita on healthcare, the life expectancy for a Cuban is almost exactly that of an American,[27] and the rate of death between the ages of 15 and 60 per capita is lower in Cuba than it is in the United States.[28]

For countries other than Cuba, though, the effects of neoliberalism's mandated technological monopoly were so strong and long-lasting that they continue to alter the Latin American outlook on technology to this day, culturally as well as economically. A certain fear of or apprehension toward interaction with technology permeates Latin American culture, as explained by Silvia G. Kurlat Ares in "A Farewell to the Future": "Recent polls show that scientific and technological activity is considered by large segments of the Argentine population as crucial for national development. Yet, the same groups know or understand very little about scientific themes, about the most basic scientific questions, or about research methodology."[29] In other words, science and technology in countries affected by neoliberalism was so dominated by American markets for so long that today, while science and technology may be considered

[26] Ibid., 348.
[27] "Cuba," World Health Organization, accessed April 2, 2018, http://www.who.int/countries/cub/en/.
"United States of America," World Health Organization, accessed April 2, 2018, http://www.who.int/countries/usa/en/
Statistics: 77 for men/81 for women in Cuba, 77 for men/82 for women in US
[28] Ibid., Statistics: 113 for men/70 for women in Cuba, 128 for men/77 for women in US
[29] Silvia G. Kurlat Ares, "A Farewell to the Future," *Technology, Literature, and Digital Culture in Latin America : Mediatized Sensibilities in a Globalized Era*. Routledge Interdisciplinary Perspectives on Literature, (2016): 62.

important by people in Latin America, it is not a field about which those people themselves feel comfortable or educated enough to take part in personally.

This distance from technology within Latin American culture is visible in the region's art, as well. Consider, for example, what Argentinian writer Ernesto Sábato wrote in his book of essays *Uno y el universo* [*One and the universe*] in 1945: "El poder de la ciencia se adquiere gracias a una especie de pacto con el diablo: a costa de una progresiva evanescencia del mundo cotidiano. Llega a ser monarca, pero, cuando lo logra su reino es apenas un reino de fantasmas" ["The power of science is acquired thanks to a pact with the devil: the cost is a gradual evanescence of the everyday world. Science becomes king, but when it does, its kingdom is just a kingdom of ghosts"].[30] *Uno y el universo* was written far before the era of neoliberalism in Latin America, but that mentality, one which eschews technology as some sort of unknown, distant power, has since engrained itself in some Latin American cultures. As Ares confirms,

> With the advent of neoliberal policies over the last twenty years, what had been a growing fear and deep mistrust [of technology] became not only the common sense of the cultural field, but also the ideological backbone of lettered discourses on scientific and technological production. Moreover, this dread extended to the individual's ability to produce practical knowledge, so scientists, engineers, inventors, small industry owners, elementary teachers and sometimes even blue collar workers all became part of the reviled petit bourgeoisie, whose pedestrian goals and way of life [were] seen as an obstacle for national development: there was nothing grandiose or transcendent about activities so deeply anchored in the material world.[31]

It's clear, then, that the way in which Latin America lags behind other regions technologically speaking is more complicated than just funding and geography - the ambivalence and hesitance

[30] Ernesto Sábato, *Uno y el universo* (Barcelona: Editorial Seix Barral, S. A., 1945), 12.
[31] Ares, "A Farewell to the Future," 62.

felt in many parts of Latin America toward technological advancement can be observed in everything from its economic system to its literature.

Following Cuba's example, one way to rapidly and effectively improve the social outlook surrounding technology and simultaneously increase its production is to create a system that encourages people and companies in Latin America to create their own tools entirely from scratch. In order to allow this to happen, then, it's necessary that the tools and processes used to build technology in Latin America are comfortable, accessible, and, most importantly, geared directly toward Latin America and its inhabitants. It's unreasonable to expect non-English speaking people in Latin America who want to learn to code to be able to dive, frustration-free, into a world of programming languages where everything from the physical keywords in the code, like "for," "while," and "return," to the extensive documentation of code and coding tools, requires a substantial knowledge of English.

For this reason, I'd like to examine the potential impact of a programming language that is based entirely in Spanish, so that, at the very least, any Spanish speaker who would like to learn to code need not have any understanding of English to get involved in the world of technological innovation.

2.3. Interpersonal Trust

Another aspect of Latin American culture to analyze when discussing the region's relationship with technology is its social setup. What is specifically notable is which sources of information and power are considered reliable by Latin Americans, which are considered untrustworthy, and how that relates to technology in the region. One helpful, albeit unfortunate source for gauging this kind of relationship with power is the seismic activity in Mexico in the last century. In 1985, Mexico had a famously devastating earthquake that killed thousands throughout the country and all but destroyed the nation's capital. In the aftermath of this earthquake, Mexico's main "vertical" source of influence and power, the government, was notoriously inept at handling the situation. Reports from foreign news sources as well as interviews from Mexicans reflect the frustration that individuals felt regarding the bureaucratic emergency management protocols in place and the lack of progress that was made to help rebuild Mexico City. Eventually, this frustration reached a boiling point, and civilians began taking matters into their own hands. As the *Wall Street Journal* explained in 1985,

> Nevertheless, the failure of the system this time has wrought the unthinkable: People are fending for themselves, outside the system -- behavior that is anathema to the Mexican ruling elite and its machine, the PRI. For Mexico's president, Miguel de la Madrid, what began as a natural disaster has become a political turning point . . . All over the city, independent groups have sprung up, feeding victims, dispensing medicine -- and either rejecting government help altogether or making sure it goes through private hands.[32]

[32] Steve Frazier and Mary Williams Walsh, "Mexico -- Borrowed Time --- Nation in Jeopardy: Earthquake Aftermath Points Up Weaknesses of Mexican Leadership --- Citizens Scorn Ruling Party for Inaction, Inefficiency; Big Boost for Dissidents --- Rebuffing the Yankee Giant," *Wall Street Journal*, Eastern edition (New York, NY), Oct. 15, 1985.

The government's attempt to assist the people of Mexico was met with disdain and frustration, shedding light on just how much people in Mexico did not trust their government and would rather work with each other in times of struggle, instead.

This type of behavior is indicative of a very "horizontal" trust structure, where people trust one another more than they trust powerful groups. In "Income inequality, distributive fairness and political trust in Latin America," a paper by Sonja Zmerli and Juan Carlos Castillo that explores the way people in Latin America think about their social structure, income, and politics, a group of Latin Americans from various countries was polled about how much they trust their political system. Specifically, they were asked to place the amount of trust they had for their country's government, national congress, judiciary, and political parties on a scale of 1 to 4, where 1 is no confidence, 2 is little confidence, 3 is some confidence, and 4 is a lot of confidence. The average response was 2.09.[33] A handful of countries (Mexico, Peru, Dominican Republic, Guatemala, and Honduras) had average responses below 2, and Uruguay's average trust level, which was the highest out of any Latin American country, only reached 2.5.[34] In other words, Zmerli and Castillo's research indicates that, broadly speaking, people in Latin America have quite limited trust for anything that they think has to do with their government or people otherwise in power.

As another, perhaps more relevant example, Latin America's trust layout can be observed in the online community. Of all mainstream social media sites, Twitter's structure is the most horizontal, in that, essentially, every account has the same amount of power. A user sees only the accounts that they follow, and a tweet will rarely show up on someone's timeline simply because

[33] Sonja Zmerli and Juan Carlos Castillo, "Income inequality, distributive fairness and political trust in Latin America," *Social Science Research*, (July 2015).
[34] Ibid.

Twitter thinks it is important to see. In other words, a private citizen has just as much (or as little) publishing power on Twitter as, for example, CNN does: each account's tweets will be read only be people who want to read them, and if a Twitter user wanted to pretend that CNN didn't exist and instead get their news entirely from their peers on Twitter, it would be easy to do. It should come as no surprise, then, that Latin America has a notably high rate of Twitter users, at close to 12% of the entire population.[35] This is made more impressive by the fact that only about 50% of people in Latin America are internet users at all,[36] which means that almost one out of every four people on the internet in Latin America is an active Twitter user. Comparing that to the United States' 286 million internet users,[37] 69 million of whom use Twitter[38] (around 25% of all internet users and 20% of the nation's population as a whole), it's clear that Twitter has caught on among internet users in Latin America just as powerfully as it has in the United States, and that it is more than just a casual social media site in the region.

All this data indicates that, for a Latin American citizen, trust is a rare commodity, and trust in something that doesn't come directly from the people (as opposed to the elites and the government) is especially infrequent. With that in mind, it's clear that one of the best ways to help Latin America embrace and work with technology on a large scale is to make it feel more proprietary and personal. Imagine having to convince a Mexican journalist, in the wake of

---

[35] Matteo Ceurvels, "Twitter's Riding a Political Wave in Latin America," eMarketer, August 15, 2017, https://www.emarketer.com/Article/Twitters-Riding-Political-Wave-Latin-America/1016334.
[36] "Estadísticas e indicadores," Comisión Económica para América Latina y el Caribe, 2018, http://estadisticas.cepal.org/cepalstat/WEB_CEPALSTAT/estadisticasIndicadores.asp?string_busqueda=internet.
[37] "United States Internet Users," Internet Live Stats, accessed April 2, 2018, http://www.internetlivestats.com/internet-users/us/.
[38] "Number of monthly active Twitter users in the United States from 1st quarter 2010 to 4th quarter 2017 (in millions)," Statistica, accessed April 2, 2018, https://www.statista.com/statistics/274564/monthly-active-twitter-users-in-the-united-states/.

mid-2017's huge spyware problem,[39] to buy a new phone, where one option is a phone manufactured internationally in various countries and imported into Mexico, and the other is a phone made from the ground up by a Mexican company. Surely the latter would be easier to sell to a security-minded journalist: it's a local product - a Mexican phone built by Mexicans is less likely to have anti-Mexican spyware installed on it.[40]

While domestic manufacturing of technology products surely will not change Latin America's trust structure overnight, I do believe that it will help mitigate the fear and apprehension that some Latin Americans feel toward technology, allowing them to think of it as something that is their own and therefore more trustworthy. It is precisely for this reason that creating technology-building infrastructure is so important for Latin America. If users in the region are expected to embrace technology, one of the best ways to make that happen is to allow technology to be created in Latin America from the ground up - and the ground, in this case, is programming languages, which are at the core of any technology product. To have a proprietary programming language for Spanish speakers would mean having more Spanish speakers writing proprietary code, which would in turn result in more home-grown technology that Latin American users would trust, use, and embrace. In the next section, I will explore the fundamentals of how that first step, creating Spanish-based code, works and feels.

[39]Azam Ahmed and Nicole Perlroth, "'Our Phones Are Being Monitored': How a Hacking Story Unfurled," *New York Times*, (New York, NY), June 19, 2017.

[40] One could argue that this is not entirely true - it's possible that the hypothetical Mexican phone company could have an agreement with the Mexican government to allow a back door or something of the sort to their devices, which would make the international phone in reality a safer purchase. But domestic products undeniably have a certain appeal to consumers: do you think the average US citizen would rather buy a Chinese phone built by a Chinese company for Americans, or an American phone built by an American company for Americans?

3. Non-Native Language Computer Programming

3.1. Rationale behind Investigation

One large problem that needs to be overcome in the process of creating a more productive relationship between Spanish speakers and technology is the language gap at the lowest level of technological production. In order to build their own technology, people in Latin America need to learn how to write code. Today, there is no widely used programming language that is not in English, which means that in order for Spanish speakers to learn to write code, they need to either know English or be prepared to memorize many keywords which to them, at best, will be cognates to Spanish words, and at worst will seem like gibberish. This is a serious issue, and the fact that code, which is not very accessible at first even to English speakers, seems even less approachable to people who don't speak English as their first language, may make the idea of becoming a software developer seem exhausting and time-consuming.

The Python language, for example, is one that many English speakers may find to be readable at first glance. It uses many physical words to represent computational and logical ideas, which makes reading the code fairly easy, and it also makes writing the code seem more natural to the user than in other programming languages. This, combined with the fact that Python has an extensive online community, the fact that it's not a syntactically strict language (brackets, semicolons, and parentheses are either not necessary to use or not used at all), and the fact that it is a relatively high-level language, meaning a lot of the obscure and complex functionality of the language and the machine it is running on are hidden from the user, make it a powerful yet relatively welcoming language to use to learn programming.

The issue, though, is that Python's only language is English. The keyword `include`, for example, is only that: there is no `incluir` or `inclure` or `शामिल`. So for a non-English speaker, one of the main draws of Python, its readability, is essentially lost. Consider, for example, a simple function written in a made-up Russian version of Python, which, to anyone who doesn't speak Russian, might seem particularly hard to decipher:

```
опр someFunction(someParameter):
    если длина(someParameter) не является 1:
        вернуть someFunction(someParameter[1:]) + someParameter[0]
    еще:
        вернуть someParameter
```

Even if I were to reveal that the function takes a string and reverses it, that doesn't make it much easier to determine what happens on each line. And, surely, the idea of writing, debugging, and eventually making a profession out of code, all written in Russian, does not appear to be a fun or easy activity, especially for someone who is not willing to learn Russian or memorize dozens of keywords that mean nothing to them. In normal Python, however, to an English speaker, the exact same function looks a lot less intimidating:

```
def someFunction(someParameter):
    if len(someParameter) is not 1:
        return someFunction(someParameter[1:]) + someParameter[0]
    else:
        return someParameter
```

The importance of the spoken, human language that a programming language is written in, therefore, should not be overlooked, especially when it comes to making code look more accessible and welcoming at first glance.

It is also worth understanding that the coding process is more than just the physical code: warnings, error messages, and other feedback from the coding environment are huge components of understanding code and figuring out why it does or does not work. Imagine, again, the first day of class in Russian coding school, and after creating a list and attempting to access the first element of it, the code fails and prints out this error to indicate what's wrong:

```
Ошибкаиндекса: индекс индекса вне диапазона
```

It's no use at all - you might as well have not read it in the first place. The problem is that, in reality, this is an incredibly common error, for new and veteran programmers alike:

```
IndexError: list index out of range
```

Seeing it usually indicates a fairly simple problem to solve that occurs when attempting to access a list item at an index that does not exist. If Python's error messages were in Russian, though, any user who does not speak Russian would not know what to begin looking for or even conceptualizing as the source of the problem.

I should also mention, especially in this case, that the idea of just memorizing foreign words and what they mean when it comes to code errors is impossible in the long run. Errors from a language like Python never tell the user what the exact solution is to their problem, but instead serve more as hints to where to begin when solving the problem. As code gets more complex, so do error messages. Building an app, a website, or really any sort of large-scale programming product where error and debugging information is in a language that the programmer does not understand would not be feasible.

It's arguable, then, that one might suggest that, if English-based keywords are the problem, the solution is to promote international coding in programming languages that have a

nonexistent English vocabulary, or one that is at least smaller than Python's. This is a valid point, and such languages do exist,[41] but it creates two new problems. The first of these problems is that code without keywords still is neither readable nor easy to learn for the novice programmer. Here is the string reversal function discussed above written in a programming language called Haskell, with no English keywords (except for `Char`, for Character, whose existence in this function is unavoidable):

```
someFunction :: [Char] -> [Char]
someFunction (x:[]) = [x]
someFunction (x:xs) = someFunction xs ++ [x]
```

This function is, on a line-by-line basis, exactly the same as the Python function above, and even though a plus sign is a plus sign in Spanish or in English, all that the removal of English keywords from the code has done is make it as difficult to read for an English speaker as it is for a Spanish speaker.

The second issue that non-word-based code creates is that, to put it simply, languages like Haskell are not very popular in the world of software development. Being taught to code using Haskell could hypothetically allow someone to learn how programs work and what the core concepts of computer science are, but to attempt to enter the job market for software developers knowing nothing but languages with few keywords like Haskell, Scheme, or even C would limit the opportunities and growth available to any prospective employee.

For a non-English speaker, the ability to write code that is readable and understandable to them and to those who speak their language opens up a new world of opportunity for software development. With translation software that can convert, for example, Python in Spanish to

[41] There are plenty of programming languages with a smaller English vocabulary that Python's, but there are actually not many at all that have no keywords whatsoever. Even assembly language, the closest a programmer can get to the actual zeroes and ones of a computer, has keywords like `add`, `mov`, and `push`, which are clearly English.

Python in English,[42] programmers don't need to all speak the same human language to understand each other's code, and every individual can program using the language that is most comfortable to them. With this, technology will seem more proprietary, less foreign, and more welcoming to anybody who doesn't speak English.

[42] This is not that hard to do and is not a heavy program at all - as a proof of concept, consider one that I wrote in 2016 that converts back and forth between Spanish and English for C, C++, and Python: https://github.com/akercheval/si

3.2. Espy Overview and Testing Procedure

In order to compare the experiences of programming in native versus non-native languages, I translated a copy of Python into Spanish. In other words, the functionality of the language is the same as English-based Python, but the language's core function, object, and error names are all in Spanish. Here is the string reversal function, written first in English Python:

```
def reverse(word):
    if len(word) is not 1:
        return reverse(word[1:]) + word[0]
    else:
        return word
```

And in Spanish Python, which I've named "Espy":

```
def reverse(word):
    si tam(word) es no 1:
        volver reverse(word[1:]) + word[0]
    sino:
        volver word
```

In my design of Espy, I intentionally keep its objective visual properties as similar as possible to those of standard Python - that is, when deciding how to translate words, I examined whether a translation was truly necessary in the first place (the English Python word "def" can stand for both "define" and "definir," so there was no need to translate that into Spanish), as well how to make the Spanish word look (I decided to translate"len," which is short for "length," into "tam" instead of "tamaño"), with the goal of making Espy precisely as usable to a native Spanish speaker as Python is to a native English speaker.

One reality that comes with something like Espy and the process of translating Python into Spanish is the fact that a 100% translation is not possible, at least not as a single-person

project. Python is a huge language, and there are far too many English-based built-in aspects of it for one person to try to find and translate everything. This is, of course, not even considering the multitude of third-party packages that exist for Python, the vast majority of which are in English. The scope of this project, then, at least in the short term, was to translate the core elements of Python, the built-in objects, errors, and keywords, into Spanish. What this means for Espy is that its target audience is people who speak Spanish and want a more welcoming and understandable introduction to the world of coding. Though Espy could, on a technical level, be used as a full-fledged programming language, it would probably be a good idea to switch from Espy to Python after the user has a better understanding of how programming works.

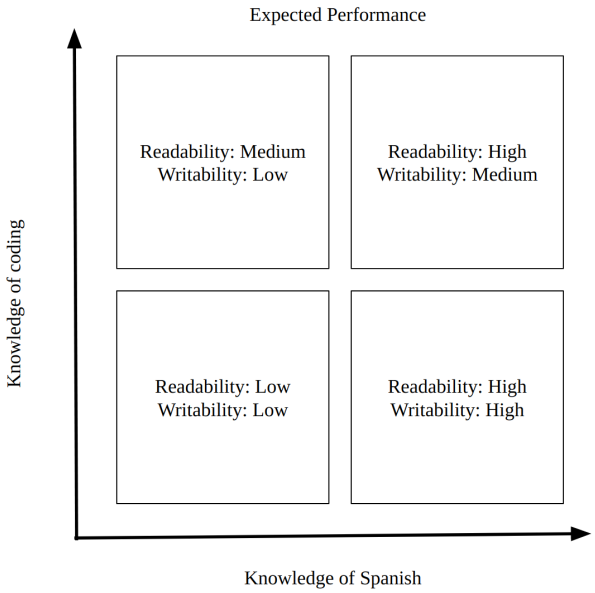
In order to evaluate the process of coding in a non-native language, I will collect both quantitative and qualitative data on how users (who will be native English speakers) interact with Espy to be able to discuss both its readability and its writability as compared to English Python. There are two main variables that I expect will affect each user's feelings toward coding in Spanish: their previous coding experience and their Spanish skill level. Before any testing is done, I will ask participants to self-rank their own skill level in those fields on a scale of 1 to 10, in order to provide context for the data gathered.

To test the readability of non-native language code, I will have participants look at two short functions of about equal complexity where one is in standard Python and the other is in Espy, and I will have them attempt to explain what the function does and predict its output given certain input. This will allow me to time each participant's response to gather objective data on how easily they understand Espy as compared to Python, and I will also ask them to "think out loud" during the process to gain some insight into their more qualitative responses to the reading

process. I anticipate that participants who have a high understanding of both Spanish and programming concepts will find the process of reading and interpreting Espy the easiest, while participants who have low levels of experience in both will find it the most challenging. For every participant, except those who understand both the coding process and Spanish, I plan to explain briefly to them the general syntax of Espy (and of Python, if need be) before showing them any code, the idea behind this being that showing someone who knows little code or little Spanish a block of Espy and asking them to interpret it would be essentially impossible for them. On the other hand, for the participants who know both Spanish as well as how to program, I'd like to see whether they can read Espy without guidance from me. I expect that they will be able to, but whatever the case, it will provide good insight as to whether programmers who learn to code on a Spanish-based programming language and then learn English, once they have decided they would like to pursue programming more seriously, will have a hard time understanding and switching to English-based programming languages.

To test the writability of non-native language code, I will have participants engage in a similar exercise to the one above, where, after explaining to them a few basic keywords in Espy (and in Python, if necessary), I will ask them to write or modify small functions of similar complexity in standard Python and in Espy. Again, I will time them and ask them for their thoughts during the process. This also will provide an opportunity to see how well participants can understand error messages in Spanish, and how debugging in their native language compares to debugging in a foreign one. Like the readability test, I expect that people who have no knowledge of Spanish or coding will find this process particularly challenging, and I expect brute-force memorization of Spanish keywords to be the only way those individuals will be able

to write code in Espy. However, unlike the readability test, I expect that participants who are experienced in programming to find this test challenging as well, regardless of their knowledge of Spanish. Those who know Spanish will obviously not have to memorize keywords as much as those who don't know Spanish, but I anticipate it will be hard either way. Speaking from my own personal experience writing code using Espy, I have found that once you begin to "think in code," you think of keywords not in terms of their dictionary definitions, but instead in terms of what job they do for the flow of the code. Replacing keywords that you have used extensively, regardless of whether or not you understand them, breaks up your thought process when you are writing code and makes you slow down significantly. I think, therefore, that the best performer in the writability test will be the participant who is skilled in Spanish but is relatively new to programming. That participant will not have any understanding of what keywords do what jobs, and so to them the keywords "while" and "mientras" are both entirely new concepts; neither means more or less to the participant than the other.



A final piece of data to keep in mind during testing is that, as of 2015, two-thirds of Spanish speakers reported being proficient in English.[43] This is promising news in terms of

[43] Jens Manuel Krogstad et al., "English Proficiency on the Rise Among Latinos," Pew Research Center Hispanic Trends, May 12, 2015, http://www.pewhispanic.org/2015/05/12/english-proficiency-on-the-rise-among-latinos/.

bringing the coding community together as a whole, because if the testing reveals that the native English-speaking participants who are proficient in Spanish can read and write Espy with relative ease, then it can be inferred that people who are native Spanish speakers but proficient in English will be able to read and write English code with similar ease. On the other hand, if it is the case that the test subjects who are proficient in Spanish still have trouble reading and writing code in Espy, it would indicate that proficiency in a non-native language is not enough when it comes to understanding code written in the language, and that fluency in a spoken language is a must before attempting to interact with code written in it.

3.3. Notes on Data Collection

The data collection process outlined above was carried out on a sample size of 10 participants, all of whom were fluent English speakers and students at Tufts University. Two participants, namely participants 2 and 4, were native Spanish speakers. As mentioned in the previous section, two variables were tested: knowledge of Spanish and knowledge of programming, and when plotting the data they were kept separate to make the individual figures less visibly complex. Obviously, though, those two variables are simultaneously at play during any interpretation of code in Spanish, and wherever possible I interpreted the data taking both variables into consideration. In some cases, though, one variable turned out to be essentially unrelated to the participants' performance (like knowledge of Spanish in error interpretation, for example), in which case the analysis focuses on just the impact of the one variable that did affect performance.

The testing process as a whole, which is available for further examination in the appendix, was intended to be more qualitative than quantitatively rigorous. Due to time and resource limitations, as well as an intent to assess the participants' baseline feeling about coding rather than test a strict numerical hypothesis, the study design was purposefully simple. That said, efforts were made to adequately randomize the testing experience: participants were pulled from a variety of fields of study at Tufts (though all participants were students, making them all more or less the same age), and the Spanish and English functions they used, as well as the order in which they interacted with them, were randomized prior to the start of testing.

Another important note about the testing process is that, due to geographic constraints, every individual tested was a fluent English speaker with varying amounts of experience in

Spanish. This, of course, is the opposite of what the actual audience of Espy or any Spanish-based programming language would be: fluent Spanish speakers with varying amounts of experience in English. The testing process as a whole, then, should be considered less of a test of Espy specifically and more of a test of programming in native versus non-native languages. In other words, the testing data should be interpreted so that, for example, if participants who are native English speakers with low Spanish experience find it easy to read and write code in Python but hard to read and write code in Espy, the opposite should hold true for native Spanish speakers. That is, native Spanish speakers with low English experience would find it easy to read and write Espy but hard to read and write Python. This is because, as visible in the first section of the appendix, Python and Espy are one-to-one translations of each other. Neither is syntactically simpler or more complex than the other; they are the exact same programming language, they are just based off of and interacted with in different spoken languages, which allows this testing process to explore the impact of non-native language programming without being partial toward the English or Spanish version of the language being tested.

3.4. Figures

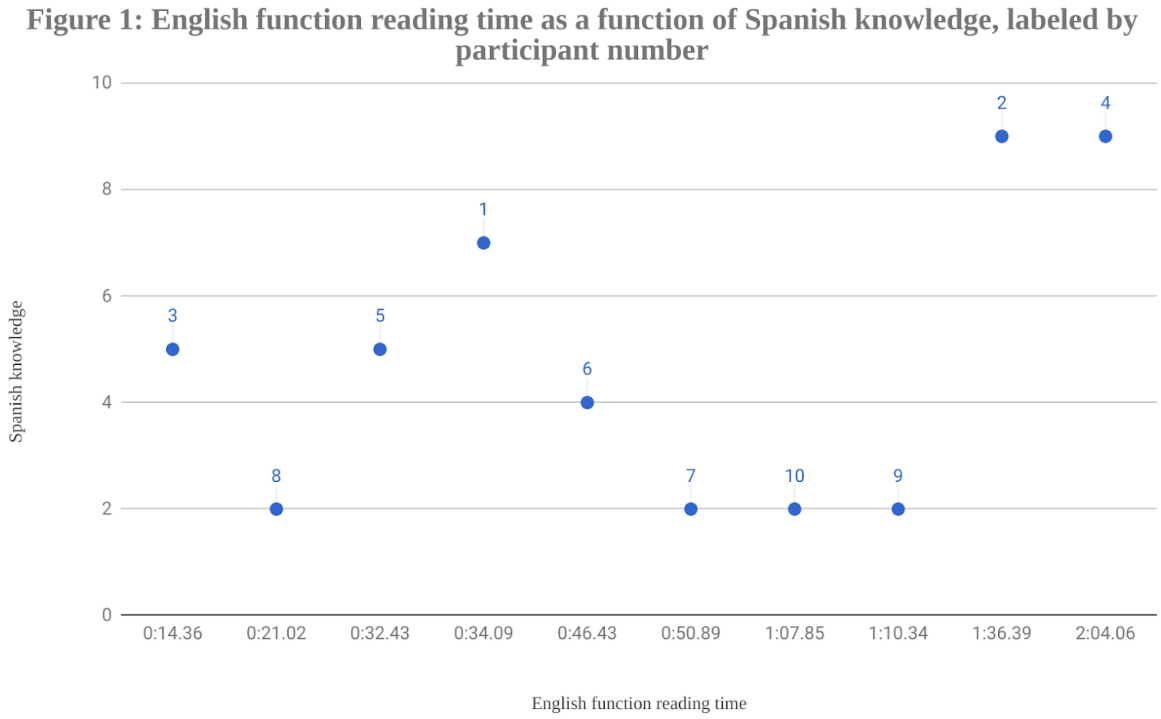


Figure 1: English function reading time as a function of Spanish knowledge, labeled by participant number
Spanish knowledge
10
8
6
4
2
0
3
8
5
1
6
7
10
9
2
4
0:14.36
0:21.02
0:32.43
0:34.09
0:46.43
0:50.89
1:07.85
1:10.34
1:36.39
2:04.06
English function reading time

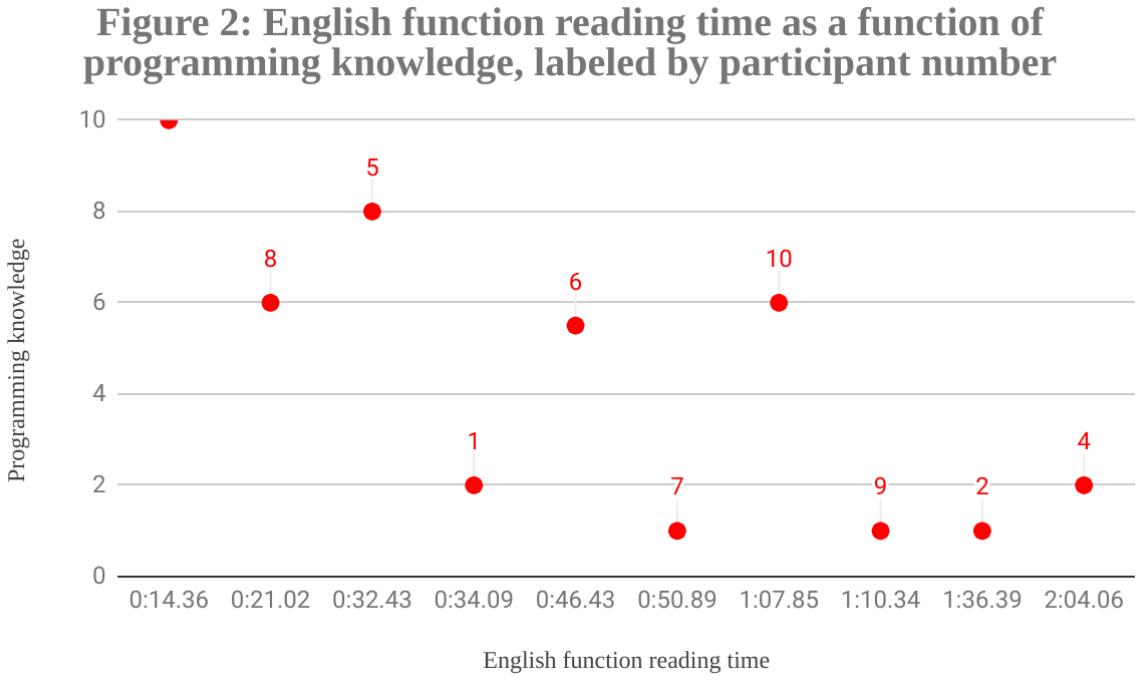


Figure 2: English function reading time as a function of programming knowledge, labeled by participant number
Programming knowledge
10
8
6
4
2
0
5
8
6
10
1
7
9
2
4
0:14.36
0:21.02
0:32.43
0:34.09
0:46.43
0:50.89
1:07.85
1:10.34
1:36.39
2:04.06
English function reading time

Figure 3: Spanish function reading time as a function of Spanish knowledge, labeled by participant number

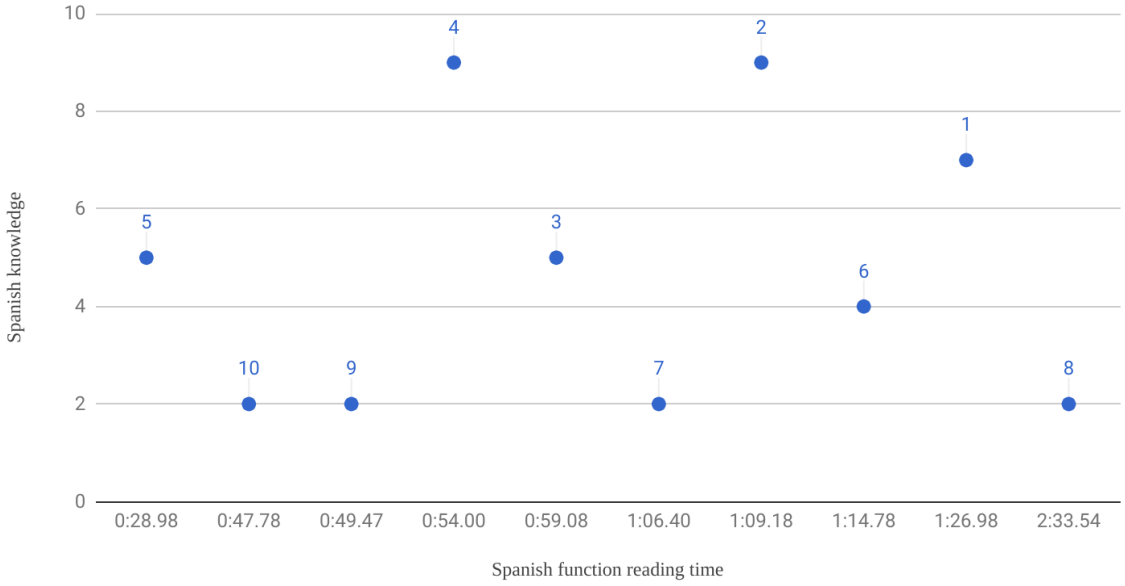


Figure 4: Spanish function reading time as a function of programming knowledge, labeled by participant number

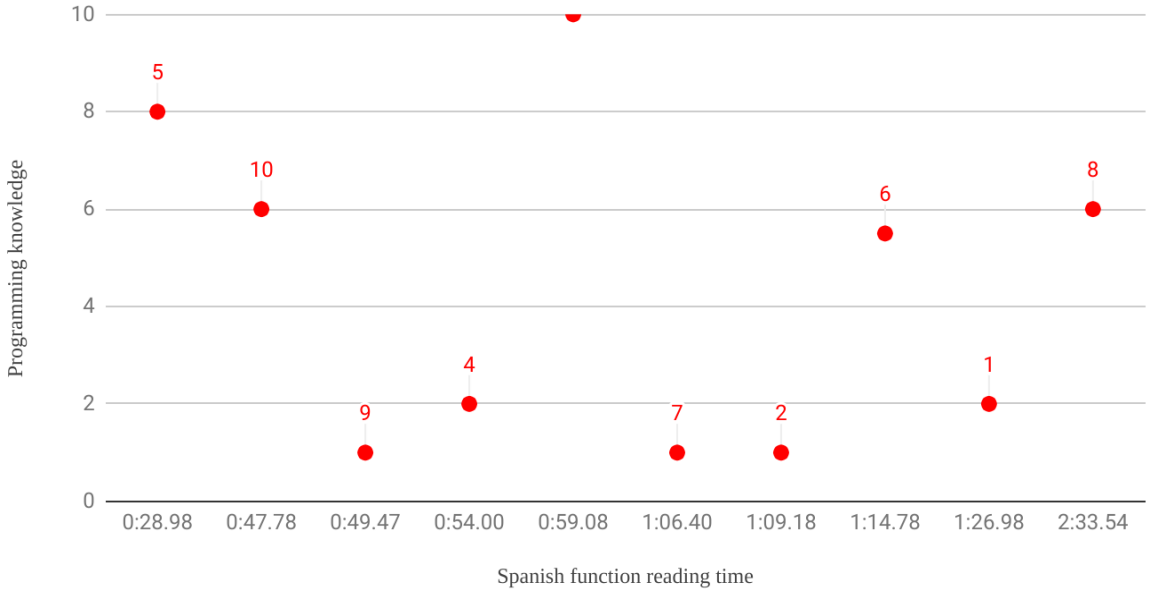


Figure 5: English function writing time as a function of Spanish knowledge, labeled by participant number

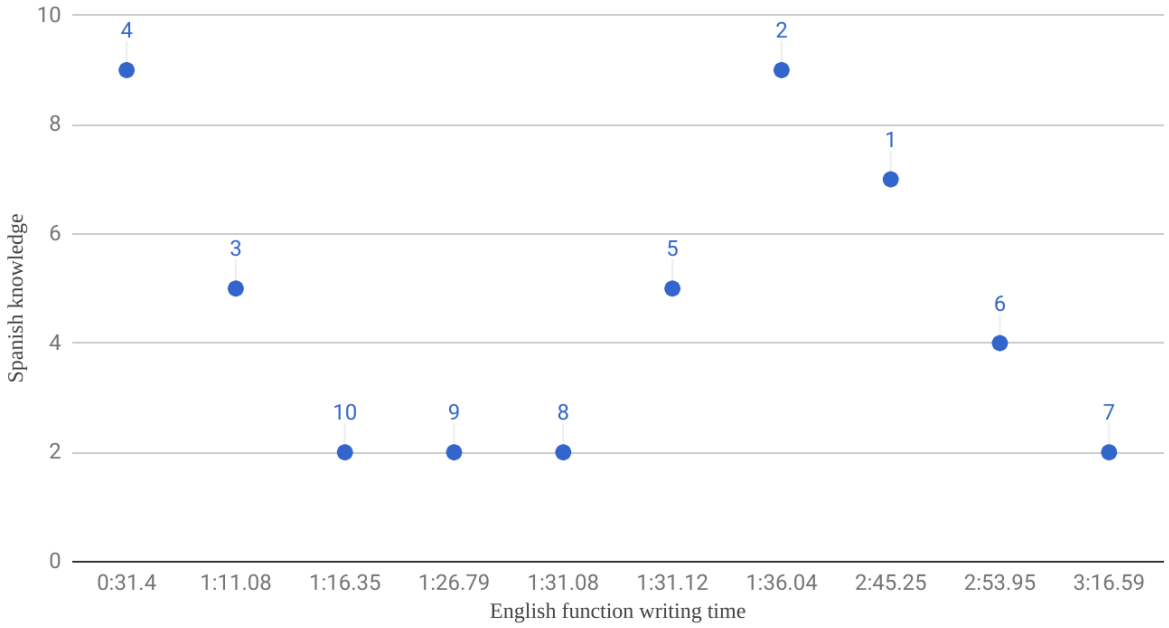


Figure 6: English function writing time as a function of programming knowledge, labeled by participant number

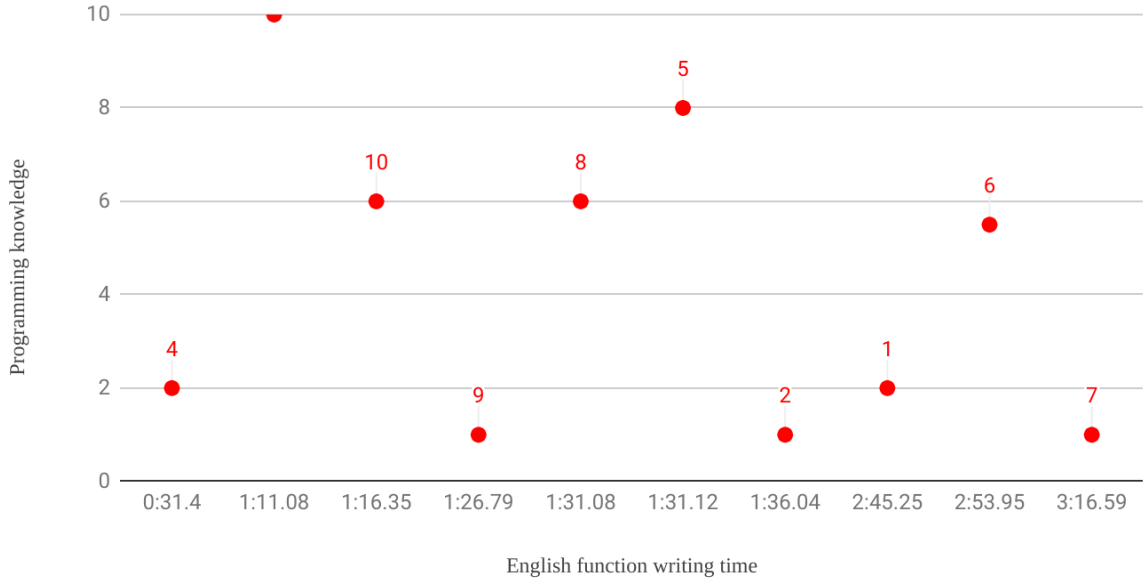


Figure 7: Spanish function writing time as a function of Spanish knowledge, labeled by participant number

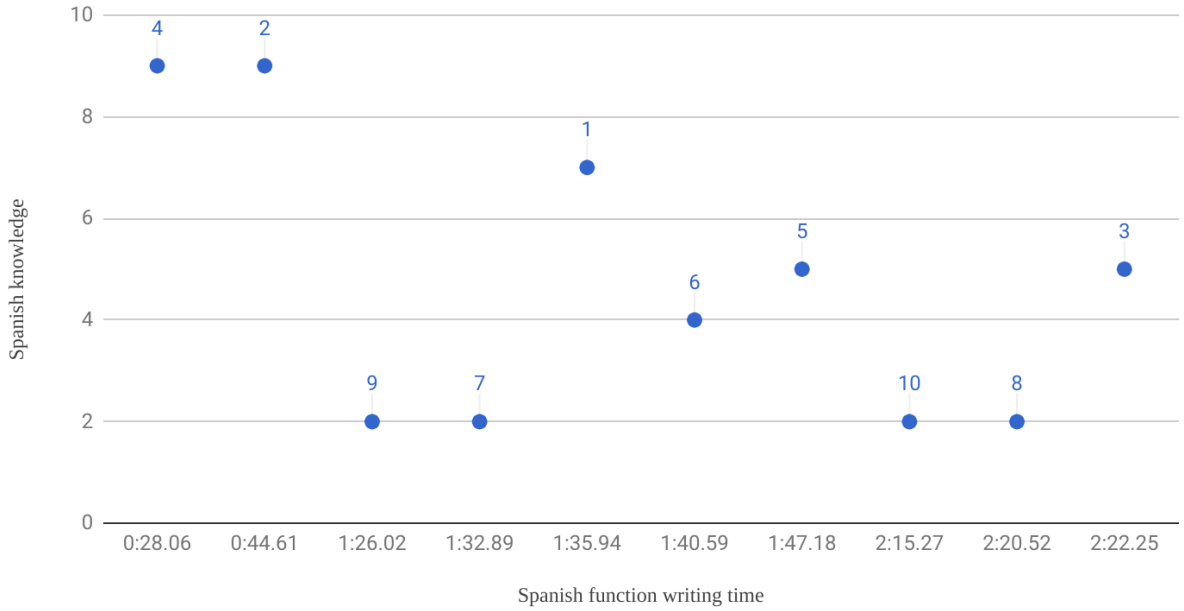


Figure 8: Spanish function writing time as a function of programming knowledge, labeled by participant number

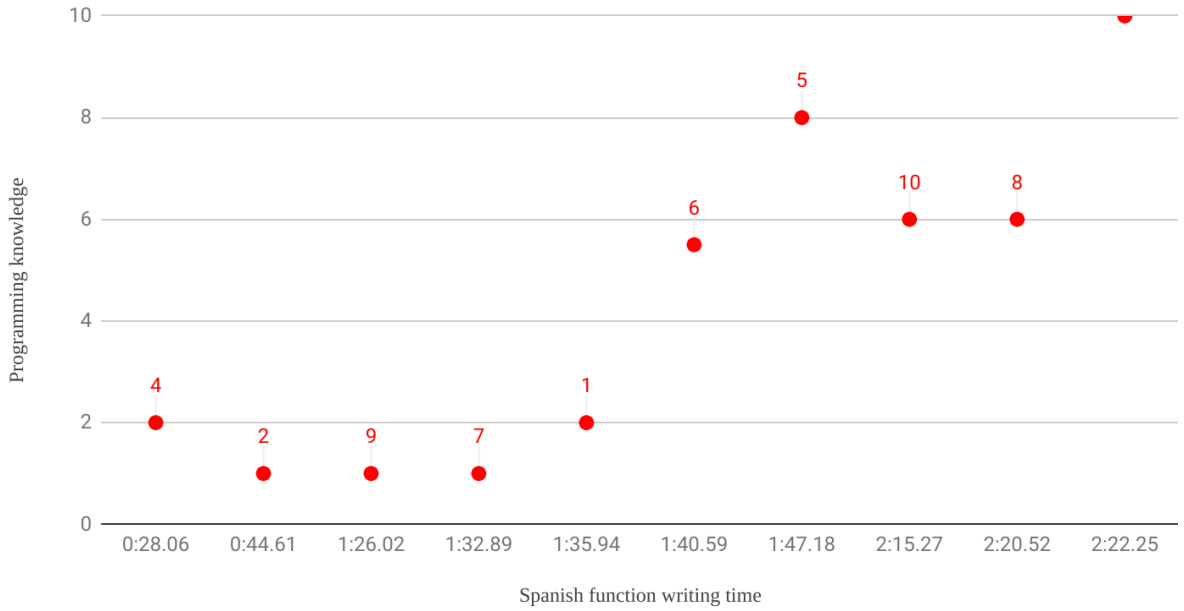


Figure 9: English error reading time as a function of Spanish knowledge, labeled by participant number

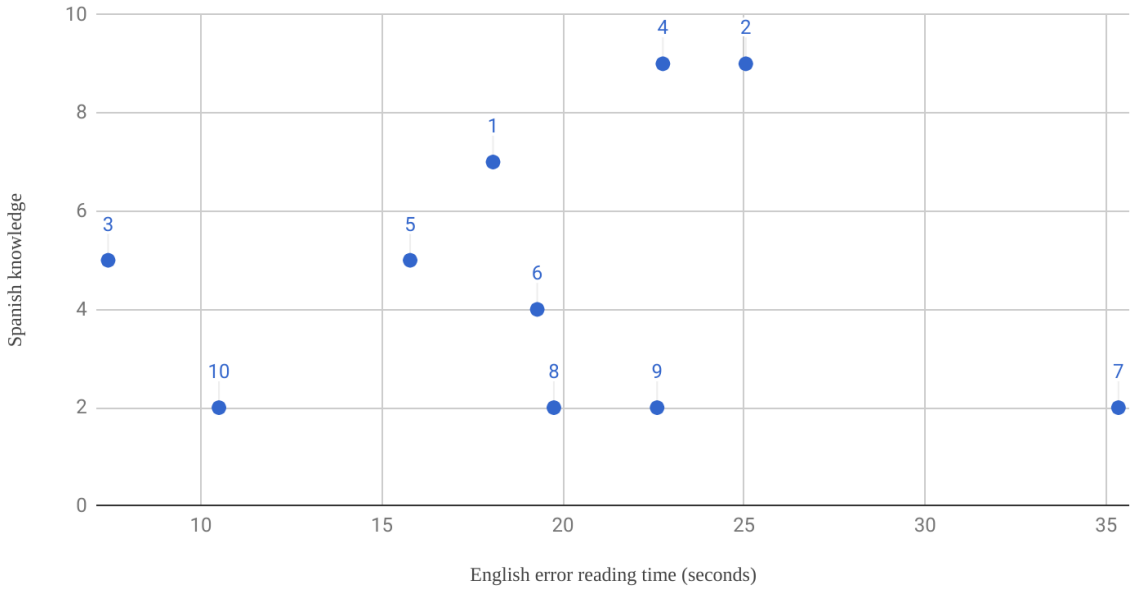


Figure 10: English error reading time as a function of programming knowledge, labeled by participant number

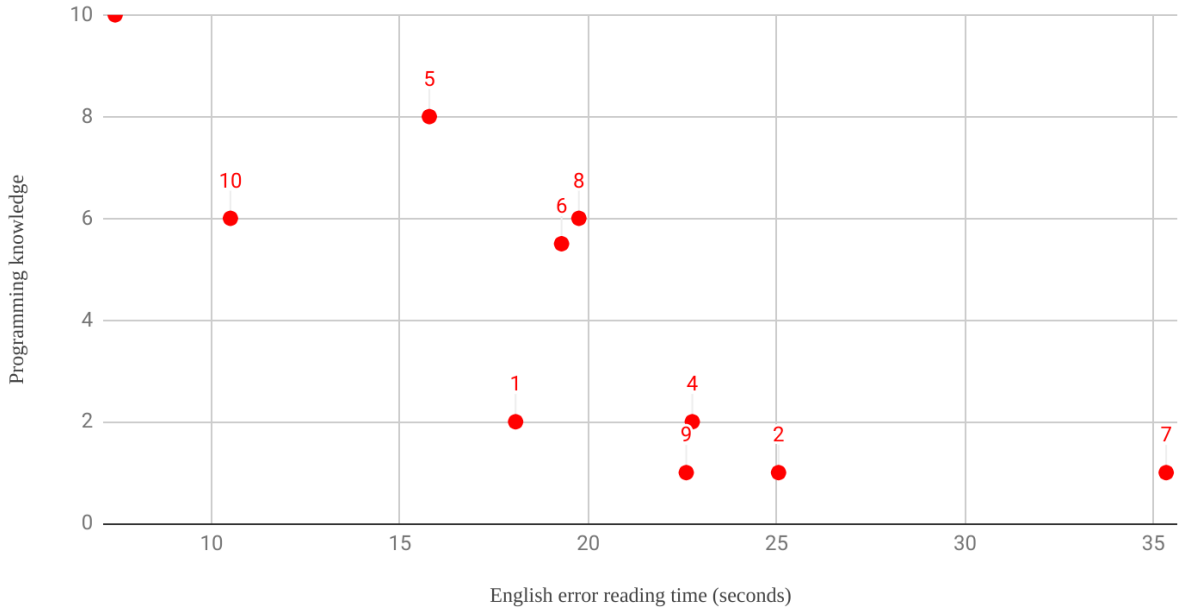


Figure 11: Spanish error reading time as a function of Spanish knowledge, labeled by participant number

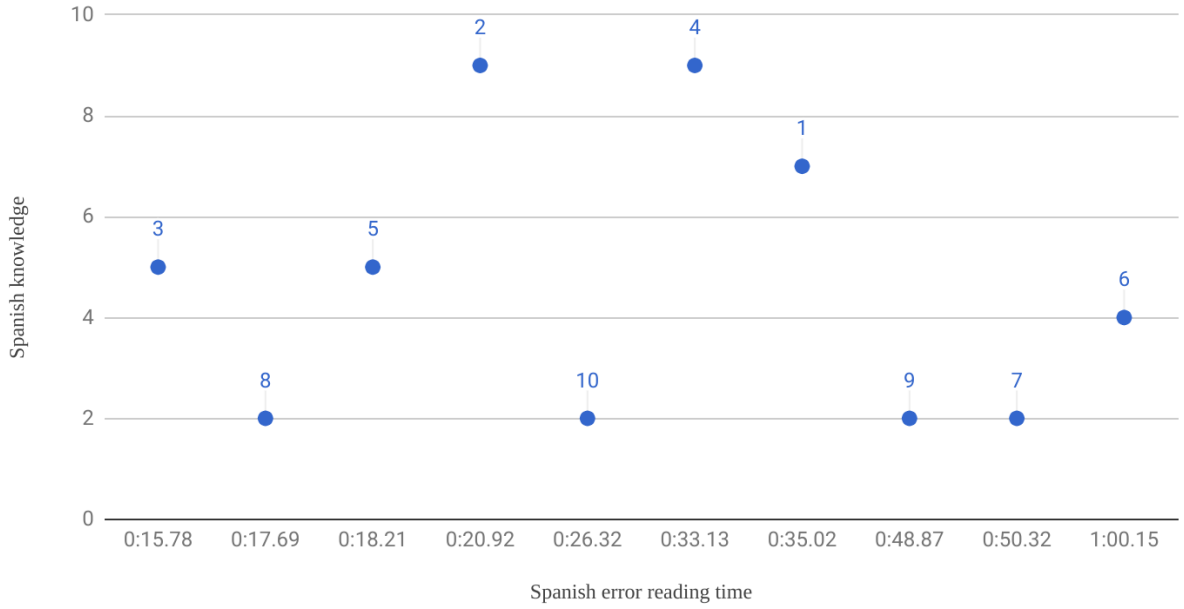
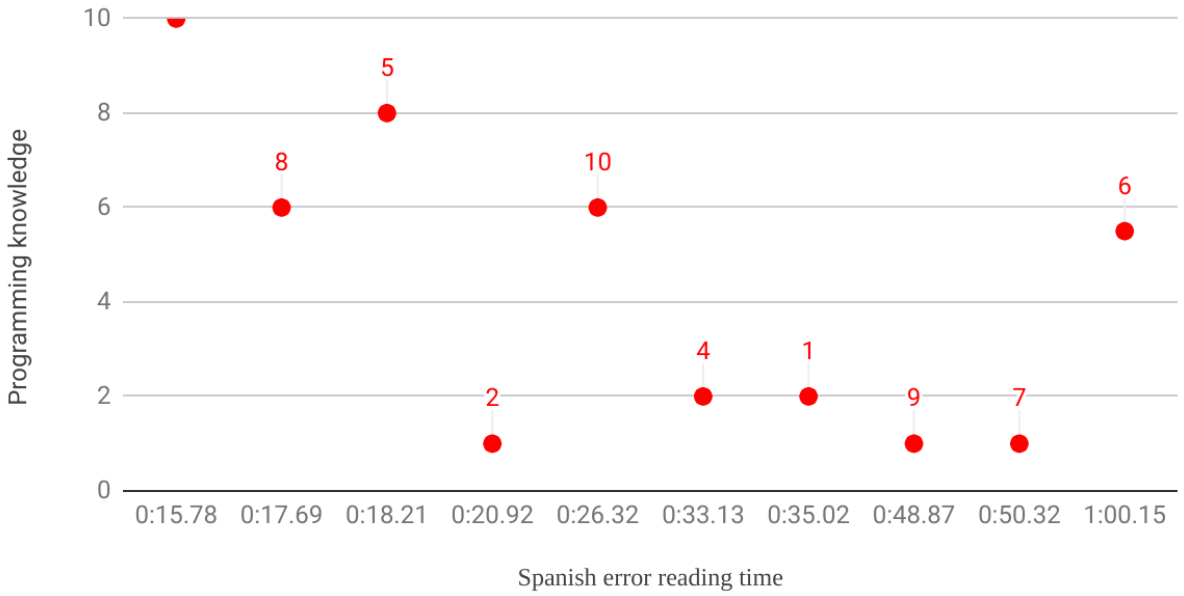


Figure 12: Spanish error reading time as a function of programming knowledge, labeled by participant number



3.5. Results and Discussion

After running the tests outlined previously on ten individuals of varied levels of Spanish and coding proficiency, the results, while not entirely what I had predicted, nonetheless revealed some interesting data.

First, I will discuss the quantitative results from the experiments: as visible in Figures 1, 2, 5, and 6, subjects who had more programming experience generally wrote and read English code quicker than those subjects who had less programming experience, regardless of Spanish experience. Likewise, Figures 9 and 10 show that participants with previous coding experience were able to read and interpret English errors generally quicker than those who had less coding experience, with Spanish experience showing no notable trends in error interpretation.

The data becomes interesting, though, upon comparing the trends in English Python with those in Espy. Figures 3 and 4 show that there were essentially no general trends in the speed with which the participants were able to read Espy code, perhaps indicating that, from a first impressions standpoint,[44] interacting with code in a non-native language is fairly difficult for everyone, regardless of coding experience or any spoken language proficiency. Writing code in Espy, though, did reveal some interesting data: as visible in Figures 7 and 8, participants with more programming knowledge and less Spanish knowledge generally took longer to write functions in Spanish, and participants with more Spanish knowledge and less programming knowledge wrote the functions quicker. The former trend could be attributed either of two reasons: the first is that the functions that subjects who were more experienced in programming

[44] I say from a first impressions standpoint because reading the code always came before writing it during the testing process - when participants read Espy, it was the first time any of them had seen it. I imagine that having them read Espy more frequently and in larger blocks might have yielded some more trends in the data, but that would have required the participants to take a fair amount of time to understand and get comfortable with Espy, which would clearly have taken too much time to do in one testing session.

were asked to modify were more difficult than the corresponding functions for their less experienced counterparts, and the second is because people who have more programming experience are not used to writing code in a language other than English and thus have to think about it for longer than they would when dealing with English code. The qualitative responses further along in this section will help to confirm that theory.

In all, though, the key element revealed through this quantitative data is that people with the most knowledge of Spanish and the least knowledge of programming tended to have the least trouble writing code in Spanish. In fact, the two participants who wrote code in Espy the quickest (see Figures 7 and 8) both had high Spanish knowledge but low programming knowledge. The next two fastest participants had low Spanish knowledge as well as low programming knowledge, and they were all still faster than the rest of the participants, who had higher knowledge of both Spanish and programming. So when it comes to writing code in a non-native language, the data indicates that those who have the easiest time with it are those who are both new to programming and who are familiar with the spoken language of the code they are writing in.

Reading errors in Espy, as visible in Figures 11 and 12, was more complex, because in order to do it successfully one essentially must have a high command of both Spanish and programming. The single fastest participant was relatively highly ranked in both, but it generally appeared for the rest of the participants that more programming experience, as opposed to more Spanish experience, allowed for easier interpretation of errors, though the trend was markedly less steep than it was for the interpretation of English errors (see Figures 9 and 10).

The qualitative feedback from the participants, though, is where the distinction between the experience in working with Python and with Espy becomes clearest. All but two of the participants said that they found the English function they were given to be easier to read than the Spanish one, with some indicating that the reason they felt this way was because of word definitions: “I would say the English [function was easier to read], because [in the Spanish function] I didn’t know what ‘tam’ was - I just thought that was some arbitrary function, not ‘length,’” and “[The] Spanish [function] was a little trickier because I had to figure out what some of the words meant, but [the] English [function] was easy to read.”

When asked how they would feel about reading a large block of code in English and in Spanish, some, like participant 4, said they anticipated both would be just as difficult: “I guess [the difficulty would be] about the same - just in terms of words and language, I understood all the words to the same degree,” while others, namely participants 7, 8, 9, and 10, who all had low levels of Spanish experience, agreed that at a large scale, English code would be far easier to process, saying “English would be easier to read most likely, because it’s my native language,” and “The English one [would be easier to read a lot of], because I know English and I don’t know Spanish.”

An interesting trend that came up in the qualitative responses was that participants with even a moderate grasp of Spanish did not seem more intimidated by the idea of reading a large block of Spanish code over a large block of English code. Participant 6, for example, who had a self-ranked Spanish level of 4 out of 10, said “With my Spanish capabilities I have no trouble translating things like ‘if’ and ‘or’ and ‘return.’ That’s easy, but if it were a real sentence [that I had to translate], obviously I wouldn’t understand that.” This feedback, combined with the fact

that most participants said they found the English code to be more enjoyable to read, points to the notable idea that, while it's not necessary to be fluent in a human language to be able to read or understand lots of code written in it (or at least not feel intimidated by it), it's far more enjoyable for someone to interact with code in their native language. Specifically, in the case of native Spanish speakers, the feedback indicates that while most of them, being proficient in English,[45] would not have trouble reading large amounts of English-based code, it would be far more comfortable for them to read code written in Spanish.

When writing code in Espy and asked about their experience doing so as compared to their experience writing code in English Python, the participants had mixed reviews with regard to their enjoyment of the process. Most were not particularly opinionated, except for participant 8, who had high programming skill and low Spanish skill, who said "The English code was more enjoyable [to write in], because I know English much better than I know Spanish," and participant 5, who had relatively high skill in both Spanish and programming: "The Spanish one [was more enjoyable], because it's novel and I get to think about Spanish and use fun new keywords. I was more focused on making it correct. I felt less sure of what I was doing, so I was more focused." While I find participant 8's answer to be fairly predictable and on-par with my general predictions about the participants' interaction with the code based on their Spanish and programming experience, I find the latter answer (participant 5's answer) to be particularly interesting. That quote indicates that people who already have coding experience may find the act of writing code in a language that they are not used to to be more work than writing code in a language that they are used to, even if they speak the novel language quite well. This is not to

[45] Krogstad et al., "English Proficiency on the Rise Among Latinos."

say that native Spanish speakers who already know how to code wouldn't enjoy Espy, but rather that, native language or not, switching from writing code in English to writing code in Spanish requires a bit of extra brain-power at first.

After writing a bit of code in English Python and in Espy, I asked the participants which language they felt was more understandable to them, or which language they would find easier to explain to somebody else. The first 5 participants, who all had Spanish skill levels of at least 4 out of 10 (which is enough to understand the translations for simple words like "if" and "return") all either said that the languages were equally understandable or that Espy was easier to understand. Participants 1, 2, and 4, who responded the latter, all had Spanish knowledge of at least 7 and low programming knowledge (2 or 1 out of 10). Participant 3, who also responded that Espy was easier to understand, had Spanish knowledge of 5 out of 10 and programming knowledge of 10 out of 10. Predictably, the last four participants, who all had Spanish knowledge of 2 out of 10, reported finding English Python to be easier to understand and explain to somebody else than Espy. This question of understandability and explainability, while it might seem minor, is one of the most important aspects of learning to code. When someone is new to programming, if they find it hard to or are unable to discuss their code with somebody else, the process of learning to code, which is challenging enough as it is, becomes practically impossible and infinitely more frustrating. The good news is that the qualitative feedback from the participants indicates that even a moderate amount of knowledge in the code's spoken language makes the code as easy for an individual to discuss with other people as code in their native language is to them, but anybody who is learning to code and has no experience with the spoken language that the code is built off of will essentially be unable to discuss it with others.

Another key aspect of learning to code is the feeling of ownership of code. To someone who is new to the world of programming, blocks of code often look foreign and uninviting, and being able to write code, understand it, and feel as though it's "yours" is very important when it comes to continuing in the field of programming. In other words, anecdotal evidence suggests that a novice programmer who feels as though everything they write is alien to them, no matter how much they practice, will not likely stay with programming in the long run, whereas somebody who feels like they know and own their code will. In that vein, I asked the participants which of the code blocks that they wrote felt more like "theirs" in a broad sense. All the participants except for participants 3, 5, and 7 said that they felt more of a sense of propriety over the English code than they did over the Spanish code. Participant 8 took an objective view toward the concept of ownership, saying that the English code felt more like theirs because "I can write cleaner code in English,"[46] whereas participant 6 took a broad stance on ownership: "It's like ownership of language… My language is English and I'm a lot more confident in things that I write in English over things that I write in Spanish, so ownership-wise it's easier for me to say I wrote [the English code] over [the Spanish code] because English is my primary language." It is clear, then, that ownership of code can be directly tied to the code's spoken language: code written in a native language feels more proprietary as well as "cleaner" and better-structured than non-native language code does.

In all, the qualitative feedback indicates that people prefer to write code (from an enjoyment, understandability, and ownership standpoint) in their native language. Whether they actually perform better writing code in their native language is a different question, but the

[46] Note that this participant had relatively high programming experience, at 6 out of 10 - I doubt that someone unfamiliar to programming would understand or feel the importance of "clean code."

reality is that speed and performance is not what is important when learning to code. Rather, learning to code is tricky enough as it is, and it feels more accessible to people when the bit of human words they do see in the code are in their own language.

Furthermore, this preference for native-language programming revealed during the testing of Espy means that using it or a language like it to teach programming to Spanish speakers would be more effective than using Python or another English-based programming language. Allowing people to read, write, and debug code in a language that they feel comfortable in means that they will feel more at home in the code that they write themselves, and as the qualitative feedback from the testing indicates, those new coders will find their work to be more understandable to them and easier to work with and discuss with others. Using tools like Espy, therefore, to teach people in Latin America how to code would be a large step in the right direction for easing some of the cultural apprehension that surrounds building technology there.

Furthermore, the feedback that participants who were proficient in Spanish gave about reading Espy, and specifically the fact that they found it relatively easy to do, is good news with regard to the level of damage that a product like Espy could do to the coding community. A valid concern about a product like Espy is the possibility that if it became widely used, it could splinter the world of developers, which right now is a fairly unified community, into the world of developers who code in English and the world of developers who code in Spanish. If the latter group, which would surely be smaller than the former for a long time, found it impossible to read code written in English, they would lose access to the ability to communicate with other programmers and discuss their code. The data gathered during testing, though, indicates that even moderate exposure to a non-native language makes code written in it equivalent to code in

one's native language in terms of legibility. That data, combined with the fact that two-thirds of Latin Americans are proficient in English,[47] means that that problem of splintering will likely not occur down the line.

Of course, a product like Espy will not instantly make Latin America the next Silicon Valley, but over time, and if it is used widely enough, it has the potential to turn the outlook on learning to code from one that is intimidating and exclusive to one that is entertaining, fun, and challenging. And if it makes people feel as though the code they write belongs to them and to their culture, instead of belonging to the English-speaking world, people who would otherwise have been inhibited by the challenges of coding in their non-native language will keep coding for longer. Eventually, this group will be numerous enough to form a new generation of Latin American developers who did not give up when learning to code because the process felt like theirs instead of the English-speaking world's.

[47] Krogstad et al., "English Proficiency on the Rise Among Latinos."

4. Conclusion

To conclude, I'd like to return to Ernesto Sábato's quote from *Uno y el universo*: "El poder de la ciencia se adquiere gracias a una especie de pacto con el diablo: a costa de una progresiva evanescencia del mundo cotidiano. Llega a ser monarca, pero, cuando lo logra su reino es apenas un reino de fantasmas" ["The power of science is acquired thanks to a pact with the devil: the cost is a gradual evanescence of the everyday world. Science becomes king, but when it does, its kingdom is just a kingdom of ghosts"].[48] The world that Sábato feared would evanesce when he wrote *Uno y el universo* in 1945 maybe has not totally disappeared, but it certainly has changed as a result of science and technology. Even if you've never used a phone, satellites circle above you, anybody in the world can see a picture of your home with Google Earth, and technology has sped up globalization and connectedness between people everywhere. Today, then, the risk of a way of life evanescing can only be prevented by embracing technology, not by avoiding it. The world is advancing with technology one way or another, and the only way for a group of people to stay relevant and alive in the global community is to take part in technology. Latin America was dealt a very strong blow toward their ability to do this by the neoliberal expansion of the last century, and so now, time is of the essence if Latin America is to become a global, independent technology-producing region.

Espy is not perfect. It's small, unable to translate third-party packages, and as of right now it has only one maintainer and bug-fixer. The data presented in this thesis, though, indicates that, whether or not Espy is the ideal tool, the spoken language that a programming language is

[48] *Sábato, Uno y el universo*, 12.

written in does make a difference in the process of coding and to the individuals themselves who are coding.

Focusing more on making students feel as comfortable as possible when they learn to code, including by not making them interact with software that is not in their native language, would be a step in the right direction toward making coding for non-English speakers an accessible and welcoming idea for students. And once those new programmers start making full-scale technology for themselves and their peers that can rival that which is made in Silicon Valley, they will be taking part in a movement toward creating independent, full-grown technological innovation centers in every region of the world.

Appendix

1. Translation Tables between Python and Espy

Python	Espy
abs	abs
abstract	abstracto
abstractmethods	metodosabstractos
add	mas
all	todo
alloc	asign
alphanumeric	alfanumérico
and	y
any	cualq
append	adjuntar
apply	aplicar
arg	arg
argument	argumento
ArithmeticError	AritmeticaError
as	como
as_integer_ratio	como_ratio_entero
assert	afirmar
AssertionError	AfirmacionError
attribute	atributo
AttributeError	AtributoError
BaseException	BaseExcepcion
basestring	basepalabra
bin	bin
binary	binario
bit	bit
BlockingIOError	ImpedirIOError
bool	bool
brace	llave
break	romper
BrokenPipeError	PipaRotaError
buffer	búfer
BufferError	BuferError
bufferstr	búferpal
builtin	prehecho
bytearray	bytematriz
bytes	bytes
BytesWarning	BytesAviso
call	llamar
callable	llamable
capitalize	mayuscular
center	centro
character	carácter
ChildProcessError	ProcesoHijoError
chr	carac
class	clase
classmethod	metclase
clear	limpiar
cmp	cmp
codec	codec
coerce	forzar
collection	colección
compile	compilar
conjugate	conjugar
ConnectionAbortedError	ConexionAbortadaError
ConnectionError	ConexionError
ConnectionRefusedError	ConexionRechazadaError
ConnectionResetError	ConexionRecolocadaError
contains	contiene
continue	continuar
copyright	copyright
count	total

Python	Espy
credits	atrib
debug	depruar
decode	decodificar
def	def
default	estándar
del	elim
delattr	elimatr
delete	eliminar
delitem	elimartic
denominator	denominador
deprecated	despreciado
DeprecationWarning	DespreciadoAviso
descr	descr
descriptor	descriptor
dict	dicc
digit	dígito
dir	dir
div	div
divmod	divmod
do	hacer
double	doble
elif	osi
else	sino
empty	vacío
encoding	codificación
endswith	terminacon
enumerate	enumerar
EnvironmentError	AmbienteError
EOFError	EOFError
eq	ig
eval	eval
except	excepto
Exception	Excepcion

Python	Espy
exec	ejec
execfile	ejecarchivo
expandtabs	expandtabs
extend	extender
False	Falso
field	campo
file	archivo
FileExistsError	ArchivoExisteError
FileNotFoundError	ArchivoNoEncontradoError
fillchar	llenacarác
filter	filtrar
finally	porfin
find	encontrar
flag	señal
float	flot
FloatingPointError	FlotError
floor	inferior
floordiv	divinferior
for	para
format	formato
format_spec	espec_formato
formfeed	salto de página
fromhex	dehex
fromkeys	declaves
function	función
FutureWarning	FuturoAviso
ge	mai
GeneratorExit	GeneradorSalir
get	sacar
get_empty_storage	saca_almacenamiento_vacio
get_strategy	saca_estrategia

Python	Espy
getattr	sacaatr
getattribute	sacaatributo
getformat	sacaformato
getitem	sacaartic
getitem_str	sacaartic_pal
getitems_fixedsize	sacaartics_tamañofijo
getiteritems_with_hash	sacavaloresiter_con_hash
getiterkeys	sacaclavesiter
getitervalues	sacavaloresiter
getnewargs	sacanuevosargs
getslice	sacaparte
globals	globales
gt	maq
has_key	tiene_clave
hasattr	tieneatr
hash	hash
head	cabeza
heap	pila
help	ayuda
hex	hex
hexadecimal	hexadecimal
iadd	imas
id	id
if	si
ignore	ignorar
import	importar
ImportError	ImportarError
ImportWarning	ImportarAviso
imul	imul
in	en
in place	en su lugar

Python	Espy
IndentationError	SangriaError
index	índice
IndexError	IndiceError
infinity	infinito
init	inic
initialize	iniciar
input	entrada
insert	insertar
instancecheck	instanciaverif
instantiate	instanciar
int	ent
integer	entero
intern	intern
InterruptedError	InterrumpidoError
invert	vuelta
ipow	ipot
is	es
is_integer	es_entero
is_true	es_cierto
IsADirectoryError	EsDirectorioError
isalnum	esalnum
isalpha	esalfa
isdecimal	esdecimal
isdigit	esdig
isinstance	esinstancia
islower	esminusc
isnumeric	esnumerico
isspace	esespac
issubclass	essubclase
issubtype	essubtipo
istitle	estitulo
isupper	esmayusc
item	artículo

Python	Espy
itemsize	tamartic
iter	iter
iterable	iterable
iterate	iterar
iterator	iterador
iteritems	iterartics
iterkeys	iterclaves
join	juntar
keepends	guardarcolas
key	clave
KeyboardInterrupt	TecladoInterrumpir
KeyError	ClaveError
keyword	palabra clave
lambda	lambda
le	mi
len	tam
length	tamaño
leq	mei
license	licencia
list	lista
ljust	ijust
locals	locales
long	larg
LookupError	BusquedaError
lower	minusc
lshift	imover
lstrip	idecapar
lt	meq
map	mapa
marshal	alinear
max	max
MemoryError	MemoriaError
memoryview	vistamemoria

Python	Espy
metaclass	metaclase
min	min
mod	mod
mul	mul
mutable	mutable
name	nombre
NameError	NombreError
NaN	NuN
ne	ni
neg	neg
new	nuevo
newline	lineanueva
next	sig
None	Nada
nonzero	nocero
not	no
NotADirectoryError	NoUnDirectorioError
NotImplemented	NoImplementado
NotImplementedError	NoImplementadoError
numerator	numerador
object	objeto
oct	oct
offset	offset
open	abrir
operand	operando
or	o
ord	ord
OSError	OSError
OverflowError	SobranteError
padding	relleno
pair	par
partition	particion
pass	pasa

Python	Espy
PendingDeprecationWarning	AvisoDespreciadoPendiente
PermissionError	PermisoError
pickle	envinagrar
pop	pop
pos	pos
pow	pot
prefix	prefijo
print	imprimir
ProcessLookupError	BusquedaProcesoError
property	propiedad
raise	levantar
rand	dy
range	rango
raw_input	entrada_cruda
rdiv	ddiv
rdivmod	ddivmod
read-only	solo-leer
readonly	sololeer
recursion	recursión
RecursionError	RecursionError
reduce	reducir
reduce_ex	reducir_ex
ReferenceError	ReferenciaError
reload	recargar
remove	quitar
replace	reemplazar
repr	repr
ResourceWarning	RecursoAviso
return	volver
return	volver
reverse	invertir
reversed	invertido

Python	Espy
rfind	dencontrar
rindex	dindice
rjust	djust
rlshift	dimover
rmod	dmod
rmul	dmul
ror	do
round	redond
rpartition	dparticion
rpow	rpot
rrshift	ddmover
rshift	dmover
rsplit	dquebrar
rstrip	ddecapar
rsub	drest
RuntimeError	EjecError
RuntimeWarning	EjecAviso
self	mismo
sentinel	centinela
sequence	secuencia
set	poner
set_strategy	pon_estrategia
setattr	ponatr
setdefault	ponestan
setitem	ponartic
setslice	ponparte
shape	forma
shift	mover
sign	signo
signature	firma
sizeof	tamde
slice	cortar
sort	ordenar

Python	Espy
sorted	ordenado
space	espacio
spec	espec
split	quebrar
splitlines	quebrarlineas
stable	estable
StandardError	ErrorEstandar
start	empieza
startswith	empcon
staticmethod	metestat
step	paso
StopAsyncIteration	PareAsyncIteracion
StopIteration	PareIteracion
str	pal
strides	trancos
string	palabra
strip	decapar
sub	rest
subclasscheck	subclaseverif
suboffsets	suboffsets
subscript	subíndice
suffix	sufijo
sum	suma
super	padre
swapcase	minmayusc
SyntaxError	SintaxisError
SyntaxWarning	SintaxisAviso
SystemError	SistemaError
SystemExit	SistemaSalir
tab	tab
TabError	TabError
tail	cola
title	titulo

Python	Espy
tobytes	abytes
tolist	alista
translate	traducir
True	Cierto
truediv	divcierto
trunc	trunc
truncate	truncar
try	probar
tuple	tuple
type	tipo
TypeError	TipoError
UnboundLocalError	LocalSueltoError
unichr	unicarac
unicode	unicod
UnicodeDecodeError	UnicodDecodError
UnicodeEncodeError	UnicodCodError
UnicodeError	UnicodError
UnicodeTranslateError	UnicodTraducirError
UnicodeWarning	UnicodAviso
unwrap	desembalar
update	actualizar
upper	mayusc
UserWarning	UsuarioAviso
value	valor
ValueError	ValorError
values	valores
vars	vars
viewitems	verartics
viewkeys	verclaves
Warning	Aviso
weakref	refdebil
while	mientras

Python	Espy
width	ancho
wrap	embalar
wrapkey	claveembalaje
wrapvalue	valorembalaje
xor	oex
xrange	xrango
ZeroDivisionError	CeroDivisionError
zfill	cllenar
zip	zip

2. Testing Functions

The code that participants were shown had function names and variable names shortened or otherwise modified so that their purpose wasn't immediately clear (asking someone to explain what a function called "negate" does doesn't test much more than their ability to read a function's name).

The ideas for the functions are mostly lifted from the "Try it yourself" sections of http://ap-n.us/books/Programming/Python%20Crash%20Course.pdf, and most are modified a bit.

Function 1: prints "abc" - change it to use the input() function to print someone's name

```
def f1():
    x = "abc"
    print(x)

f1()

def f1es():
    x = "abc"
    imprimir(x)

f1es()
```

Function 2: prints "abc" in upper, lower, titlecase - change it to print user input the same way

```
def f2():
    x = "aBc"
    print(x.lower())
    print(x.upper())
    print(x.title())

f2()

def f2es():
    x = "abc"
    imprimir(x.mayusc())
    imprimir(x.minusc())
    imprimir(x.titulo())
```

```
f2es()
```

Function 3: makes a list of four elements, prints the last one - use the quitar/remove function to remove the an element in the list, then print the last element again, without using the index function

```
def f3():
    l = ["John", "Paul", "George", "Ringo"]
    print(l[l.index("Ringo")])

f3()

def f3es():
    l = ["Juan", "Pablo", "Jorge", "Ringo"]
    imprimir(l[l.indice("Ringo")])

f3es()
```

Function A: negation function - change it to print "the negation of [param] is [result]"

```
def fa(x):
    if x is True:
        return False
    else:
        return True

def faes(x):
    si x es Cierto:
        volver Falso
    sino:
        volver Cierto
```

Function 4: makes a list of types of pizza, prints them all in a sentence - use append to add another type of pizza to the list, and if there are more than 4 types of pizza in the list, print "I really love pizza!" at the end

```
def f4():
    l = ["cheese", "pepperoni", "mushroom"]
    for e in l:
```

```
        print("I like", e, "pizza!")

f4()

def f4es():
    l = ["cheese", "pepperoni", "mushroom"]
    para e en l:
        imprimir("I like", e, "pizza!")

f4es()
```

Function 5: returns whether the given input is even (never used during testing)

```
def f5(n):
    if n % 2 is 0:
        return True
    else:
        return False

def f5es(n):
    si n % 2 es 0:
        volver Cierto
    sino:
        volver Falso
```

Function 6: returns whether the input is greater than zero (never used during testing)

```
def f6(n):
    if n > 0:
        return True
    else:
        return False

def f6es(n):
    si n > 0:
        volver Cierto
    sino:
        volver Falso
```

Function 8: factorial function - change it so that if n is 0, return 0; if n is negative, print a message indicating negative input, and return the factorial of the absolute value of n

```
def f8(n):
    if n < 1:
        raise ValueError
    if n is 1:
        return 1
    return n * f8(n - 1)
f8(n)

def f8es(n):
    si n < 1:
        levantar ValorError
    si n es 1:
        volver 1
    volver n * f8es(n - 1)

f8es(n)
```

Function 9: power function - change it so that if the base is 0, raise ValueError/ValorError; if the base is negative, handle it correctly: f9(-2, 3) = 8

```
def f9(b, p):
    if p is 0:
        return 1
    elif p is 1:
        return b
    else:
        return b * f9(b, p - 1)

def f9es(b, p):
    si p es 0:
        volver 1
    osi p es 1:
        volver b
    sino:
        volver b * f9es(b, p - 1)
```

Function 10: reverses a string - change it so that it prints a message when the word passed in is a palindrome

```
def f10(w):
    if len(w) is 1:
        return w
    return f10(w[1:]) + w[0]

def f10es(p):
    si tam(p) es 1:
        volver p
    volver f10es(p[1:]) + p[0]
```

Function 11: the isPrime function - change it so that if n is -1, 0, or 1, raise a ValueError/ValorError. If n is negative (and not -1) return whether its absolute value is prime.

```
def f11(n):
    for i in range(2, n):
        if n % i is 0:
            return False
    return True

def f11es(n):
    para i en rango(2, n):
        si n % i es 0:
            volver Falso
    volver Cierto
```

3. Participant Data Collection Form

Knowledge of Spanish (1-10):
Knowledge of programming (1-10):

Reading
English function:
English time:
Output prediction errors:

Spanish function:
Spanish time:
Output prediction errors:

Which was easier to read? Which would you find easier to understand a lot of? How comfortable would you be explaining this code to somebody else?

Writing
English function:
English time:
Errors:

Spanish function:
Spanish time:
Errors:

Which was more enjoyable for you? Was one easier than the other? Which modifications are more understandable to you? Which of the changes you made would you find easiest to explain to somebody else? Which code feels more like yours?

If no errors came up during testing
English error:
Time:

Spanish error:

Time:

4. Raw Participant Data

Participant number	Spanish knowledge	Programming knowledge	First reading function	First writing function	English function	English reading time	English output errors
1	7	2	Spanish	Spanish	f3	0:34.09	0
2	9	1	English	Spanish	fa	1:36.39	0
3	5	10	Spanish	English	f8	0:14.36	0
4	9	2	English	English	f2	2:04.06	0
5	5	8	Spanish	Spanish	f11	0:32.43	0
6	4	5.5	English	English	f8	0:46.43	0
7	2	1	Spanish	English	f2	0:50.89	1
8	2	6	Spanish	English	f8	0:21.02	0
9	2	1	Spanish	Spanish	f2	1:10.34	0
10	2	6	English	Spanish	f9	1:07.85	0

Participant number	Spanish knowledge	Programming knowledge	Spanish function	Spanish reading time	Spanish output errors
1	7	2	f4	1:26.98	0
2	9	1	f2	1:09.18	0
3	5	10	f10	0:59.08	0
4	9	2	fa	0:54.00	0
5	5	8	f8	0:28.98	0
6	4	5.5	f9	1:14.78	1
7	2	1	fa	1:06.40	0
8	2	6	f11	2:33.54	0
9	2	1	f1	0:49.47	0
10	2	6	f8	0:47.78	0

Participant number	Spanish knowledge	Programming knowledge	English writing time	English writing errors	Spanish writing time	Spanish writing errors
1	7	2	2:45.25	1	1:35.94	1
2	9	1	1:36.04	0	0:44.61	0
3	5	10	1:11.08	3	2:22.25	0
4	9	2	0:31.4	0	0:28.06	0
5	5	8	1:31.12	1	1:47.18	1
6	4	5.5	2:53.95	1	1:40.59	0
7	2	1	3:16.59	2	1:32.89	1
8	2	6	1:31.08	1	2:20.52	0
9	2	1	1:26.79	0	1:26.02	0
10	2	6	1:16.35	1	2:15.27	0

Participant number	Spanish knowledge	Programming knowledge	English Error	Time	Spanish Error	Time
1	7	2	NameError	18.07	SintaxisError	0:35.02
2	9	1	NameError	25.05	IndiceError	0:20.92
3	5	10	IndexError	7.44	NombreError	0:15.78
4	9	2	IndexError	22.76	NombreError	0:33.13
5	5	8	SyntaxError	15.78	NombreError	0:18.21
6	4	5.5	IndexError	19.29	NombreError	1:00.15
7	2	1	IndexError	35.34	SintaxisError	0:50.32
8	2	6	NameError	19.75	IndiceError	0:17.69
9	2	1	SyntaxError	22.6	NombreError	0:48.87
10	2	6	NameError	10.5	SintaxisError	0:26.32

5. Participant Qualitative Feedback Transcriptions

Participant 1:

Reading:

Which was easier to read?
The one in English

Which would you find easier to understand a lot of?
English

How comfortable would you be explaining this code to somebody else?
The Spanish function, because the index() function in the English code was unfamiliar

Writing:

Which was more enjoyable for you?

I liked the Spanish one better

Was one easier than the other?

The Spanish one was easier, just because of the content of the function

Which modifications are more understandable to you?

The Spanish one, because of the content of the function

Which code feels more like yours?

The English one because it was changed more

Participant 2:

Reading:

Which was easier to read?
The Spanish one

Which would you find easier to understand a lot of?
Neither - they might be equally difficult. There isn't any extreme where one is very easy and the other is impossible to read.

How comfortable would you be explaining this code to somebody else?
They're both pretty equal, both are like a 5 or 6 out of 10 in difficulty, the Spanish one.

Writing:

Which was more enjoyable for you?

Neither.

Was one easier than the other?

The Spanish one was easier

Which modifications are more understandable to you?

The Spanish one, just because in the English one uses the print statement which has a lot of different options. The Spanish one would be easier to explain to somebody else.

Which code feels more like yours?

The English one

Participant 3:

Reading:

Which was easier to read?
I would say the English one, because [in the Spanish function] I didn't know what "tam" was - I just thought that was some arbitrary function, not length.

Which would you find easier to understand a lot of?
About the same

How comfortable would you be explaining this code to somebody else?
Very comfortable, assuming they had basic knowledge of programming, but I'd find Spanish to be just as easy to explain as English.

Writing:

Which was more enjoyable for you?

I enjoyed the Spanish function because I thought it was a more interesting problem.

Was one easier than the other?

Neither was easier to work with than the other, the only issue I ran into was not remembering some of the keywords

Which modifications are more understandable to you?

The Spanish one

Which code feels more like yours?

Neither in particular

Would you feel more comfortable with writing a big function in English or in Spanish?

Probably English, just because I don't have a list of Spanish syntax in front of me.

Participant 4:

Reading:

Which was easier to read?

I thought the Spanish was easier to read, just because of the syntax.

Which would you find easier to understand a lot of?

I guess about the same - just in terms of words and language, I understood all the words to the same degree.

How comfortable would you be explaining this code to somebody else?

Really comfortable for both of them.

Writing:

Which was more enjoyable for you?

The English one

Was one easier than the other?

The Spanish one was easier because it was more intuitive to me, I felt like I knew what to do off the bat. The language English vs Spanish didn't make a difference to me.

Which modifications are more understandable to you?

The Spanish one

Which code feels more like yours?

The English one, because I gave it more to do and changed it more.

Did English vs Spanish make a difference to you a lot, a little, or not at all?

Not at all

Participant 5:

Reading:

Which was easier to read?
They were about the same, Spanish was a little trickier because I had to figure out what some of the words meant, but English was easy to read.

Which would you find easier to understand a lot of?
If I could ask what things meant I feel like they would be fairly similar, but if I had to do a bunch on my own, the one in English would be easier. It was hard to try to guess what things mean, like “levantar” which I think means “raise” but I don’t know for sure.

How comfortable would you be explaining this code to somebody else?
Probably the English one by a little, but they’re both fairly straightforward.

Writing:

Which was more enjoyable for you?

The Spanish one, because it’s novel and I get to think about Spanish and use fun new keywords. I was more focused on making it correct. I felt less sure of what I was doing, so I was more focused.

Was one easier than the other?

No, there was more to do in the English one, but it wasn’t more difficult.

Which modifications are more understandable to you?

About the same, they're not super conceptually rigorous. The Spanish one would be easier to explain to someone else.

Which code feels more like yours?

The Spanish one

Did English vs Spanish make a difference to you a lot, a little, or not at all?

The Spanish one makes you go step by step more, because you have to think about what everything is called, the English one is just "I know how to do it, let's do it."

Participant 6:

Reading:

Which was easier to read?

Neither of them, they were both equally difficult.

Which would you find easier to understand a lot of?

Honestly no, because with my Spanish capabilities I have no trouble translating things like "if" and "or" and "return" is easy, but if it were a real sentence, obviously I wouldn't understand that.

How comfortable would you be explaining this code to somebody else?

No, because they're both just simple functions and I can explain simple functions. I'd be able to do it even in Spanish.

Writing:

Which was more enjoyable for you?

No, they were both equally enjoyable.

Was one easier than the other?

No - they both had the same challenges.

Which modifications are more understandable to you?

I think it's kind of the same for both. Explaining the changes I made to somebody else would both be the same.

Which code feels more like yours?

It's like ownership of language… My language is English and I'm a lot more confident in things that I write in English over things that I write in Spanish, so ownership-wise it's easier for me to say I wrote [the English code] over [the Spanish code] because English is my primary language.

Did English vs Spanish make a difference to you a lot, a little, or not at all?

None at all, other than the fact that I had to ask you "what do I put for 'return'?" If I had to code a lot more with no guidance it would be fine as long as I had access to translations.

Participant 7:

Reading:

Which was easier to read?

The Spanish function was more straightforward, but obviously I know English --

Which would you find easier to understand a lot of?

The English would definitely be easier.

How comfortable would you be explaining this code to somebody else?

I suppose I'd be more comfortable explaining the English one.

Writing:

Which was more enjoyable for you?

No.

Was one easier than the other?

One thing was that when I edited the Spanish one I already had some practice [editing the English function], so since I did it second, I had already been warmed up.

Which modifications are more understandable to you?

The English ones.

Which of the changes you made would you find easiest to explain to somebody else?

Which code feels more like yours?

Neither.

Did English vs Spanish make a difference to you a lot, a little, or not at all?

For the Spanish I definitely had to have you tell me what some of the words meant, but beyond that it didn't make too much of a difference.

Participant 8:

Reading:

Which was easier to read?

I had less trouble understanding the English function.

Which would you find easier to understand a lot of?

English would be easier to read most likely, because it's my native language.

How comfortable would you be explaining this code to somebody else?

I could explain them both pretty well.

Writing:

Which was more enjoyable for you?

The English one was more enjoyable, because I know English much better than I know Spanish.

Was one easier than the other?

English was easier.

Which modifications are more understandable to you?

The English one I understand better just off the bat.

Which of the changes you made would you find easiest to explain to somebody else?

Right off the bat I'd explain English better, but if I could look at both of them for a while I could explain them equally.

Which code feels more like yours?

Probably the English one. I can write cleaner code in English.

Did English vs Spanish make a difference to you a lot, a little, or not at all?

A little bit of difference at this length and level.

Participant 9:

Reading:

Which was easier to read?
The English one is easier to understand because the function is easier and because I speak English.

Which would you find easier to understand a lot of?
The English one would be easier to infer instead of the Spanish one because I don't speak Spanish.

How comfortable would you be explaining this code to somebody else?
No, they'd be the same.

Writing:

Which was more enjoyable for you?

Not really.

Was one easier than the other?

No, about the same.

Which modifications are more understandable to you?

The English one because it's easier to remember what the functions like "mayusc" mean [in Spanish].

Which of the changes you made would you find easiest to explain to somebody else?

The English one, because it's in English.

Which code feels more like yours?

The English one just because of the language.

Did English vs Spanish make a difference to you a lot, a little, or not at all?

A little difference. It depends on if I speak Spanish or not. The English one is more intuitive so it's easier to remember and understand.

Participant 10:

Reading:

Which was easier to read?
The English function.

Which would you find easier to understand a lot of?
The English one, because I know English and I don't know Spanish.

How comfortable would you be explaining this code to somebody else?
Now that I know what all the Spanish words mean, I'd feel equally comfortable, but with a larger block of code I'd feel more comfortable with English just because I speak English.

Writing:

Which was more enjoyable for you?

Neither, but English would be more fun on a larger block of code.

Was one easier than the other?

The English one was easier to write.

Which of the changes you made would you find easiest to explain to somebody else?

Probably about the same for each.

Which code feels more like yours?

The English one, because I didn't know most of the words I had to use for the Spanish one.

Did English vs Spanish make a difference to you a lot, a little, or not at all?

Given the fact that you were here to tell me translations, it made a little different, but if nobody was here to tell me how to say “return” in Spanish, it would have made a lot of difference.

6. Espy Codebase

All espy code that was written for this project is freely available for distribution, modification, personal use, and download at https://github.com/akercheval/espy.

Bibliography

Ahmed, Azam and Perlroth, Nicole. "'Our Phones Are Being Monitored': How a Hacking Story Unfurled." New York Times, (New York, NY). June 19, 2017.

Buono, R. A.Dello. "Technology and Development in Latin America: Urgent Challenges for the 21st Century." Perspectives On Global Development & Technology 11, no. 3 (June 2012): 341-351. Academic Search Premier, EBSCOhost (accessed April 2, 2018).

Ceurvels, Matteo. "Twitter's Riding a Political Wave in Latin America." eMarketer. August 15, 2017. https://www.emarketer.com/Article/Twitters-Riding-Political-Wave-Latin-America/1016334.

"Estadísticas e indicadores." Comisión Económica para América Latina y el Caribe. 2018. http://estadisticas.cepal.org/cepalstat/WEB_CEPALSTAT/estadisticasIndicadores.asp?string_busqueda=internet.

Frazier, Steve and Walsh, Mary Williams. "Mexico -- Borrowed Time --- Nation in Jeopardy: Earthquake Aftermath Points Up Weaknesses of Mexican Leadership --- Citizens Scorn Ruling Party for Inaction, Inefficiency; Big Boost for Dissidents --- Rebuffing the Yankee Giant." *Wall Street Journal*, Eastern edition (New York, NY). Oct. 15, 1985.

"Global mobile OS market share in sales to end users from 1st quarter 2009 to 2nd quarter 2017." Statistica. April 2, 2018. https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/.

"Internet/Broadband Factsheet." Pew Research Center Internet and Technology. February 5 2018. http://www.pewinternet.org/fact-sheet/internet-broadband/.

"Internet usage in Latin America - Statistics & Facts." Statistica. Accessed April 2, 2018. https://www.statista.com/topics/2432/internet-usage-in-latin-america/.

"Internet Usage Statistics." Internet World Stats. December 31, 2017. https://www.internetworldstats.com/stats.htm.

Krogstad, Jens Manuel, Stepler, Renee, and Lopez, Mark Hugo. "English Proficiency on the Rise Among Latinos." Pew Research Center Hispanic Trends. May 12, 2015. http://www.pewhispanic.org/2015/05/12/english-proficiency-on-the-rise-among-latinos/.

Kurlat Ares, Silvia G. "A Farewell to the Future." *Technology, Literature, and Digital Culture in Latin America: Mediatized Sensibilities in a Globalized Era*. Routledge Interdisciplinary Perspectives on Literature, (2016): 62

"Latin America Population." World Population Review. November 2 2017. http://worldpopulationreview.com/continents/latin-america-population/.

"Nuestra Empresa." FAdeA - Fábrica Argentina de Aviones "Brig. San Martín" S. A. accessed April 2, 2018. https://www.fadeasa.com.ar/fadea/?page_id=212.

"Number of monthly active Twitter users in the United States from 1st quarter 2010 to 4th quarter 2017 (in millions)." Statistica. Accessed April 2, 2018. https://www.statista.com/statistics/274564/monthly-active-twitter-users-in-the-united-states/.

Penix-Tadsen, Phillip. "Latin American Game Design and the Narrative Tradition." (2015): 205-230.

Ryan, Camille and Lewis, Jamie M. "Computer and Internet Use in the United States: 2015." American Community Survey Reports. September 2017. https://www.census.gov/content/dam/Census/library/publications/2017/acs/acs-37.pdf.

San Pedro, Emilio. "Cuba internet access still severely restricted." *BBC*. March 21 2016. http://www.bbc.com/news/world-latin-america-35865283.

Sábato, Ernesto. *Uno y el universo*. Barcelona: Editorial Seix Barral, S. A., 1945.

Solanas, Fernando E., dir. *Argentina latente*. 2007. Argentina: Cinesur. Web (https://www.youtube.com/watch?v=lXxDMzjVPng).

"United States Internet Users." Internet Live Stats. Accessed April 2, 2018. http://www.internetlivestats.com/internet-users/us/.

Zmerli, Sonja and Castillo, Juan Carlos. "Income inequality, distributive fairness and political trust in Latin America." *Social Science Research*, (July 2015).