

Jione Ban (2032228), Gordon Ng (2031408)

R. Vincent , Instructor

Advanced Programming, Section 1

Final Project

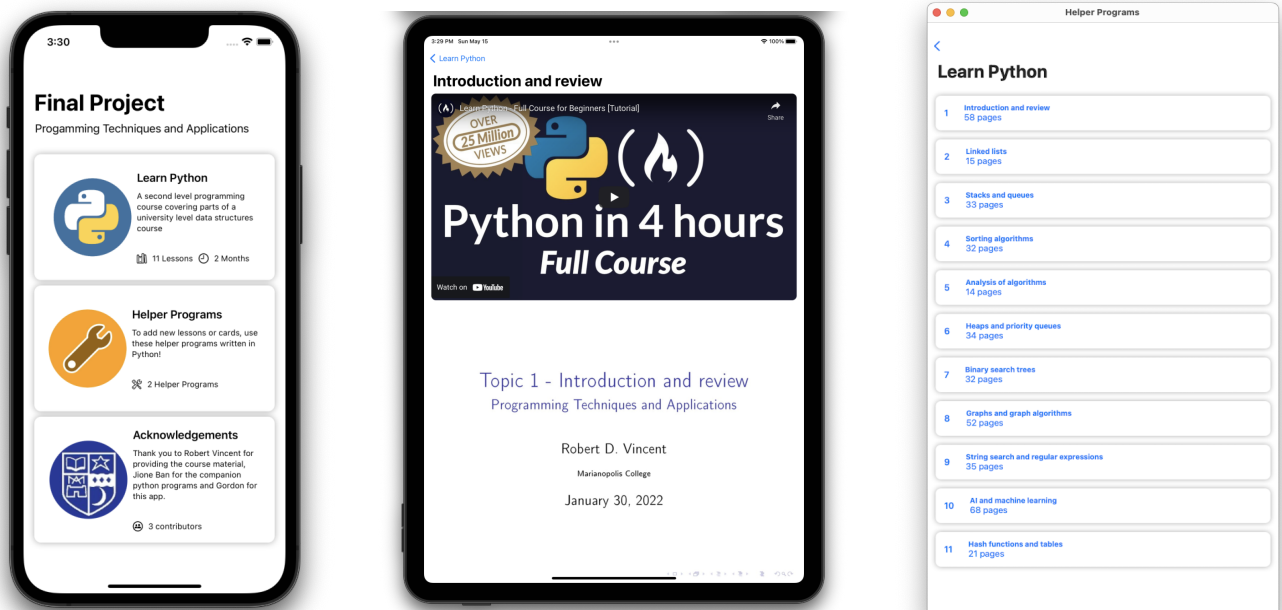
## Final Project Documentation

### User Manual

#### Preface

The program is an iOS app that contains all the notes from this school year for the course of Programming Techniques and Applications. It serves as a companion app to use while learning in class from the course and can be used on iOS, iPadOS and macOS devices. The following are example screenshots from each operating system, as well as a video demonstrating every function within the app:

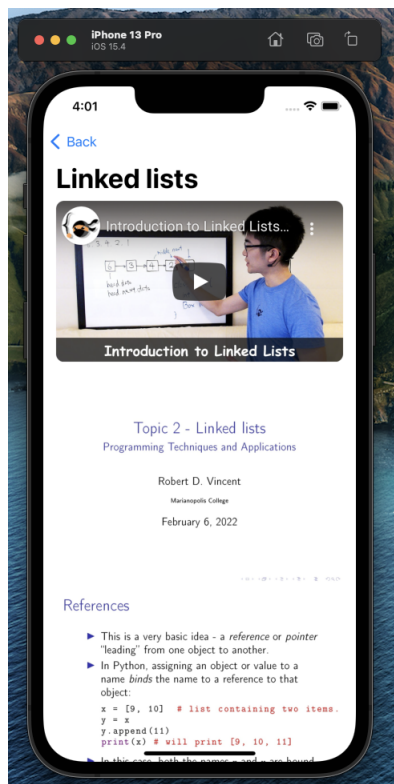
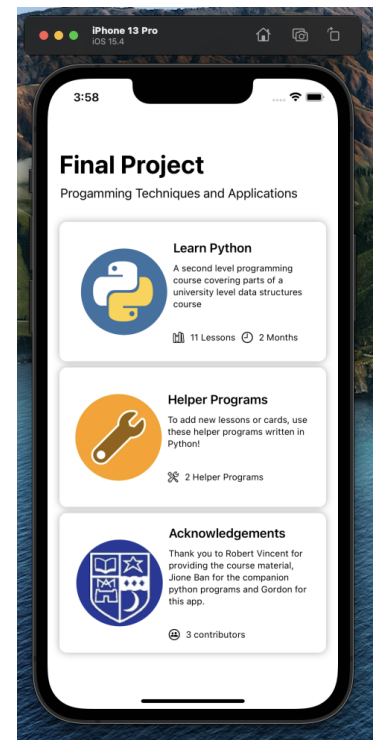
Demonstration video: <https://youtu.be/UbjNyS9u384>



## Part 1: The App

In the homescreen, the user is introduced to three different buttons. One to access the lessons module, another one to access the helper programs used for conversion of lectures and a final one acknowledging those who made the project possible.

Pressing the button for the “Learn Python” module leads to a new screen where there are 11 different cards, detailed with a title and number of pages. Pressing on any of the cards will bring the user to a new screen of that specific lesson.

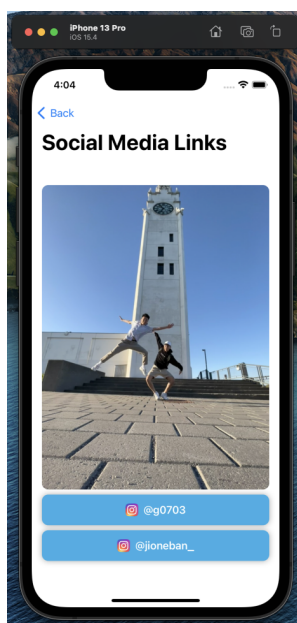
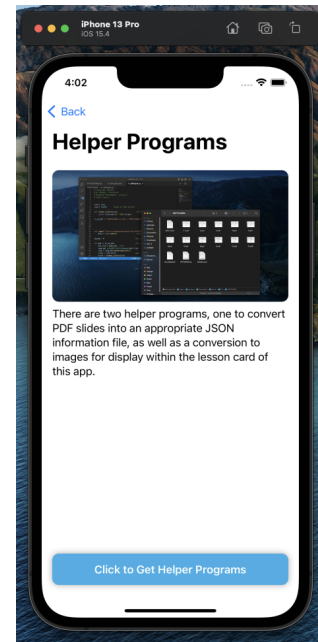


At the top of the lesson is a video giving either a visualization, a summary or an overview of each lesson.

This video player behaves exactly like the Youtube player.

To look at the lesson slides, the user can scroll down or use the scroll bar to scroll fast. Once the lesson is finished, the user can continue to the next lesson by pressing a button at the bottom of the lecture that will bring them to the top of the next lesson. The user can also exit the notes by pressing the back button on the top left or by swiping from the left corner.

From the homescreen, the user can access the helper programs by pressing the second module. Once pressed, a new screen is brought up along with an image of the programs, a description and a button displaying "Click to Get Helper Programs". Once the button is clicked, it opens a new link to Repl.it in the preferred browser containing the pdf2json and pdf2image helper programs.



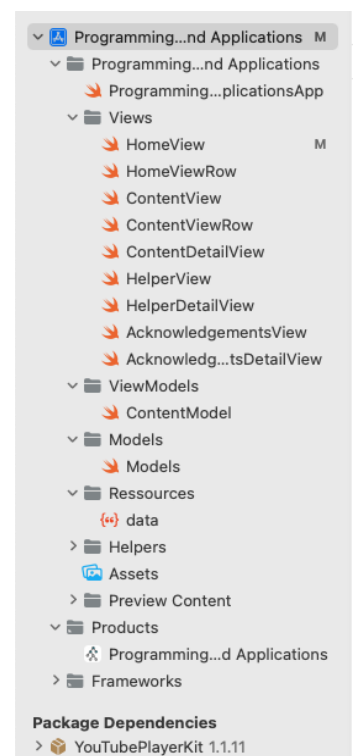
Again from the homescreen, the acknowledgements module can be accessed. On the module itself, it credits the contributors as well as the aspects they were a part of in the project. Pressing on the module, there is a picture of two developers and links to their social media.

## Documentation

### Part 1: Swift and SwiftUI APP

#### Introduction

This iOS app follows the file organization method known as MVVM, or Model, View, View-Model. Within these separate files, Models are files which hold the classes or structures holding or denoting the data. In this case, because the app sources most of its



data from a JSON file, these structures in the app denote the data. Views are UI elements which the user can interface and access. View-Models serve as the middleman between the View and View-Models. Additional folders are present that hold everything from the JSON file, external packages, image assets and project files.

## Tools Required

The creation of this application would not have been possible without a macOS computer as well as the Xcode development platform. This app was also distributed via testflight on the App Store for beta testing. This app also leverages a great amount of native Apple technologies to accomplish many of its functions. The app also uses [YoutubePlayerKit](#) , a third party swift package that interfaces with the Youtube API.

Since this app fetches information from a remote server, it would not have been possible without the free hosting provided by GitHub. The following link below demonstrates the remote server capability in action:

Demonstration: <https://youtu.be/N19-vEkd8cY>

## The Code

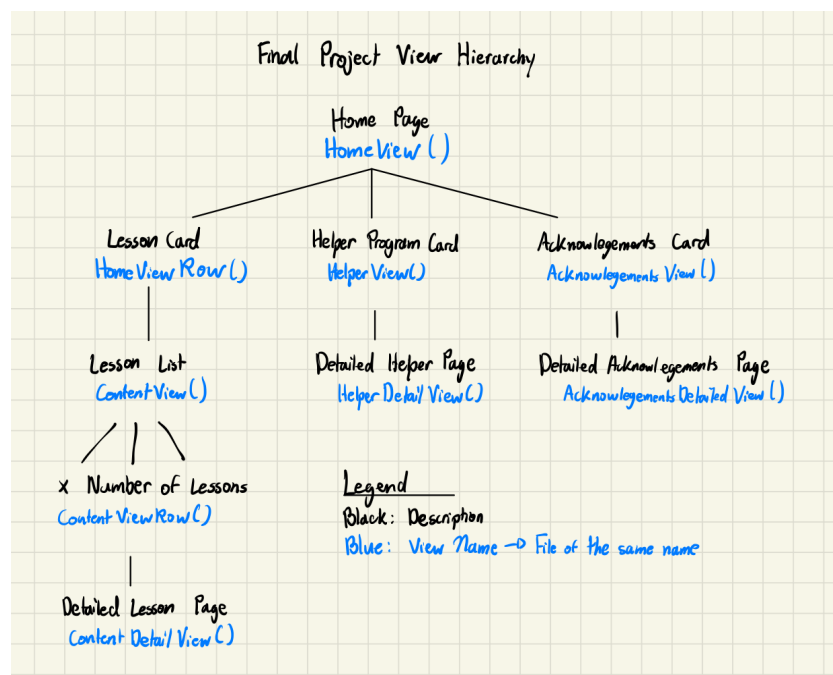
### Views

This describes the code under the views folder. Every file here leverages SwiftUI and its UI elements, as well as view structures or classes (It is therefore a form of Object Oriented Programming). SwiftUI specific UI elements used in views include:

- Stacks (VStack/HStack/ZStack): Allows for the layering of UI elements on the vertical, horizontal or Z axis

- **NavigationView:** View container allowing for linking of different view structures
- **NavigationLink:** Connects different view structures together
- **NavigationTitle:** Sets the title for that page
- **ScrollView:** View container allowing for scrolling
- **ScrollViewReader:** Obtains information from current location in a ScrollView page
- **ForEach():** Serves as a for loop iterating over different UI elements or views, used in order to be able to remotely add content
- **Button:** Requires an action to execute as well as a view to display
- **Rectangle:** Creates a rectangle UI element
- **Text:** Text UI element
- **Image:** Image UI element
- **.modifier:** Specific modifiers to customize a specific UI element

In the code are also custom view structures representing a collection of UI elements on screen. The following is an image of their description and hierarchy:




A more detailed visualization follows:



- HomeView()


## Final Project

Programming Techniques and Applications


**Learn Python**


A second level programming course covering parts of a university level data structures course

 11 Lessons  2 Months


**Helper Programs**

To add new lessons or cards, use these helper programs written in Python!


 2 Helper Programs

**Acknowledgements**


Thank you to Robert Vincent for providing the course material, Jione Ban for the companion python programs and Gordon for this app.

 3 contributors

- HomeViewRow()

**Learn Python**

A second level programming course covering parts of a university level data structures course

 11 Lessons  2 Months

- ContentView()

< Final Project

## Learn Python

1	Introduction and review	58 pages
2	Linked lists	15 pages
3	Stacks and queues	33 pages
4	Sorting algorithms	32 pages
5	Analysis of algorithms	14 pages
6	Heaps and priority queues	34 pages
7	Binary search trees	32 pages
8	Graphs and graph algorithms	52 pages

- ContentViewHolder()

1	Introduction and review	58 pages
---	-------------------------	----------

- ContentDetailView()

< Back

## Linked lists



Topic 2 - Linked lists  
Programming Techniques and Applications


Robert D. Vincent  
Marietta College  
February 6, 2022

### References

- This is a very basic idea - a *reference* or *pointer* "leading" from one object to another.
- In Python, assigning an object or value to a name *binds* the name to a reference to that object:  


```
x = [9, 10] # list containing two items.
y = x
y.append(11)
print(x) # will print [9, 10, 11]
```

- `HelperView()`



## Helper Programs

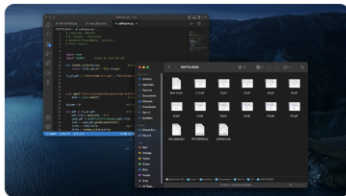
To add new lessons or cards, use these helper programs written in Python!

 2 Helper Programs

- `HelperDetailView()`

[< Back](#)

## Helper Programs

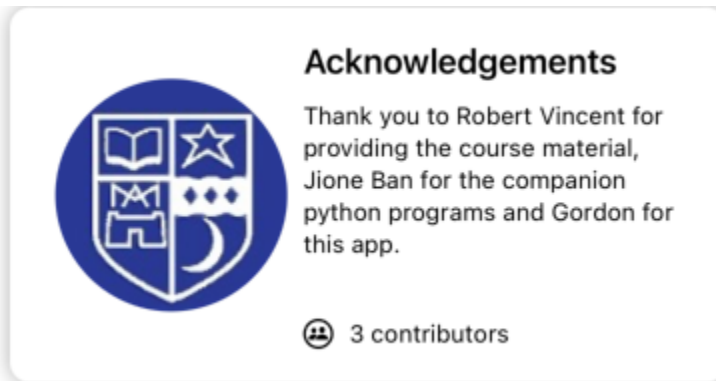


There are two helper programs, one to convert PDF slides into an appropriate JSON information file, as well as a conversion to images for display within the lesson card of this app.

[Click to Get Helper Programs](#)



- AcknowledgementsView()



- AcknowledgementsDetailedView()

[< Back](#)

## Social Media Links



 @g0703

 @jioneban\_

## ViewModels

View-Models serve as the middleman between the View and View-Models. In this app, this task is handled by the ContentModel() class. The following is a summary of its properties and methods:

- Initialiser: Fetches data from a server, in this case hosted by GitHub
- Properties:
  - modules: An array of the different modules, in this case, lessons from the JSON file
  - currentModule: Represents the current module and is an unwrapped optional of custom type Module -> defined in Models later in documentation
  - currentLesson: Represents the current lesson and is an unwrapped optional of custom type Lesson -> defined in Models later in documentation
  - currentModuleIndex / currentLessonIndex: Index integers for their respective types
- Methods:
  - getRemoteData(): Downloads remote JSON file and parses through it
  - beginModule(\_ moduleid: Int):
    - beginModule finds the appropriate module to display
    - moduleid of type int serves to compare with id property of modules
  - beginLesson(\_ lessonIndex: Int):
    - beginLesson finds the appropriate lesson to display

- lessonIndex of type int serves to compare with the total number of lessons
- hasNextLesson() -> Bool:
  - hasNextLesson serves to see if theres another lesson using index
  - Returns a boolean
- nextLesson():
  - nextLesson serves to set the next lesson

## Models

Models are files which hold the classes or structures holding or denoting the data. In this case, Models.swift denotes the data that is imported from a JSON file that is stored on a remote GitHub server.

## Other Files and Code

- Data.json represents the json file used by the app. This file was used for local testing before implementing remote server file access.
- Other files include the project .xcodeproject file, the YoutubePlayerKit dependency as well as file used only by Apple devices to run the app

## Part 2: Python Helper Program

There are two helper files that compose the helper program. The program is responsible for the integration of any pdf slides into the notes app. The app prompts for the files wanted and outputs the necessary files for use by the main app.

The first file is called `pdftojson` which opens a json file, handles information about the pdf and stores it as a new json file. The main library used was `PyPDF2` which had functions that gave access to titles of pdfs and length of pdfs. Each of the pdfs were held in a list to automate the editing of the json file. The program would open the json file and load it as a dictionary. Each lesson was tagged with an id number starting from 0, the title, the amount of pages and a link to a video for the lesson if the user chose to do so through the terminal and add it to the lessons list. Once the process was done for all the lessons, it would create a new json file called `"new_data.json"`.

The second file is called `pdf2image` and it handles the conversion of pdfs to images in the format of `jpg`. The main library used was `pdf2image` which contains the function `convert_from_path`, responsible for converting the pdf to an image. It takes in as an argument a pdf and outputs each slide as an image. Again, the pdfs are held in a list to automate the conversion. The images are then saved using the `save()` function so that the title of each image is properly named in the format `I1-01.jpg`.

The main helper program asks the user for the path of the pdf files and creates a list. Then the list is passed through as an argument to `pdftojson` and `pdf2image` and outputs the information about the pdf as a json file and converts the pdf to `jpg` images.