Develop in Swift
AP® CS Principles
Course Syllabus

Contents

Course Description

The Develop in Swift AP® CS Principles course is designed to build a foundation in programming using Swift—a powerful and intuitive open source programming language designed by Apple—while preparing students for the AP® Computer Science Principles Exam. Students will get practical experience with the tools, techniques, and concepts they need to build a basic iOS app. They'll learn the impact of computing, privacy, and security, and they'll explore the technology behind their own activities through interactive stories.

Course Textbooks

Develop in Swift AP® CS Principles, Apple Inc., 2020

Develop in Swift AP® CS Principles with Swift Teacher Guide, Apple Inc., 2020

Assessment

Self-grading quizzes are included at the end of each lesson, and test students' content knowledge. Skills are assessed through a series of activities and projects built into the Xcode playgrounds that accompany most of the lessons, as well as in the "Apply and Extend" sections in the Teacher Guide for this course, that encourage students to apply their skills in a new context.

Course Syllabus

Technology has a language. It's called code. And we believe coding is an essential skill. Learning to code teaches you how to solve problems and work together in creative ways. And it helps you build apps that bring your ideas to life. We think everyone should have the opportunity to create something that can change the world. So we've designed a new approach to coding that lets anyone learn, write, and teach it.

Curricular Mapping

The Develop in Swift: AP® CS Principles course is mapped to the AP® Computer Science Principles framework, which is organized around Big Ideas and Computational Thinking Practices. These are described in the AP® Computer Science Principles Course and Exam Description document located on the College Board website.

Big Idea 1: Creative Development **(CRD)**

Big Idea 2: Data **(DAT)**

Big Idea 3: Algorithms and Programming **(AAP)**

Big Idea 4: Computer Systems and Networks **(CSN)**

Big Idea 5: Impact of Computing **(IOC)**

Computational Thinking Practice **(CTP)**

Unit 1: Values (4 weeks)

1.1 Get Started with Values **(CRD, DAT, AAP)**

1.2 Play with Values

- 1.2.A Playground Basics **(CRD, DAT, AAP)**
- 1.2.B Naming and Identifiers **(AAP)**
- 1.2.C Simulation **(DAT, AAP)**
- 1.2.D Constants and Variables **(CRD, AAP)**
- 1.2.E Strings **(AAP)**
- 1.2.F Word Games **(AAP)**

1.3 Build a PhotoFrame App **(CRD)**

1.4 Design for People **(IOC, CRD)**

Episode 1: The TV Club (1 week)

E1.1 Searching for Content **(IOC)**

E1.2 Sharing Personal Information **(IOC)**

E1.3 Ordering Online **(CSN, IOC)**

Unit 2: Algorithms (6 weeks)

2.1 Get Started with Algorithms **(AAP)**

2.2 Play with Programs

- 2.2.A Functions **(AAP)**
- 2.2.B Types **(DAT, AAP)**
- 2.2.C Parameters and Results **(AAP)**
- 2.2.D Making Decisions **(AAP)**
- 2.2.E BoogieBot **(CRD, AAP)**
- 2.2.F Data Visualization **(DAT)**

2.3 Build a QuestionBot App **(CRD, AAP)**

2.4 Design an Experience **(CRD)**

Episode 2: The Viewing Party (1 week)

E2.1 Accessing the Show **(IOC)**

E2.2 Streaming on the Network **(CSN, DAT)**

Unit 3: Organizing Data (10 weeks)

3.1 Get Started with Organizing Data **(DAT, AAP)**

3.2 Play with Complex Data

- 3.2.A Instances, Methods and Properties **(CRD, AAP)**
- 3.2.B Arrays and Loops **(AAP)**
- 3.2.C Structures **(DAT, AAP)**
- 3.2.D Enums and Switch **(DAT, AAP)**
- 3.2.E Testing Code **(CRD, DAT)**
- 3.2.F Processing Data **(CRD, DAT, AAP)**
- 3.2.G Pixel Art **(DAT, AAP)**
- 3.2.H Password Security **(AAP)**
- 3.2.I Visualization Revisited **(DAT)**

3.3 Build a BouncyBall App **(DAT, AAP)**

3.4 Design a Prototype **(CRD)**

Episode 3: Sharing Photos (1 week)

E3.1 Capturing Images **(DAT, AAP, IOC)**

E3.2 Posting on Social **(DAT, AAP, CSN)**

Unit 4: Building Apps (6 weeks)

4.1 Get Started with App Development **(CRD, AAP)**

4.2 Play with App Components

- 4.2.A Color Picker **(AAP)**
- 4.2.B ChatBot **(AAP)**
- 4.2.C Rock, Paper, Scissors **(CRD, AAP)**
- 4.2.D MemeOkMaker **(CRD, AAP)**

4.3 Build an ElementQuiz App **(CRD, AAP)**

4.4 Design for Impact **(IOC)**

Unit 1: Values

Timeframe: 4 weeks

Introduction

Students will learn that all programming is about data, and will begin to use numbers and letters to express and manipulate values to solve problems. They'll consider how to name and describe different types of data, and how values can be selected and used to simulate the real world.

Students will explore these concepts in Xcode playgrounds, build a word game in a playground, and get started with Interface Builder to build and run their own app that displays a photo.

Students will consider how computing innovations have changed people's lives, and how technology impacts people differently. They'll also consider how they can reduce bias in their own design, and protect their own and others' copyrights.

Concepts covered

- What do programmers do?
- Skills (including collaboration)
- Fundamental concepts underlying the course—in particular, abstraction and algorithms
- Tools (Xcode IDE, Swift programming language, suited to building apps)

1.1 Get Started with Values

These Get Started modules can be used as a general introduction prior to coding the lessons in Xcode, as a way to get students thinking about the concepts they will be covering by connecting them to experiences students have already had or concepts they may already be familiar with.

In this module, students are introduced to the concept of a computer as a data processing machine capable of doing enormous amounts of mathematical calculations very quickly based on their input. They're introduced to the concept of values and expressions and the importance of proper naming and identifiers to represent more complex data. Computer simulations are examples of abstractions that help us to explore a complex system in detail. Strings are abstractions that represent an ordered grouping of characters.

1.2 Play with Values

1.2.A Playground Basics

Students will become familiar with playgrounds as they change values in code and see the results in the sidebar. This prepares students to use playgrounds as a flexible, dynamic learning space where they can experiment with running code and see the results immediately.

Concepts covered

- Discussion of numbers and place values, emphasizing that numbers and numerical concepts are fundamental to coding
- Experiment with a function that converts decimal numbers to binary numbers

- Experiment with the Color Picker as a way to display hexadecimal and decimal values and relate it to something visual
- Learn to translate decimal numbers to binary and vice versa
- Xcode playgrounds
- Make simple calculations with the arithmetic operators
- Code comments, error messages

Learning goals

Students will be able to:

- Change values in a playground and review results.
- Add and remove comments (non-executing code).
- Show and hide error messages.
- Change values to fix errors.

Sample activities

Students will use an Xcode playground to program simple mathematical calculations by editing some of the existing expressions and adding a few of their own, and see the results updating in real time in the sidebar. **[AAP-2.E][CTP-2.B][CTP-4.B]**

Students will practice commenting and uncommenting lines of code in an Xcode playground. They'll then practice troubleshooting error messages in the playground and fix a division by zero error. **[CRD-2.I] [CTP-4.C]**

Students will create a new playground and add a series of numbers to the playground, along with comments explaining what the numbers mean. They'll use their computational artifact to perform simple calculations, such as someone's birth year, or simple unit conversions. **[CRD-2.E] [CTP-1.B]**

1.2.B Naming and Identifiers

In this lesson, students are introduced to the concepts of names and identifiers. They'll learn the fundamentals of solving problems by using good names and identifiers as a basis for their work. The key concept of naming will help them to create reusable code that can be easily understood by others. Students learn how to declare constants and are introduced to abstraction as a way to manage the complexity of an app that contains many changing numerical values. Students also have hands-on practice with the live updating Xcode playground as a way to learn through real-time experimentation.

Concepts covered

- Using names in code, identifiers
- Autocompletion
- Correct naming convention, camel case
- Good naming practice—for self and others—and working with others
- Statements
- Declaring a constant
- Assignment operator

Learning goals

Students will be able to:

- Declare a new constant using the `let` keyword.
- Choose a meaningful, specific name for a value.
- Refer to a previously declared value by its name.

Sample activities

Students will write code in an Xcode playground to calculate the number of animals at a pet show. They'll create and name constants and learn that this is an example of abstraction, in which updating the value where the constant is defined is easier and safer. **[AAP-1.Db][CTP-3.C]**

Students will write code in an Xcode playground to calculate how much data can fit on an iPhone, creating and naming new constants to represent different amounts of binary data. **[AAP-2.A][CTP-2.A]**

1.2.C Simulation

Students will use a simulation of ant behavior to test and explore the effects of pheromones on ant trail formation. They'll explore how visualization tools such as the Playground Live View can help to communicate information about data. **[AAP-3.F] [CTP-1.A] [DAT-2.D] [CTP-2.B]**

1.2.E Strings

Students are introduced to the concept of strings, which are used to store a series of characters. They'll also learn about string interpolation, a way to construct a string using value placeholders within a string literal. And they'll learn how numeric values can be added to strings so they report values of different variables while a program is running.

Concepts covered

- Defining strings
- Unicode
- String interpolation
- How to view results in the playground
- Escape sequences
- Writing simple programs to generate a given text
- Writing simple programs to state the result of a calculation
- Text vs. images

Learning goals

Students will be able to:

- Identify different types of characters.
- Declare a new string constant using literal text, including emoji.
- Combine two existing strings.
- Combine a string of literal text with one or more previously declared values.
- Use the Show Result button to view longer string values.
- Combine a string with a number.
- Use escape characters to output special text.

Sample activities

Students declare string constants in an Xcode playground and combine them to form new strings. **[AAP-1.C][CTP-3.A]**

1.2.D Constants and Variables

Students expand their understanding of naming as they are formally introduced to the concepts of constants and variables, with specific examples and exercises of when and how they are used.

Concepts covered

- Variables
- Modeling real-world items in terms of constants and variables
- Declaring variables, assigning values
- Compound assignment
- Error messages in Xcode
- Debugging a program that uses variables and constants
- Models and simulations

Learning goals

Students will be able to:

- Describe the difference between constants and variables.
- Identify appropriate times to use variables over constants.
- Recognize and apply compound assignment operators to variables.
- Describe an ordered process for troubleshooting errors in code.
- Collaborate with others on a categorization activity.
- Describe how interrelated variables make up a model that simulates ant behavior.

Sample activities

Students will model a dart game strategy by using compound assignment to increment variables representing scores. **[AAP-3.F] [CTP-1.A]**

Students will work in collaborative groups to identify and categorize attributes of real-world objects into constants and variables. **[AAP-3.F] [CTP-1.A] [CRD-1.C] [CTP-1.C]**

1.2.F Word Games

Students will build a simple game that creates a silly story by prompting the user to fill in the blanks without seeing the rest of the story.

Concepts covered

- Defining strings
- Multiple-line string constants
- Writing simple programs to generate a given text

Learning goals

Students will be able to:

- Substitute the value of a string variable in an output statement.
- Concatenate two separate strings into a single string variable.

Sample activities

Students write a program to generate an original story using string interpolation. **[AAP-1.C][CTP-3.A]**

1.3 Build a PhotoFrame App

Students will create their first app using Xcode. They'll get familiar with the Xcode interface and learn how to add media as a resource within an Xcode project. They'll also learn how to build and run a project using the iOS Simulator. Students practice pair programming as a way to collaborate on creative endeavors.

Concepts covered

- Working with Xcode to build and run an app
- Using a template
- iOS Simulator
- Interface Builder
- Editing the storyboard
- Images as assets

Learning goals

Students will be able to:

- Start a new Xcode project.
- Build and run an app in the iOS Simulator or on a device.
- Show and hide inspectors in Xcode.
- Change the background color of a view.
- Add a resource file to an Xcode project.
- Edit the main storyboard in an Xcode project.

Essential questions

How might we build an app that fills a need or solves a problem?

How do apps convey information and/or provide inspiration?

Sample activities

Students are introduced to the concept of pair programming as a way to collaborate on creating a computational artifact. Together, students work on building the First App project in Xcode. **[CRD-1.A] [CTP-1.C] [CRD-2.J] [CTP-1.B] [CTP-6.A]**

1.4 Design for People

Computing innovations solve problems or provide a means of self-expression. Students will develop an understanding of the audience for the app they're designing by developing user profiles. Students also prototype and test their ideas in wireframe, paying particular attention to copyright considerations when using images. Creative Commons is presented as a resource for finding images that are free to use under certain restrictions. **[IOC-1.F] [CTP-5.E] [CTP-6.C]**

Additional college-level resource: 60 Second Prototyping. Apple Worldwide Developer Conference. 2017. developer.apple.com/videos/play/wwdc2017/818/

(Teacher Guide) Students will select a computing innovation and create an infographic, a video, or a podcast providing a description of the innovation and how a program is an integral part of the innovation. **[CRD-2.A] [CTP-1.A]**

Episode 1: The TV Club

Students follow the members of a high school film club as they anticipate the new season of their favorite show. They discover how the devices they use and the internet they connect to form a computing system, and learn how searching on the web and signing up for accounts relates to their personal information, and how to think about their privacy while using apps. **[IOC-2.A] [CSN-1.A] [CTP-5.A] [CTP-5.C]**

Unit 2: Algorithms

Timeframe: 6 weeks

Introduction

Students learn about the fundamentals of structured code. They explore how functions can encapsulate repetitive tasks, and use if/else statements to make decisions in their code. They also learn how Swift uses types to distinguish different kinds of data. They apply their skills in playgrounds to create song lyrics, a dancing BoogieBot, and a data visualizer. The culminating project is a QuestionBot app that responds to user input from the keyboard.

2.1 Get Started with Algorithms

In this lesson, students are introduced to the concept of an algorithm as a set of instructions. They learn that sequencing, selection, and iteration are three characteristics that algorithms share. Students learn that functions are a form of procedural abstraction. They also learn that multiple levels of abstraction can be used to provide modularity to a program and allow developers to focus on different portions of the code at once. Types are also a form of abstraction that group related attributes and behaviors of an object. Students are then introduced to the concept of parameters that can generalize the behavior of functions. Finally, conditional statements are introduced as a way of making decisions while a program is running.

Students will continue to explore bias in technology design, and analyze how bias might enter at all levels of software development. They'll explore how to design user experiences that extend beyond the app interface, and learn how to collect data about a user's experience, applying this to their QuestionBot app.

2.2 Play with Programs

2.2.A Functions

Students learn what makes functions so powerful, as they combine detailed steps into a definition that can be used again and again. Students learn how to define simple functions to create reusable groups of code.

Concepts covered

- Detail and complexity in code
- Abstraction
- Defining a simpler way to refer to something complex
- Handling complexity
- Concept of a function, calling, declaring
- Decomposition
- Writing functions to handle simple text repetition
- Nesting functions
- Infinite loops
- Modifying code more efficiently

Learning goals

Students will be able to:

- Describe why functions are an important concept.
- Define and implement basic functions.
- Recognize how abstraction can make code less complex, easier to read, and easier to debug.
- Use pseudocode to articulate a series of instructions.

Sample activities

Students will work in an Xcode playground to create functions to manage different portions of a nursery rhyme, and then create an original program that uses abstraction to efficiently perform a common song or meme. **[AAP-3.B] [CTP-3.C] [AAP-2.A] [CTP-2.A]**

(Teacher Guide) Students will write the steps for performing simple household tasks in terms of functions using pseudocode. **[AAP-2.A] [CTP-2.A]**

(Teacher Guide) In this activity, students will practice writing pseudocode for a Tic-Tac-Toe algorithm. They're introduced to the concept of a heuristic as a “best guess” technique that can be used when traditional methods of finding a solution are too slow. **[AAP-2.G] [CTP-2.A] [AAP-4.Ab] [CTP-1.D]**

2.2.B Types

Students become more familiar with the underpinnings of Swift by learning about the type system. They learn about the types that are included as part of the Swift standard library, and are introduced to the concept of creating custom types.

Concepts covered

- Exploring types
  - String, integer
  - Binary operator
  - Float, Double
- Type inference
- Type annotation
- Where do types come from?
  - Swift standard library
- Naming types
- Foundation framework
- Importing frameworks
- Working with the Date type

Learning goals

Students will be able to:

- Understand and explain a variety of types from the Swift standard library (for example, `String`, `Int`, `Double`).
- Identify cases where the Swift type system will infer a symbol’s type.

Sample activities

Students will write a program to calculate the circumference of a circle of given diameter, using an approximate value of pi. **[AAP-1.A] [CTP-3.A] [AAP-1.B] [CTP-4.B] [AAP-2.A] [CTP-2.A]**

Students will model a music collection using different types, and discuss various attributes that might be used to create a custom type. This provides an introduction to types that will be expanded upon in 3.2.A Instances, Methods, and Properties. **[AAP-1.Db] [CTP-3.C]**

Students will explore the limitations of built-in numeric types. **[DAT-1.A] [CTP-3.C]**

2.2.C Parameters and Results

Students expand their knowledge of functions by learning about parameters and return values to make functions more flexible and powerful. They learn how to define and call functions that accept parameters and return values when the function has completed.

Concepts covered

- Parameters and results
- Return values
- Simple functions that take values and return a string
- Increasing the power of functions as building blocks by using parameters and return values
- Generating simple texts using parameters and return values
- Good practices for creating clear functions and arguments

Learning goals

Students will be able to:

- Create a function that accepts and uses parameters.
- Create a function that returns a value.
- Use appropriate naming conventions to clearly describe functions and parameters.

Sample activities

Students will model real-world processes and tasks using functions with parameters and return values. **[AAP-3.B] [CTP-3.C]**

2.2.D Making Decisions

Students learn how to make decisions in code with conditional if/else statements, true or false values, and comparison operators.

Concepts covered

- Selection
- Boolean values
- Comparison operator == vs. assignment operator =
- Wrapping conditional code into a function
- Remainder operator (modulo)

Learning goals

Students will be able to:

- Compare values.
- Understand, explain, and use conditionals to write code that will run specific lines of code based on certain values.
- Use conditional statements correctly to evaluate expressions.
- Write if/else statements in the proper order to achieve a desired effect.

Sample activities

Students will work in an Xcode Playground to generate different print statements in response to different lengths of a video using an if statement and an if/else statement. **[AAP-2.H][CTP-2.B][CTP-4.B]**

Students will write a program that incorporates conditional statements and mathematical operations to determine if a given year is a leap year. **[AAP-1.A] [CTP-3.A] [AAP-1.B] [CTP-4.B] [AAP-2.H][CTP-2.B] [CTP-4.B] [AAP-2.F] [CTP-2.B]**

Students will play the FizzBuzz game, then write pseudocode that describes a set of conditional statements that represent the rules of the game. Finally, they will write, test, and debug their code in a new Xcode playground. **[AAP-2.B] [CTP-2.B] [AAP-2.C] [CTP-4.B]**

2.2.E BoogieBot

Students put their knowledge of functions to work by controlling an animated dancing robot within the playground. They'll also learn how to work with preexisting code via an API, or Application Programming Interface. In addition, they will sign their work and save an animated GIF of their work.

Concepts covered

- Use of Playground Live Views for developing animation and visuals
- Introduction to APIs
- Use of APIs to make an animated robot dance
- Benefits of functions
  - Reusable
  - Hide complexity
  - Facilitate program modification
- Save and share an animated GIF
- Concept of an algorithm
- Identify abstractions in everyday activities
- Use of pseudocode to articulate a series of instructions
- Reflection on functions, abstraction, and algorithms

Learning goals

Students will be able to:

- Use functions to build complex routines out of smaller parts.
- Describe how functions provided by other developers allow the students to build on their work.

Sample activities

Students will design a dance routine in pseudocode, then implement it in code to make an animated robot dance, troubleshooting and fixing problems as they go. **[CRD-2.F] [CTP-1.B] [AAP-1.D] [CTP-3.C]**

Students will write algorithms for everyday activities using pseudocode and identify opportunities to use abstraction to make the code less complex. **[AAP-2.A] [CTP-2.A]**

2.2.F Data Visualization

Students will plan and execute a data gathering process to answer questions about a problem that affects them. They'll process and visualize their data in a spreadsheet. **[DAT-2.D] [CTP-2.B] [DAT-2.E] [CTP-5.B]**

2.3 Build a QuestionBot App

Students will get experience modifying an existing Xcode project by writing new logic for QuestionBot. The project models a design in which the app logic is separated from the display. They'll learn how two or more people might collaborate on a single project, with each person or team developing a different part of an app. They'll also get additional practice with Xcode as they apply their learning to parse strings and experiment with their code in playgrounds before putting it into the project.

Concepts covered

- Implementation of simple question-answer conversation using the QuestionBot app
- Collaborative app development
- Use of methods to handle cases with strings
- Default answers
- Use of playgrounds to iteratively test code
- Model-View-Controller (MVC) programming paradigm

Learning goals

Students will be able to:

- Move code written in a playground into the right place in Xcode.
- Use conditionals to provide varied functionality based on user input.
- Create default answers to handle all input cases.

Sample activities

Students will model working as part of a collaborative team by creating a portion of a working app (the QuestionBot "brain") that interacts with already existing code. Students are asked to rewrite and modify existing functions, as well as create new functions to model desired behavior. Students follow an iterative development model by repeatedly testing and improving their functions in an Xcode playground before moving their code to an Xcode project file for use within the app. **[CRD-2.F] [CTP-1.B] [CRD-1.C] [CTP-1.C] [CTP-6.A] [AAP-2.M] [CTP-2.A]**

2.4 Design an Experience

Students learn about the importance of designing an inclusive data collection process, and making sure that their apps are designed for the widest range of users. They'll learn how to anticipate their audience's goals and design an app that meets those needs. They'll also gain understanding in designing user experience flows that allow them to test their users' experience in a systematic way.

Concepts covered

- User interface design
- Data collection
- Algorithmic bias
- Limitations of innovations
- User testing

Learning goals

Students will be able to:

- Map out a user experience flow using a flowchart.
- Describe specific goals a user might have for a given app.
- Describe any limitations of the app in meeting those goals.

Sample activities

Students will watch a video from WWDC to learn more about how Apple considers the overall user experience when designing apps that use machine learning. **[CRD-2.F] [CTP-1.B]**

Additional college-level resource: Designing Great ML Experiences. Apple Worldwide Developer Conference. 2019. https://developer.apple.com/videos/play/wwdc2019/803/

Students will choose an app that provides a delightful user experience, and identify the ways it takes in input and provides output by creating a visual that shows how they believe the model of the app works. **[CRD-2.C] [CRD-2.D] [CTP-3.A]**

Episode 2: The Viewing Party

Students continue the story of the TV club as they stream the episode while texting each other. They open a window into how data is represented inside their devices at the lowest level, and how it flows across the internet. They also learn more about security and privacy of data.

Students design a simple website prototype that explains in simple terms how computing resources can be protected, and what users should look for to know their data is protected. **[IOC-2.B] [CTP-5.E]**

Students participate in an activity in which they form a decentralized network by sharing interests on index cards to find connections with other students in the classroom, and route the cards around students who are sitting down—a process that mimics the way that the internet has been engineered to transmit data as a fault-tolerant system. **[CSN-1.E] [CTP-1.D]**

Students will read about the structure of the internet and the protocols that undergird it, then create a short Keynote presentation that illustrates one aspect of how the internet works. For example, students might create an animation that shows how web pages listen for incoming packets and send back the requested information using TCP/IP, or they might create a series of slides that show what elements can affect the lag on a server. **[CSN-1.B] [CTP-5.A]**

Students will create a model of a binary cash register and use it to convert decimal numbers to binary numbers and binary numbers to decimal numbers. They'll then explore the Programmer's Calculator app to discover the connections between binary, decimal, hexadecimal, and ASCII. **[DAT-1.A][DAT-1.C][CTP-2.B] [CTP-3.C]**

Unit 3: Organizing Data

Timeframe: 6 weeks

Introduction

Students learn skills for organizing and processing data. They explore how to create custom types using structs, group large quantities of items into arrays, and process them using loops. They also learn how to use enums to represent a set of related values. They apply their knowledge by extending their data visualizer and creating an algorithm to determine password security. In the app project at the end of the unit, they build an interactive app with colorful shapes.

3.1 Get Started with Organizing Data

Students learn that Types are a form of abstraction. They experiment with properties and methods, and learn that instances of types support various attributes and behaviors. Students are introduced to the concept of arrays, which are collections of data that are structured in an orderly way to facilitate storage and retrieval. Students learn that loops, or iteration, can be used to access the elements of an array. Finally, students are introduced to the idea of undecidable problems in computer science.

Additional college-level resource: Three Problems Computers Will Never Be Able to Solve. (2016, January 19). Retrieved January 21, 2020, from www.youtube.com/embed/Hex2hqPvOQ8 **[AAP-4.B] [CTP-1.A]**

3.2 Play with Complex Data

3.2.A Instances, Methods, and Properties

Students build on their knowledge of types by learning about the methods and properties that make up an instance of that type. Students learn the difference between a type and an instance, and learn how to use the Xcode documentation viewer to find information about unfamiliar types, properties, or methods.

Concepts covered

- Instances, methods, and properties
- Initializing a type
- Type safety
- Methods on strings—`hasPrefix()`, `hasSuffix()`
- APIs—finding out more about the API that a type offers, using documentation
- Introduction to classes and structs

Learning goals

Students will be able to:

- Create values without using literals.
- Describe the difference between a type and an instance.
- Describe the difference between a method and a property.
- Describe how type safety rules apply to methods and properties.
- Use documentation to find out information about unfamiliar types.

Sample activities

Students will create multiple instances of the BoogieBot robot for a dance competition and visually assess the correctness of their algorithms, making adjustments as necessary. **[CRD-2.I] [CTP-4.C]**

Students will create an instance of a predefined type and call existing methods to calculate how much weight can be lifted to a treehouse. **[AAP-1.Db] [CTP-3.C]**

Students will consult API documentation to learn more about properties and functions supported by the `Integer`, `Date`, and `URL` types. **[AAP-3.D] [CTP-2.B] [AAP-3.Ab] [CTP-4.B]**

3.2.B Arrays and Loops

Arrays are a common way to group objects in an ordered list. Students will learn how to create and work with arrays by adding and removing objects. Students will also learn about for-loops to work with each object in an array.

Concepts covered

- Arrays
- Looping over an array
- Mutable and immutable arrays
- Operations on arrays
- Index out of bounds error
- Data privacy

Learning goals

Students will be able to:

- Understand and explain common use cases for arrays.
- Use arrays to manage collections of objects.
- Iterate through an array to perform a common action on each item in the collection.
- Understand the real-world implications of being able to efficiently analyze large amounts of personal data.

Sample activities

Students will write programs to handle modifications to a karaoke song list, search an array of messages for those that contain a particular name, and tally votes in an array. **[AAP-2.Ka] [AAP-2.O] [CTP-2.B]**

Students will write a function that takes a number as an argument, traverses an array, and returns different messages as a string. **[AAP-2.N] [CTP-2.B]**

Students will look at `map()` and `filter()` functions for working with collections. **[AAP-3.D] [CTP-2.B]**

(Teacher Guide) Students will learn about polynomial algorithms within the context of various sorting algorithms such as selection sort. **[AAP-4.A] [CTP-1.D]**

3.2.C Structures

Students will recognize that it is often useful to group related information and functionality into a custom type. Students learn how to define a custom type by using a structure with accompanying properties and methods. Students will practice this by creating a custom Song type that has related properties to define the title, artist, and duration of each instance. They'll also work with a Rectangle type with methods to compare sizes.

Concepts covered

- Data model
- Defining structures
- Grouping related information and functionality into a custom type
- Calculated properties
- Defining instance methods
- Unsolvable problems and undecidability

Learning goals

Students will be able to:

- Understand and explain the importance of custom structures and their common use cases.
- Design a custom structure to group related data into one type.
- Define custom properties and methods for custom structures.
- Understand that some problems cannot be solved algorithmically.
- Understand that some problems are undecidable.

Sample activities

Students will add a new calculated property for the area of a rectangle to an existing type in order to simplify one of its methods. **[AAP-3.C] [CTP-3.B]**

Students will model a real-world object as a custom type. **[DAT-1.A] [CTP-3.C]**

3.2.D Enums and Switch

Enumerations, or enums, are a way to define a named list of options. Students will learn what enums are used for, how to define them, and common ways to work with them. Students will learn to use the switch statement to conditionally run blocks of code based on the value of an enum instance.

Concepts covered

- Defining and using enums
- Working with a named list of options
- Selection with switch and enum values

Learning goals

Students will be able to:

- Understand and explain the benefits and proper use cases of enumerations.
- Use a switch statement on an enum value to conditionally trigger specific code.

Sample activities

Students will write a loop that traverses an array, using enums to keep an accurate count of types with certain properties. **[AAP-2.N] [CTP-2.B] [DAT-1.A] [CTP-3.C]**

Students will modify a struct to use an enum instead of a boolean value to keep track of the score of a target game. **[DAT-1.A] [CTP-3.C]**

Students will write a new function to analyze the data in an array. **[AAP-2.O] [CTP-2.B] [AAP-3.A] [CTP-3.B]**

3.2.E Testing Code

Students will learn various strategies for testing their functions to make sure they run properly. They'll also learn how to choose specific input values, and associated outputs, to verify that their code does what it is intended to do.

Concepts covered

- Limits of specific data types
- Testing limits of functions using test cases
- Using boolean expressions in testing
- Floating point imprecision

Learning goals

Students will be able to:

- Identify and fix overflow errors.
- Identify and fix errors caused by floating point imprecision.
- Develop specific inputs and expected outputs to test the limits of a function.

Sample activities

Students will identify and fix an integer overflow error. **[CRD-2.I] [CTP-4.C]**

Students are introduced to a PiggyBank whose functions are intended to operate within certain limits. They'll create test cases that test the limits of that function and then verify the expected output. **[CRD-2.J] [CTP-1.B]**

Students will experiment with floating point imprecision by performing operations using Doubles, then learn a strategy for multiplying and rounding those variables to maintain precision. **[DAT-1.B] [CTP-1.D]**

3.2.F Processing Data

Students will learn various strategies for cleaning and combining data to get a complete picture of what they're trying to visualize. They'll also transform data by putting it into new contexts or deriving extra information from it. In addition, students will learn how to incorporate and cite third-party code.

Concepts covered

- Cleaning data by discarding errors
- Cleaning data by correcting errors
- Citing code written by a third party
- Calculating statistical data

Learning goals

Students will be able to:

- Identify and discard data errors.
- Identify and fix data errors.
- Cite and use a third-party algorithm to identify close matches.

Sample activities

Students will write an algorithm to process survey data using iteration. **[AAP-2.Ka] [CTP-2.B]**

Students will write an algorithm to process survey data using selection to identify invalid show names. **[AAP-2.Ha] [DAT-2.D] [CTP-2.B]**

Students will perform a statistical analysis of their data by calculating the median and the median absolute deviation, and create groupings that provide insight and knowledge about the data. **[DAT-2.E] [CTP-5.B]**

Students will experiment with an edit distance algorithm to catch spelling errors. They'll learn the proper way to cite third-party code. **[IOC-1.F] [CTP-5.E]**

3.2.G Pixel Art

Students will create low-resolution graphics by turning individual pixels on and off using a series of prewritten functions.

Concepts covered

- Using functions with parameters and structs
- Use of abstraction to combine functions
- Using code as a form of creative expression

Learning goals

Students will be able to:

- Use functions that take specific parameters.
- Use the coordinate system to specify pixels on a two-dimensional plane.
- Create simple multi-frame animations.

Sample activities

Students will combine functions to create simple shapes and fill them with colors, or draw a series of simple shapes at various coordinates on the screen. **[AAP-3.D] [CTP-2.B] [AAP-3.Ab] [CTP-4.B]**

Students will use built-in functions to create simple animations. **[DAT-1.A] [CTP-3.C]**

3.2.H Password Security

Students will explore the effect that different combinations of characters have on the security of a password. They will also discover the point at which the runtime of an algorithm to guess a password becomes unreasonable.

Concepts covered

- Reasonable and unreasonable runtime
- Strong passwords

Learning goals

Students will be able to:

- Identify an algorithm that runs in an unreasonable time.
- Recognize factors that contribute to the strength of a password.

Sample activities

Students will devise a program to determine whether or not a password is secure. **[AAP-2.M] [CTP-2.A] [IOC-2.B] [CTP-5.E]**

Students will explore the privacy and security implications of weak passwords. They'll learn the relationship between the length of a password and a brute force algorithm running in a reasonable amount of time and an unreasonable amount of time. **[AAP-4.A] [CTP-1.D]**

3.2.H Visualization Revisited

Students will explore the functionality provided by an API that creates and modifies pie charts, bar graphs, and plots. They'll discover different ways to visualize data they collected in an earlier lesson.

Concepts covered

- Using external functions to display data
- Using parameters to generalize the behavior of functions

Learning goals

Students will be able to:

- Call functions with multiple parameters.
- Display data using a pie chart.
- Display data using a bar graph.
- Display data using a scatter plot.

Sample activities

Students will change the values of type properties and class properties to change how the visualization displays data. **[DAT-1.A] [CTP-3.C]**

3.3 Build a BouncyBall App

In this project, students will build a physics-based game that uses touch interactions—a much more complex app than those in their previous projects. They already have most of the basic skills necessary to complete this app. Students will be introduced to the incremental development process, building the app in discrete phases and testing as they go. They'll also learn how to refactor their code to ensure that it remains readable and flexible as it grows in complexity. The primary new coding concept in this lesson is the *callbacks* they use to manage code that runs in response to events such as collisions and touch interactions.

Concepts covered

- Callbacks
- Function types
- Refactoring
- Incremental development

Learning goals

Students will be able to:

- Follow the incremental development process to build an app in small stages, testing their code after each step.
- Refactor their code to keep it readable and well organized.
- Describe how callbacks enable code to be triggered by events.
- Use functions as callbacks.
- Use a physics-based game API to display interactive shapes on the screen.

3.4 Design a Prototype

Students are introduced to a development process for designing and developing an app. They'll learn how to integrate their audience into the development cycle. They'll make a plan for adding functionality and increasing the usability of their app within a specific timeframe.

Students will work in small teams to explore applications of machine learning that relate to other fields, such as medicine, business, ecology, and the arts. They'll collate their findings in a collaborative Keynote presentation to be shared with the class. **[CRD-1.C] [CTP-1.C]**

Episode 3: Sharing Photos

Students complete the story of the TV club as they share pictures of the viewing party on social media. They explore how privacy can be affected by sharing data online, along with other unanticipated consequences. They also look at how images are captured and processed by a mobile device and how a social media service handles posts.

Students will participate in an activity that involves sending a 10 x 10 1-bit image from one student to another using only their voice. This introduces students to the concepts of data transmission, error checking, and compression. **[DAT-1.C] [CTP-2.B]**

Students will learn about crowdsourcing and how the internet enables new ways of solving problems. They'll explore the concepts of open source software and solving problems with distributed data. **[IOC-1.E] [CTP-1.C]**

Unit 4: Building Apps

Timeframe: 10 weeks

Introduction

Students deepen their skills in Xcode and interface Builder in guided projects. These guides show students how to build an app from the ground up using techniques to add user interface elements to a screen, connect those elements to their code, and respond to the events generated by user interaction. They also experience the iterative development of an app by gradually expanding its capabilities, testing it at each phase. They apply their knowledge to create a Meme Maker app. The culmination of the unit is a quiz app.

4.1 Get Started with App Development

Students are introduced to the concept of building apps that accept input and provide output. Actions and outlets are introduced as a way for apps to receive input in the form of events. The incremental development process is reinforced as an effective way to debug and optimize code in segments. Debugging and testing is covered in greater depth, with specific strategies for finding and fixing errors in code, from visualization in the debugger to hand tracing.

4.2 Play with App Components

4.2.A Color Picker

Students will learn how to build user interfaces in the Xcode Interface Builder tools, and tie user interface elements into code via actions and outlets. Students will practice creating outlets to access properties of a user interface view, and actions to respond to user interaction with buttons and other controls as they build a color picker app. They'll get more practice using the incremental development process to build the app in small phases.

Concepts covered

- Building a Color Picker app
- Connecting the app interface to Swift
- Using buttons, switches, and sliders
- Testing the app at checkpoints
- Working with color in the display
- Working with `CGFLoat`

Learning goals

Students will be able to:

- Explain actions and use them to run code when the user taps the user interface.
- Explain outlets and use them to connect objects in the storyboard to code.
- Understand, explain, and use a variety of `UIControl` events such as switches and sliders.
- Understand that `UIControl` elements are instances of abstraction that provide a generalized way to create multiple interface designs.

Sample activities

Students will create a new project with a single button to change background color depending on a control event. The activity introduces using instances of UI library objects that generalize functionality through the customization of their various attributes in Xcode. **[AAP-3.D] [CTP-2.B]**

4.2.B ChatBot

Students expand on the QuestionBot application by building ChatBot, an app that displays the history of the conversation. Students learn about the data source pattern, and build a simple data source object to provide information on Message objects to display in the message list view. Students practice appending to an array to store messages on the data source object to maintain a history of the conversation.

Concepts covered

- Using the ChatBot app to display the conversation history of a QuestionBot
- Analyzing incoming data to an app
- Tracking the flow of incoming messages and display them as output
- Deploying an app to a real iOS device

Learning goals

Students will be able to:

- Understand and explain the data source pattern.
- Understand and explain data source objects and their relationship to other objects in an application.
- Implement a simple data source object that communicates data to and from the user interface layer of an application.
- Relate the ChatBot data source model to the way packets are transmitted over the internet.

Sample activities

Students will traverse an array and perform list operations to manage the stored data. **[AAP-2.N] [AAP-2Oa] [CTP-2.B]**

Students will create an app that gives original answers to questions that the user poses. They'll then take the app they have coded in Xcode and deploy it to a real iPhone or iPad. **[AAP-2.Ha] [CTP-2.B]**

4.2.C Rock, Paper, Scissors

Students will complete a Rock, Paper, Scissors game from scratch. Students will review a variety of concepts covered in the course and build the user interface, the model data, and the controller objects that make up the entire application.

Concepts covered

- Building a Rock, Paper, Scissors game app
  - Analyzing game states
  - Setting up signs and icons
  - Using an algorithm for gameplay
  - Using an API for generating random numbers to control app moves
  - Designing and connecting an interface

Learning goals

Students will be able to:

- Integrate all of the concepts from the course into a basic Rock, Paper, Scissors app.

Sample activities

Students will write the algorithms and create the assets to complete the Rock, Paper, Scissors app. **[CRD-2.C] [CRD-2.D] [CTP-3.A] [CRD-2.I] [CTP-4.C] [AAP-1.A] [AAP-2.B] [CTP-2.B] [AAP-2.M] [CTP-2.A] [CTP-2.B]**

4.2.D MemeMaker

Students will complete a Meme Maker app from scratch. Students will review a variety of concepts covered in the course and build the user interface, the model data, and the controller objects that make up the entire application.

Concepts covered

- Building a Meme Maker app
  - Adding assets to a project
  - Designing an interface
  - Using segmented controls
  - Defining a structure to represent possible choices

Learning goals

Students will be able to:

- Integrate all of the concepts from the course into an app that displays text over photos.

Sample activities

Students will design the interface, structure, and assets to complete the Meme Maker app. **[CRD-2.C] [CRD-2.D] [CTP-3.A] [CRD-2.I] [CTP-4.C] [AAP-1.A] [AAP-2.B] [CTP-2.B] [AAP-2.M] [CTP-2.A] [CTP-2.B]**

4.3 Build an ElementQuiz App

In this final project, students will complete an app for studying and quizzing with the periodic table of elements. They'll apply the skills they've acquired throughout the course, using the incremental development process as they build the app from a simple interface into a full-featured app with two modes. They'll encounter subtle UI bugs in their app and learn how to describe exactly how to reproduce them, and how that enables them to fix bugs. They'll learn about how to handle keyboard input and use the standard iOS alert dialog to present the user's score.

Concepts covered

- Adding images to an app
- Planning and creating the data model of the app
- Index out of range error
- Refactoring
- Single-path UI updates
- iOS keyboard and text fields

Learning goals

Students will be able to:

- Follow the incremental development process over the course of a complex app
- Track the app state with multiple properties
- Use the single-path UI update pattern to modify the user interface as the app's state changes
- Handle keyboard input with a text field
- Identify the detailed steps to reproduce a bug, and use them to fix it
- Display an alert and handle its completion callback

Sample activities

Students will complete an app that displays chemical elements. **[AAP-1.A] [AAP-2.B] [CTP-2.B]**

4.4 Design for Impact

Students will compare their apps with similar existing apps and differentiate their functionality. They'll try to predict unintended consequences that may arise from the use of their app, particularly at scale. Students will envision their model users and develop a pitch that will appeal to those users. They'll try to formulate a plan for funding their app and making it financially sustainable. Finally, students are introduced to the concept of the digital divide, and they'll research different technological innovations and how they may have helped to bridge that gap.

Sample activities

Students are asked to choose a social media app and identify the intended and unintended consequences of that app. **[IOC-1.A] [CTP-5.C]**

Students are asked to consider the ethical implications of collecting and storing user data for the purpose of targeted advertising. **[IOC-1.F] [CTP-1.E]**

Students visit the OECD website to research varying levels of internet access between different countries, as well as disparities in resources and socioeconomic factors that could contribute to the digital divide. **[IOC-1.C] [CTP-5C]**

Create Performance Task

Students will be provided with 12 hours of in-class time to create their own project applying the skills they have learned so far, and to complete the write-up and video as specified in the Create Performance Task assignment. Students may choose to focus on one feature of the app they have been designing and prototyping up to this point, or they can modify one of the app templates that have been provided as starter code, adding in their own functions as necessary to meet the requirements of the task, which include: the use of a list of a collection to manage data, and an example of a student-developed procedure that's called with different parameters, resulting in different behavior.