https://doi.org/10.48009/4_iis_2020_295-305

*Issues in Information Systems*
*Volume 21, Issue 4, pp. 295-305, 2020*

# PREPARING TEACHING SWIFT PROGRAMMING AND ACTUAL CLASSROOM TEACHING

***Thomas L. Ngo-Ye, Alabama State University, tngoye@alasu.edu***
***Jae J. Choi, Pittsburg State University, jchoi@pittstate.edu***
***Dexter Gittens, Alabama State University, DGittens@alasu.edu***

## ABSTRACT

*Apple Swift programming is the behind-the-scenes technology powering millions of iOS and Mac OS X apps that we use every day. Our business school is offering Swift programming as an elective course for all business major students. At the beginning of this initiative, we had no expertise in Apple Mac platform and Swift programming. This paper documents our journey to overcome the challenges of unfamiliar technology as well as the knowledge learned along the way. In the process of preparing for teaching Swift programming, we found that the most efficient way to acquire Swift programming skills is through hands-on practice and exploration. In our exploration, we discovered many unique features of Swift. This study also reports our actual teaching experience and the classroom observations. We share the tips and insights gained from preparing and teaching Swift. This paper potentially makes some practical contributions to the area of teaching Swift programming. CIS faculties interested in teaching Swift programming may benefit from this study by making use of the practical guide, lessons learned, and workarounds. It is our objective to assist CIS faculties to make their preparation and teaching of Swift programming more efficient and smoother.*

**Keywords:** Swift Programming, Xcode, Playgrounds, Apple Mac Platform, Preparing Teaching, Actual Classroom Teaching, Mobile App Development

## INTRODUCTION

In recent years, Apple products and services became widely popular among consumers. Apple iPhone and iPad are adopted by a large segment of the population. Even for people who normally use Microsoft Windows-based desktop and laptop PCs, iPhone and iPad are still chosen for satisfying mobile computing needs. Of course Apple has its own desktop and laptop PC products – iMac and MacBook. While a few business school teachers and students use Apple PCs, iMacs and MacBooks are unfamiliar to the majority of business school teachers and students. Apple Swift programming is an important skill for developing the iOS and Mac OS X apps. In our business school, we are offering Swift programming as an elective course to all business students.

In this paper, we report our endeavor of preparing to teach Swift programming, as well as our actual teaching experience. We documented the struggles we encountered and presented the resolutions we found. Our first-hand experience shows that the best way to learn Apple Mac skill and Swift programming skill is through practicing. Through self-exploration of Swift programming, we made many discoveries of features unique to Swift. This study also documented our classroom observations. We believe that the insights and tips gained through preparing and teaching Swift would be valuable to CIS faculties planning to teach Swift programming. This paper is aimed at helping those CIS faculties to prepare their Swift class more efficiently and to have a smoother teaching experience.

### What are Swift Programming, Xcode, and Playgrounds?

Apple is currently promoting a new concept called “Everyone Can Code”. It is a new approach to coding computer programs, which aims to empower everyone to learn, write, and teach code (https://www.apple.com/education/k12/teaching-code/) (Apple, 2020).

Along with the “Everyone Can Code” paradigm, Apple developed a new computer programming language called Swift. Swift programming language is used to build iOS and Mac OS X apps. Swift is claimed by Apple to be powerful and easy to use, even for beginners. Swift incorporates the best of C and Objective-C, without the constraints of C compatibility (https://www.tutorialspoint.com/swift/) (tutorialspoint, 2020). As a successor to both the C and Objective-C languages, Swift contains low-level primitives such as types, flow control, and operators.

Swift also offers modern object-oriented features such as classes that programmers found very useful. The Swift syntax is concise yet expressive. Swift programming code is safe by design and at the same time it creates software that runs very fast.

Apple Xcode allows programmers to write simpler code with a declarative Swift syntax. As Xcode is native on all Apple platforms, apps written in Xcode gain incredible native performance (https://developer.apple.com/xcode/) (Apple, 2020). To create and run Swift Xcode program, we needed to download and install Xcode app on Mac PC. The most current version of Xcode as of June 2020 is 11. While Mac PC is needed to run Xcode, Swift Playgrounds can run on iPad. Swift Playgrounds is a new app for iPad that can be used to teach students to write Swift code in a fun and interactive way (https://www.apple.com/swift/playgrounds/) (Apple, 2020). Xcode app is like a powerful IDE such as Microsoft Visual Studio. Swift Playgrounds is a special type of Xcode project and environment that hides a lot of complexity of Xcode. Large sections of the book "Intro to App Development with Swift" are designed as Swift Playgrounds projects. Students can learn the basic knowledge of Swift and Xcode in this safe Playgrounds environment. One convenient feature of Playgrounds is that it displays the result of Xcode program immediately in the sidebar at the right side of Playgrounds, which makes coding very interactive.

**Why Offering Swift Programming Class?**

Our school has offered "CIS XXX Mobile App Development with Swift" since fall 2019. It is a junior level undergraduate elective class opening to all students in our business school. We strongly recommend Computer Information Systems (CIS) major students to take this course. The rationales for providing Swift Programming course as an elective course in the business school are as following.

Presently, it is a common recognition that like reading, writing, and math, computer programming is a critical skill for twenty-first century knowledge workers. Apple maintains that coding skill is critical in assisting students to thrive in the near future driven by the technology revolution. We taught students programming, and companion skills like problem solving and critical thinking at the same time. Apple believes that its "Everyone Can Code" resources can aid teachers to give all students the chance to study programming in school and to prepare for a professional career.

Swift is arguably a great first programming language for beginners. It can help open doors to the world of computer programming for many students. According to Apple, Swift was designed to be any student's first programming language. Apple crafted Swift to be a programming language that anyone can learn. In Swift we can use familiar words and phrases, such as "add" and "remove," and see what we are generating as we type in our Swift code. First-time programmers can download Swift Playgrounds —an app that provides an interactive and fun experience of learning Swift programming. The Swift Playgrounds app enables students to watch the effects immediately unfold, such as using Swift code to create dancing robots and chatting robots that can answer questions in a pre-programmed way. In summary, Swift is a programming language that is intuitive, interactive, easy and fun to learn. Moreover, for teachers, Apple created free curriculum to teach Swift. Apple's "Everyone Can Code" curriculum makes Swift Programming easy to teach.

While Swift is a great programming language for naïve beginners, Swift is also a very powerful programming language for experienced programmers. In fact, Swift is used by millions of programmers to build the iOS and Mac OS X apps that we use every day. Swift has all the tools that we need to inspire students to create advanced and innovative apps in the future.

Given the above attractive features, Swift is chosen as the programming language that we offer in the elective course to all business school students.

## PREPARING TEACHING SWIFT PROGRAMMING

In this section, we report our journey of preparing to teach Swift programming class. We highlight the struggles as well as relatively successful tips we learned in the process.

**Hardware and Software Requirements for Swift Programming**

Before presenting how to prepare human capital of teaching Swift, we first describe the necessary hardware and software required for Swift programming. As a proprietary software product, Swift is best running on Apple platform. More specifically, to run Swift Xcode, we do need Apple hardware – Mac PCs. In July 2019, our school acquired 25 new Apple Mac desktop PCs to create a designated Apple Mac Lab. The Mac Desktop PCs come with pre-loaded OS X and basic apps. However, the out-of-the-box Mac was not equipped for running Swift Xcode or Playgrounds. We had to get Xcode app installed on Mac. Moreover, we needed the free Apple Swift textbook "Intro to App Development with Swift" from Apple's "Everyone Can Code" curriculum loaded on Mac. Furthermore, we also needed the book's companion student project files downloaded to Mac. In the following sections, we will document the details of how we managed to successfully achieve the tasks of installing Xcode, textbook, and student project files. In Figure 1. We present the technical features of the Mac PCs that we used in our Apple Mac Lab for teaching Swift. Please note that the software including both the OS X and Xcode keeps receiving new updates as we always try to keep the software current all the time.


**Figure 1.** Technical Features of Mac for Teaching Swift

**Assumptions of Teacher Skills**

We assumed the majority of CIS faculty did not have specialized skills in Apple Swift programming. Like most faculty in business school, we are experts in using Microsoft Windows systems. Although we used iPhone and iPad for mobile computing, we had very limited exposure to Mac PC. Before teaching Swift programming, we needed to overcome two obvious obstacles. First, we had to learn how to operate Mac PC to the extent that we could perform basic routine operations, such as copy/paste text, file, and folders. Second, we had to learn the specific new programming language – Swift, including its unique syntax and how to navigate and run Xcode/Playgrounds. We assumed that CIS faculty, like us, also possessed the basic general programming knowledge in languages like Java or C#.

**Searching for Help**

Realizing that we were going to teach Swift programming in fall 2019, over the summer we put in a lot of efforts in preparing for teaching this new subject. Given the challenges of teaching on a new platform (Mac) and a new programming language (Swift), we had to overcome a series of setbacks and frustrations. It is our primary goal to

share our first-hand experience so that others can avoid traps and pitfall, and be more efficient in getting ready to teach Swift.

First, we conducted Internet searches with key phrases such as "teaching Swift Programming" and "syllabus for Swift Programming class" to find several PDF documents describing Apple's "Everyone Can Code" curriculum. Apple's website also provided some very general information of Swift. However, these Apple materials do not clearly provide practical step-by-step guides on how to get started for teaching Swift. A few "Swift Programming" class syllabuses we found online did offer a few useful clues though. We learned that (1) Apple provides the free textbook "Intro to App Development with Swift"; (2) We need an Apple ID/Account (3) We need install Xcode app. While this information was useful, it did not provide us with a practical guide for teaching Swift.

Next, we took the second step, which was to seek some practice guidelines from Apple. From Apple's website Education section https://www.apple.com/us/shop/goto/educationrouting (Apple, 2019), we found the instructions "If you are a student or teacher, visit the Apple Store for Education or call 1–800–692–7753". So on 6/3/2019 we called 1–800–692–7753 to transfer to education department. However, we never reached a live person who could have addressed our questions.

Then, we searched through our email contact and found two Apple employees that had email exchange with our school. These two Apple employees' email addresses end with "@apple.com". So we sent an email to them. However, for some unknown reason the above email sent to XXX@apple.com was not delivered. We do not know whether these two employees left Apple or if Apple blocked our email into its email system.

Overall, our initial effort in search for help to get started in teaching Swift programming did not generate much fruitful results.

**Most Efficient Way to Acquire Swift Programming Skills – Hands-on Practice**

In middle July 2019, the new Mac desktop PCs that our school ordered finally arrived. With the physical Macs in hand, we tried to get our hands dirty by exploring the new system. We sought the help from our University's IT department. To overcome the first obstacle of "learn how to operate Mac PC", we threw ourselves into Mac Desktop PC. We worked with the IT department on our Apple Mac Lab. The IT personnel created two accounts for the Mac Lab. The first one was a student account for all students to use. We shared the student account and its corresponding password to all students. With the student account, one can run some Apple apps on Mac, such as the Safari web browser. However, with the student account, one cannot update or install apps. The second account for Mac Lab is the administrator account. The administrator account is supposed to be used only by IT personnel. In theory, the administrator account and especially its corresponding password should not be shared with faculty. The administrator account should always be kept as a secret from students. With the administrator account and its password, one can run, update, and install apps on Mac. For our convenience, the administrator account and its password were shared with IT faculty. It later proved to be very handy when teaching the Swift programming class.

After practicing on Mac, we found some similarity with Windows PC. Mac's "Finder" is like Windows Explorer, which leads to common folders such as My Documents, Downloads, and Desktop. From Mac's "Finder" one can find all apps installed on the Mac, just like "All Programs" in Windows. Mac arranges the most commonly used apps' icons in the horizontal bar at the bottom of the Mac desktop. In MAC PC, when running an app, one can click the top menu bar and it shows the context sensitive menu. For example, when "TextEdit" app is active, the top menu shows File menu. For the most part, Mac is intuitive and easy to use. The file and folder operations in Mac are similar to those in Windows.

The Mac Desktop comes with a set of wireless Apple keyboard and mouse. The Apple mouse is a bit different from traditional Windows mouse in terms of scrolling up and down. We found that the best way to treat Apple mouse was to consider it as a small smart phone for swiping up and down. To enable context sensitive menu by right clicking Apple mouse, we had to set the configuration. We clicked the top left corner of Apple icon in the horizontal bar at the top of the Mac desktop, and then we chose the System Preference menu – Mouse. After setting this up, we could conveniently right click Apple mouse to bring up context sensitive menu for tasks such as copy, paste, delete, and rename.

Sometimes the Safari web browser on Mac accidentally was zoomed too big. To correct this problem, we clicked top menu and chose view actual size to back to normal size.

**Short Mac Workshop by an Apple Employee**

In early August 2019, our school was fortunate to be able to invite an Apple employee to come to our campus to give a short and free workshop on how to use Mac. The main purpose of this workshop was not to assist CIS faculty with their preparations to teach Swift programming, but instead to introduce all business school faculty on the use of Mac for teaching. Since we had already explored Mac features and the first part of the workshop was mainly marketing, the workshop did not add much value for our preparation of teaching Swift. However, in the question and answer section, we were able to get some valuable information.

Thinking ahead about how we wanted students to capture their future Swift programming projects by screen-shots for the purpose of grading, we raised the question of how to do Print-Screen in Mac. The Apple employee (thereafter referred to as John Doe) showed us the following ways to take screen shots on Apple Mac PCs:

- Command + Shift + 5 provides many choices from screen shots (picture) to screen record (video). From option, we can choose microphone input voice and where to save the captured screen information. After record video, we can further encode it to different resolution to save disk space and save it as a MOV format file.
- Command + Shift + 3 captures the whole screen.
- Command + Shift + 4 captures a portion screen
- Command + Shift + 4 then press space bar captures the active screen

Among the above choices, the last one is most practical and fits our grading purpose the most.

Previously, we searched on Apple website and located the free Apple Swift textbook at https://books.apple.com/us/book/intro-to-app-development-with-swift/id1118575552 (Apple, 2017). However, on that page, we only saw description of the book and its picture. We could not download the textbook. To address our question of how to get started preparing to teach Swift programming, John Doe showed us how to search and download the free Apple Swift textbook. John Doe explained to us that we had to run the iBooks app and login with an Apple ID. The Apple ID is the same ID that one uses on his/her iPhone or iPad. You can also create a brand new Apple ID. After running the iBooks app and login with an Apple ID, you can search by book title, such as "Intro to App Development with Swift". iBooks app shows multiple results of relevant books. "Intro to App Development with Swift" is the most introductory version of the Swift books. There are two editions of this textbook. One is the student edition and the other one is the teacher edition. With the help of John Doe, we downloaded both the student and teacher edition of the textbook to the instructor's Mac PC in the Lab. Next, John Doe demonstrated how to navigate the textbook by swiping back and forth on Apple mouse.

Next, we recalled that on page 5 of the student edition of textbook, there were web links to the zip file of student projects. We clicked the link, downloaded the zip file, and copied it to the desktop as a folder. John Doe showed us that when you opened the folder and double-clicked a Playgrounds file from a book chapter to run it, a window popped up to ask you whether you wanted to download and install Xcode as an app. Installing Xcode of course requires the administrator account on the Mac. After installing Xcode, John Doe ran a few Playgrounds web pages to show us what they look like.

John Doe suggested to us that we should follow the Apple's free textbook "Intro to App Development with Swift" for teaching Swift in the beginning. When we asked John Doe if there is a PDF version of the textbook "Intro to App Development with Swift", he replied that the textbook is only available in the format of iBooks app. In other words, this textbook can only be opened and viewed on Apple devices, where iBooks app can run, including, Mac PC, iPad, and iPhone, where iBooks app can run. John Doe mentioned that there was no paper copy of the textbook and no one can print the textbook. While students can view the textbook on their iPhone, running the Xcode projects requires Mac PC according to John Doe.

We also asked John Doe whether Apple's "Everyone Can Code" curriculum provided complimentary instructor materials such as PowerPoints and test bank for the textbook "Intro to App Development with Swift". John Doe answered that no such instructor materials existed for this textbook. The only available materials are the student edition and the teacher edition of the book. The teacher edition of the book includes a web link to the zip file of instructor projects files, which contain the solution codes for the programming exercises in the student edition of the book.

Moreover, John Doe also recommended the website https://www.hackingwithswift.com/ (hackingwithswift, 2020), which is a good website to learn Swift programming language. It has an introductory book called "hackingwithswift".

**Self Exploration of Swift Programming and Discoveries**

With a physical Mac PC in hand and help from University IT personnel and the valuable guide provided by the Apple employee John Doe, we were all set to learn Swift programming in preparation to teach it. We used both the student edition and the teacher edition of the textbook. We read through both books. More importantly, we worked through the student project files chapter by chapter. After completing each chapter's student exercises, we opened the instructor's edition project files and studied the solution code and compared it to ours. For the most part, Swift programming is straightforward, especially for CIS faculty who had existing general knowledge of programming language.

We discovered that Swift has some unique syntax and concepts, which are different from Java and PHP. Next, we are going to highlight those aspects that can be surprising to people unfamiliar with Swift. We will also report some special tips we learned through trial and error.

Swift programming is case sensitive for keywords including let, var, func, struct, for, if, else, enum, case, switch, and default. Swift keywords should be lowercase. If we use uppercase of these keywords, it will cause weird errors. Boolean value true and false must be lowercase. If we use "True" and "False", it will cause an error: "Use of unresolved identifier".

The best practice is that for constant, variable, function name, method name, property name, and name of a case, they should start with lowercase and use camel case for multiple words. On the other hand, struct name and enum name should start with uppercase to differentiate type from value. Type must start with uppercase, for example: String, Int, Float, Double, Bool, Date.

We can use numbers within constant, variable, and function name, as long as the first character is not a number. We can use an uppercase letter as the initial for constant, variable, and function name although it is against the best practice. Function name can only start with a letter or underscore, not a number. Moreover, underscores can be used at the beginning, middle, or the end of constant, variable, and function name.

Swift code with multiple layers of if/else statements may cause issues in Playgrounds, because Playgrounds doesn't have automatic matching feature for {}. It's easy to miss a {} in Playgrounds which leads to a confusing error. In the Xcode IDE, {} doesn't automatically match either. We had to manually ensure the match of {}. The best practice is to always type {} in pair. Otherwise it is easy to err on {}. After we typed opening {, we pressed "return" key on keyboard and the matching} was automatically entered for us. In this way, we will not miss matching {}. Similarly we needed to pay attention to matching pair of () and "". The best practice is to type () and "" first, then move the cursor back to the center and type content of code.

In Swift, after the keyword "if", we must use {}. Otherwise we will get an error message - "Expected '{' after 'if' condition", even if there is just one single statement in the {} after an "if". Similarly, in Swift, after the keyword "else", we must use {}. Otherwise we will get an error message - "Expected '{' or 'if' after 'else"', even if there is just one single statement in the {} after an "else". This feature is very different from other programming languages.

When we "copy and paste" lines of Xcode program from TextEdit (a text editing app in Mac, similar to Notepad or WordPad in Windows) to Playgrounds, we needed to manually edit double quote mark " in Playgrounds to make it

work. Otherwise, it will generate a strange error. Therefore, TextEdit behaves like Microsoft Word and PowerPoint in a way that the double quote mark " is not recognized well in programming environment such as Playgrounds.

Swift offers a new concept differentiating argument labels and parameter names, which are unnecessarily complicated. In most programming languages, we use the same identifier for argument labels and parameter names. In Swift, argument labels are used in parts of an invoking function, while parameter names are used inside the detailed codes of the invoked function. Argument labels and parameter names can have diffident identifiers, although they refer to the same argument/ parameter.

Another new concept in Swift is the so called "side effect". "Side effect" happens when a function is doing something such as printing to console, or doing something unrelated to the returning value, but the function name doesn't include such information. When "side effect" is detected, it indicates the need to rename function to be more descriptive.

In general to force quit an app on Mac, first we clicked the app to make it become the active app. Then we clicked the top left corner Apple icon in the horizontal bar at the top of the Mac desktop. Finally, we chose the menu Force Quit Application to stop the freezing app. In the Swift textbook, there is an exercise of creating infinite loop by calling a function within the same function. When running the infinite loop code in Playgrounds, as expected, it leads to Xcode program freezing or not responding to keyboard and mouse action.

If we maximize Xcode window to run playground programs, and if the Xcode freezes, there is no way to see the top menu to force quit using the keyboard and mouse. In that case, we had to resort to press power button on the back of Mac monitor and hold a few seconds. Then a menu will pop up, and from that menu we could choose restart. Next, we chose force Xcode quit. To mitigate the potential risk of Xcode playground freezing, we found the best practice was to not maximize Xcode screen. In fact, we resized the Xcode screen to very big, just enough to leave the top menu bar still visible. In this way, we could also see the time in the top menu bar.

Because Playgrounds automatically interpret and run Xcode program as we write, therefore it discovers errors in real-time. However, the downside is that sometimes the half-finished Xcode program (work-in-progress code as we type in Playgrounds) causes the Mac freezing. If we are able to enter the whole program in a batch, it will help avoid such trouble. One workaround we discovered was that for a complicated block of Xcode, we could write it in TextEdit and take our time to examine it mentally. After we ensured that the code was right, we copy and paste the whole chunk into Playgrounds.

**Plan for Swift Class Evaluation Items for Grading**

After learning Swift programming ourselves and being confident that we can present such knowledge in classroom, our next task was to prepare evaluation items for student grading. As mentioned before, the textbook "Intro to App Development with Swift" does not come with any test bank. Therefore, we had to design our own assignments and tests. We structured the Swift class as project-based. In each class meeting, in addition to a brief lecture, we spent most time working on hands-on projects from the student edition of the textbook. At the end of class meetings, we instructed students to capture the screen-shot of the last exercise project code and result and submit it to Blackboard for grading. We had about 21 such in-class projects. As for tests, we created two tests. Test 1 covered the first 10 chapters of the textbook. Test 2 covered the remaining chapters. We noticed that each chapter of the student textbook has a few multiple choice questions. After students selected an answer, the book showed the correct answer. Based on these questions, we adapted them and created test questions. We manually typed the questions and answers in two text files. (50 questions for test 1 and 45 questions in test 2). Then we entered these self-created multiple choice questions to Blackboard for tests. We have a few considerations for this arrangement. First, this is an introductory programming class opened to all business major students. We had to control the level of difficulty to be reasonable to the majority of students. Second, the Mac Lab only opens during class meeting time. It is not an open lab like other Windows Labs, due to concern about the loss of valuable new Mac, wireless keyboard and mouse to theft. Because of this concern, the business school has a policy that when teachers leave the Mac lab, all students are asked to leave also, and the Mac Lab must be locked by the teacher who is leaving. Moreover, most students do not have a personal Mac PC. Therefore, most course work is done on school Apple Mac Lab machines and the majority part of projects has to be done in class.

## CLASSROOM TEACHING AND OBSERVATIONS

In this section, we report our first-hand classroom teaching experience and observations. We highlight the issues we encountered and our resolution. In the first Swift class meeting, we spent some time teaching students how to use the Apple Mac PC. Moreover, we communicated to students that it's just an intro class to Swift programming and they must not have too much unrealistic expectation. For students interested in further learning, we suggested that they self-study the Apple Swift textbook at the next level for advanced topics of app development.

### Setup Classroom Environment

We created a brand new Apple ID for the purpose of login iBooks app to download the textbook. We manually downloaded the student edition of the textbook on each and every Mac in the Lab. We suggested that students should sit in the same seat in all the Swift class meetings so that his/her student project folder is on one specific Mac for the purpose of saving programming work in one meeting and continuing the work in the next meeting. In the first class meeting, we guided students on how to open the textbook in iBooks app and downloaded the student project zip file. We showed students how to make a copy of the zip file and place it on their desktop. We also demonstrated to students how to rename the student project folder with their name, so that they can easily identify their project folder. By including student name in his/her folder name, different students from two sections can use the same Mac PC and not corrupt each other's work. Next, we asked students to run Playgrounds project file from chapter one. Here, the administrator account is needed to install Xcode app on student's Mac. We had to go through each Mac to enter the administrator account so that Xcode is installed. Moreover, to run Playgrounds file, it also asks for the administrator account. We went through the same process of manually entering the administrator account on each Mac to run Playgrounds file. In fact, if a Mac is restarted or logged off, we had to re-enter the administrator account to run Playgrounds file. We repeated this process throughout the semester.

To make the Mac Lab instructor PC work with the regular projector, we had to set display in the instructor PC:
We clicked the top left corner Apple icon in the horizontal bar at the top of the Mac desktop --> System Preferences --> Displays --> Optimize for: VGA Display
Scaled: 1280 X 1024
On the other hand, if we choose the default Built-in Retina Display, the big screen will be blurry and hard for students to read.

### Classroom Observations

We noticed that often students made typos when writing code in Playgrounds which caused the Xcode program to not work. We helped students to correct typos so that the program could run smoothly. We suggested to students that they should use the method of copy and paste to avoid typos. In some programs, a variable or a constant appears in multiple places. The variable or constant is normally very long and it is easy to make a typo. In this scenario, we strongly recommended to students to use copy and paste, rather than manually type.

Sometimes when we opened a Playgrounds file, the Xcode screen was empty and there was no content in the Xcode editor. To fix this issue, we right clicked the Xcode icon in the bar at the bottom of the desktop, and then we selected Quit from the menu. Next, we reopened the Playgrounds file and this method worked. However, the administrator account is needed again to run the Playgrounds file.

Sometimes a Playgrounds file keeps running and there is still no result in the sidebar and console. To fix this problem, we closed the Playgrounds and quit Xcode. Then we reran the Playgrounds file. Subsequently, we saw the Xcode result in a few seconds in the sidebar and console.

Some students double-clicked their folder on Mac desktop and found no file there. If it seems to be just a link, we can download student's project file folder from the textbook again, make a copy, and paste it to desktop, then rename the folder to include the student's name.

One morning the APWMAYVILLE power stabilizer in Mac Lab suddenly had a surge. All Mac desktops lost power and shut down. We had to press power button one by one to turn on the Mac PCs. We had to hold the power button for a few seconds to power up.

In one class meeting the instructor Mac PC didn't show a login screen. The screen was black with a question mark. We called for IT support. The IT personnel unplugged the power cord and plugged it in again. It worked!

In the Mac Lab, we observed that many students used the Mac desktop's cable to charge their personal iPhones. After students left the Lab, they did not plug back the cable to charge keyboards. After a while the keyboards and mice would run out of battery and were not usable for that time period. There were some incidents where students reported to us in class meetings that a mouse didn't work in the Mac Lab due to running out of battery. Thereafter, we had to make sure to plug cables back in to charge the keyboards and mice at the end of class meetings.

Due to the limited number of class meetings and too many chapters in the textbook, we found that in one semester we could not cover all the chapters of the book. We could only reach up to chapter 19. However, for the first 19 chapters, we could not find enough time to cover Chapters 5, 16, 17, and 18. These four chapters do not use Playgrounds. They directly use Xcode IDE and focus on App Graphic User Interface. They are not directly related to Swift programming language. For the continuity of teaching Swift programming language itself, we recommend that you teach chapters 5, 16, 17, and 18 at the end of a semester if time permits.

## ADDITIONAL RESOURCES FOR TEACHING SWIFT

In addition to the textbook "Intro to App Development with Swift", we found other resources including websites, YouTube videos, and books that are also relevant for teaching Swift. In this section, we list these resources.

### Websites

- https://www.apple.com/swift/ (Apple, 2020)
- https://developer.apple.com/swift/ (Apple, 2020)
- https://www.apple.com/education/k12/teaching-code/ (Apple, 2020)
- https://www.apple.com/swift/playgrounds/ (Apple, 2020)
- https://docs.swift.org/swift-book/ (swift.org, 2020)
- https://swift.org/ (swift.org, 2020)
- https://www.tutorialspoint.com/swift/ (tutorialspoint, 2020)
- http://online.swiftplayground.run/ (online.swiftplayground.run, 2020) (Online Swift Playground)
- https://www.hackingwithswift.com/ (hackingwithswift, 2020) (As recommended by the Apple employee)
- https://swiftforwindows.github.io/ (github.io, 2020) (Swift for Windows is an open source project that provides an easy-to-use development environment for the swift programming language to compile and run on Windows OS with the graphical interface.)

### YouTube Videos

https://www.youtube.com/watch?v=RWESSFwlNf0&list=PLXiaMWHbNgp2oJe2WhwmWSoRYvy75Z2Da
Courses by iBrent provide 39 free videos on Intro to App Development with Swift - Everyone Can Code (Courses by iBrent, 2017).

### Books

In iBooks app, we can type in a key phrase such "Swift" and find other more advanced Swift textbooks published by Apple. They are also free. The other two Swift books worth considerations are (Mathias & Gallagher, 2016; Miller, 2015).

## SUMMARY

In this paper, we document our journey towards preparing and actually teaching a Swift programming class. We highlight the challenges we encountered and how we overcame them with a bit of help from IT professionals. In our pursuit of being proficient in operating in Apple Mac platform and learning Swift programming, we realized that the most efficient way to acquire Apple Mac skills and Swift programming skills is through hands-on practice and exploration. Along the way, we discovered many unique features of Swift programming. Our first-hand Swift teaching experiences and classroom observations provide valuable lessons. The goal of this study is to provide a simple guideline to CIS faculty interested in teaching Swift programming, so that they can avoid the obstacles that we faced. This will greatly reduce the unnecessary inefficiencies. We also hope that the teaching tips we shared in this paper will facilitate CIS faculty in their preparation, so that their Swift teaching will be conducted in a more efficient manner. It is a fact that the majority of CIS faculties are not familiar with both Apple Mac platform and Swift programming. Therefore, this paper makes a potential practical contribution to the area of teaching Apple Swift programming. In the next phase of this ongoing research project, we plan to increase our proficiency in more advanced Swift programming, and then use this new knowledge to help students to prepare for the App Development with Swift Certification exam offered by Apple. We will report our new findings in the next paper.

## REFERENCES

Apple. (2019). *educationrouting*. Retrieved June 3, 2019, from Apple Shop: https://www.apple.com/us/shop/goto/educationrouting

Apple. (2017). *Intro to App Development with Swift (Everyone Can Code)* (Xcode 10 edition ed.). Apple Inc. – Education.

Apple. (2020). *K–12 Education Teaching Code*. Retrieved June 13, 2020, from K–12 Education Teaching Code: https://www.apple.com/education/k12/teaching-code/

Apple. (2020). *Swift Playgrounds*. Retrieved June 13, 2020, from Apple Swift: https://www.apple.com/swift/playgrounds/

Apple. (2020). *Swift The powerful programming language that is also easy to learn*. Retrieved June 13, 2020, from Apple Developer: https://developer.apple.com/swift/

Apple. (2020). *Swift. A powerful open language that lets everyone build amazing apps.* Retrieved June 13, 2020, from Apple Swift: https://www.apple.com/swift/

Apple. (2020). *Xcode*. Retrieved June 13, 2020, from Apple Developer: https://developer.apple.com/xcode/

Courses by iBrent. (2017, June 14). *Intro to App Development with Swift - Tutorial Series*. Retrieved June 13, 2020, from youtube: https://www.youtube.com/watch?v=RWESSFwlNf0&list=PLXiaMWHbNgp2oJe2WhwmWSoRYvy75Z2Da
github.io. (2020).

*Swift for Windows.* Retrieved June 13, 2020, from github.io: https://swiftforwindows.github.io/
hackingwithswift. (2020).

*Learn Swift 5.1 for free*. Retrieved June 13, 2020, from hackingwithswift: https://www.hackingwithswift.com/

Mathias, M., & Gallagher, J. (2016). *Swift Programming: The Big Nerd Ranch Guide* (2nd ed.). Big Nerd Ranch Guides.

Miller, B. (2015). *Swift in 24 Hours, Sams Teach Yourself* (2nd, Kindle Edition ed.). Sams Publishing.
online.swiftplayground.run. (2020).

*Online Swift Playground*. Retrieved June 13, 2020, from online.swiftplayground.run: http://online.swiftplayground.run/swift.org. (2020).

*About Swift*. Retrieved June 13, 2020, from swift.org: https://docs.swift.org/swift-book/swift.org. (2020).

*Welcome to Swift.org*. Retrieved June 13, 2020, from swift.org: https://swift.org/ tutorialspoint. (2020).

*Swift Tutorial*. Retrieved June 13, 2020, from tutorialspoint: https://www.tutorialspoint.com/swift/