Over 35 Years Of Technology Training

Document Generated: 12/11/2025

Learning Style: Virtual Classroom

Technology:

Difficulty: Beginner

Course Duration: 3 Days

Swift Programming Essentials (TT4725)

About This Course:

This hands-on, instructor-led course provides a solid foundation in the Swift programming language, focusing on core programming concepts and best practices for building iOS and macOS applications. Whether you're a first-time developer or transitioning from another platform, this course will guide you through Swift's

modern syntax, type safety, control flow, and fundamental data structures.

The course emphasizes real-world coding practices, using Xcode Playgrounds to support immediate feedback and learning. Through practical exercises and guided demos, students will build confidence in writing clean, efficient Swift code that aligns with current development standards.

Course Objectives:

- Navigate and use Xcode Playgrounds for Swift development
- Work with variables, types, control flow, loops, and functions
- Understand and apply optionals, closures, structs, and classes
- Build foundational knowledge of Swift collections (arrays, dictionaries, sets)
- Explore key Swift concepts such as value vs. reference types, initializers, and enumerations
- Apply functional programming techniques using closures and higher-order

Audience:

- New and aspiring iOS/macOS developers
- Developers transitioning from other programming languages
- Anyone looking to build a strong foundation in Swift programming

Prerequisites:

- This course is designed for individuals with some prior exposure to programming concepts, such as variables, conditionals, loops, and functions.
- You do not need Swift or iOS/macOS experience, but familiarity with any programming language (e.g., Python, Java, JavaScript, etc.) will help you succeed and get the most from the hands-on labs.

Course Outline:

1. Getting Started

- Getting Started with Xcode
- Playing in a Playground
- Running Your Code
- Troubleshooting Playgrounds
- Varying Variables and Printing to the Console
- Adding Comments

2. Types, Constants, and Variables

- Types
- Type inference and type safety
- Constants vs Variables
- Operators
- String Interpolation

3. Conditionals

- if/else
- Ternary Operator
- Nested ifs
- else if

4. Numbers

- Integers
- Creating Integer Instances
- Operations on Integers
- Integer division
- Operator shorthand

- Overflow operators
- Converting Between Integer Types
- Floating-Point Numbers
- Ranges of Numbers

5. Switch

- Switch Syntax
- Ranges
- Value binding
- where clauses
- Tuples and Pattern Matching
- switch vs if/else

6. Loops

- Range operators
- for-in Loops
- where
- while Loops
- repeat-while Loops
- Control Transfer Statements in Loops

7. Strings

- Working with Strings
- Characters
- Unicode
- Unicode scalars
- Canonical equivalence

- Substrings
- Multiline Strings

8. Arrays

- Creating an Array
- Accessing and Modifying Arrays
- Combining Arrays
- Array Equality
- Immutable Arrays
- Iterating over an array
- Documentation

9. Optionals

- Optional Types
- Optional Binding
- Implicitly Unwrapped Optionals
- Optional Chaining
- Modifying an Optional in Place
- The Nil Coalescing Operator

10. Dictionaries

- Creating a Dictionary
- Accessing and Modifying Values
- Adding and Removing Values
- Looping over a Dictionary
- Immutable Dictionaries
- Translating a Dictionary to an Array

11. Sets

- What Is a Set?
- Getting a Set
- Working with Sets
- Unions
- Intersections
- Disjoint
- symmetricDifference
- Moving Between Types

12. Functions

- A Basic Function
- Function Parameters
- Parameter names
- custom labels
- Default parameter values
- In-out parameters
- Returning from a Function
- Nested Function Definitions and Scope
- Multiple Returns
- Optional Return Types
- Exiting Early from a Function
- Using a guard statement to exit a function early
- Function Types
- Void

- Variadic Parameters
- Using functions as return types
- Using functions as parameters

13. Closures
    - Closure Syntax
    - Simplifying closures
    - Closure Expression Syntax
    - Functions as Arguments
    - Closures Capture Their Enclosing Scope
    - Functional Programming
    - Higher-Order Functions
    - map(_:)
    - filter(_:)
    - reduce(_:_:)

14. Enumerations
    - Basic Enumerations
    - Enumerations with Raw Values
    - Methods
    - Associated Values

15. Structs and Classes
    - A New Project
    - Structures
    - Instance Methods
    - Mutating methods

- Comparing value types and reference types – Overview- covered later in detail
- Classes
- Making an instance of the class
- Inheritance
- Making a subclass
- Overriding a superclass method
- Looking Ahead: What Is the Real Difference?
- Deciding between classes and structures

16. Properties

- Basic Stored Properties
- Nested Types
- Lazy Stored Properties
- Computed Properties
- A getter and a setter
- Property Observers
- Type Properties
- Access Control
- Controlling getter and setter visibility

17. Initialization

- Initializer Syntax
- Struct Initialization
- Default initializers for structs
- Custom initializers for structs
- Class Initialization

- Default initializers for classes
- Initialization and class inheritance
- Required initializers for classes
- Deinitialization
- Failable Initializers

18. Value vs Reference Types

- Value Semantics
- Reference Semantics
- Constant Value and Reference Types
- Using Value and Reference Types Together
- Copying
- Equality vs Identity
- What Should I Use?

Additional Topics (Time Permitting)

The following chapters are included for extended coverage and may be addressed as time allows. Their inclusion depends on class pacing, participant engagement, and time availability.

19. Protocols

- Formatting a Table of Data
- Protocols
- Protocol Conformance
- Protocol Inheritance
- Protocols as Types
- Protocol Composition
- Mutating Methods

20. Extensions

- Extending an Existing Type
- Extending Your Own Type
- Using extensions to add protocol conformance
- Adopting a protocol via an extension
- Adding an initializer with an extension
- Nested types and extensions
- Extensions with methods
- Creating an array of different types of objects

21. Generics

- Generic Data Structures
- Generic Functions and Methods
- Type Constraints
- Associated Types
- Type Constraints in where Clauses
- Generic Composition and Opaque Types

22. Protocol Extensions

- Modeling Exercise
- Extending Exercise
- Self Types and Type Values
- Protocol Extension where Clauses
- Default Implementations with Protocol Extensions
- Implementation Conflicts

23. Error Handling

- Classes of Errors
- Lexing an Input String
- Catching Errors
- Parsing the Token Array
- Handling Errors by Sticking Your Head in the Sand
- Swift Error-Handling Philosophy