**Challenge 3: Get Your Hands Dirty**

**Section 1 - Getting Started**

**1.1 Installing the tools 4m 42s**

**Install Xcode and register as an Apple Developer.**

**1.2 Creating your first application 11m 28s**

**Create your first application using the same steps Simon describes in the video. Familiarizeyourself with the Xcode environment, specifically notice how it can be manipulated to display different helper tools and how it will attempt to fill in your code as you type it.**

**1.3 Updates to this course 3m 31s**

**Why do you think it's important to be aware of the idiosyncracies with older versions of Objective-C and to keep up with new features as they are added?**

*A:” It is really important to be aware of the history, because some other programs might have been written in older versions, as well as any other programmers might still use older version. But despite on which one do you use you have to be aware of new adding as well, that’s going to make your life simplier. For instance , a great addition of ARC, which makes to write a code easier. “*

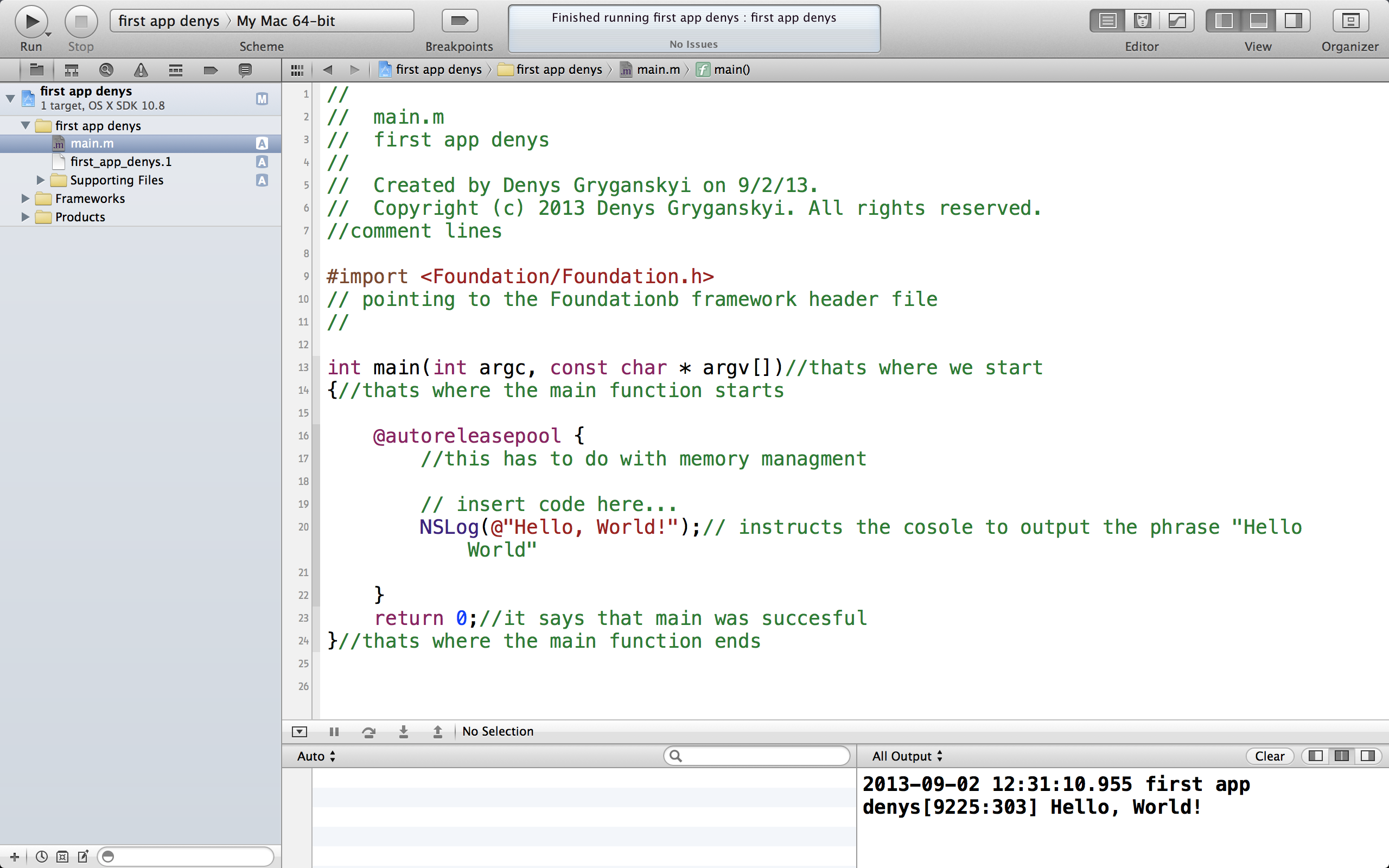
**Section 2 - Objective-C Basics**

**2.1 The Objective-C language 4m 11s**

**How did Objective-C become the language to learn if you want to make apps for the iPhone** **and iPad?**

*A:” Objective- C was not designed from the ground up as a new language. It’s a simply the C program languages with staff added to it, packed on top of it. In 1988 Objective- C was adopted as a main language of Next Computer, which Steve Jobs founded after he left Apple in 1985. And they build everything including their Next STEP operating system on Objective- C (in 1989), but in 1986 Apple bought Next and they roll over to the NEXTSTEP operating system into what become Mac OS in 2001. That’s why we use NS all over the code. So Mac OS x is built on Objective- C. The IPhone which came out in 2007, its OS is built on Mac OS X. Ipad, released in 2010, also build on this history. It is not arbitrary choice, its historical necessity, its woven through it at every level.”*

**2.2 The structure of an Objective-C program**

**Create a new project. Go to the menu option Xcode/Preferences/Text\_Editing and make sure "Line Numbers" is checked in the section marked "Show." Then add comments describing the purpose of each auto-generated line in the main.m file. For example on Line 17 I would write:" NSLog(@"Hello, World!"); //instructs the console to output the phrase "Hello,World!")"**

**2.3 Compiling and running your code 8m 37s**

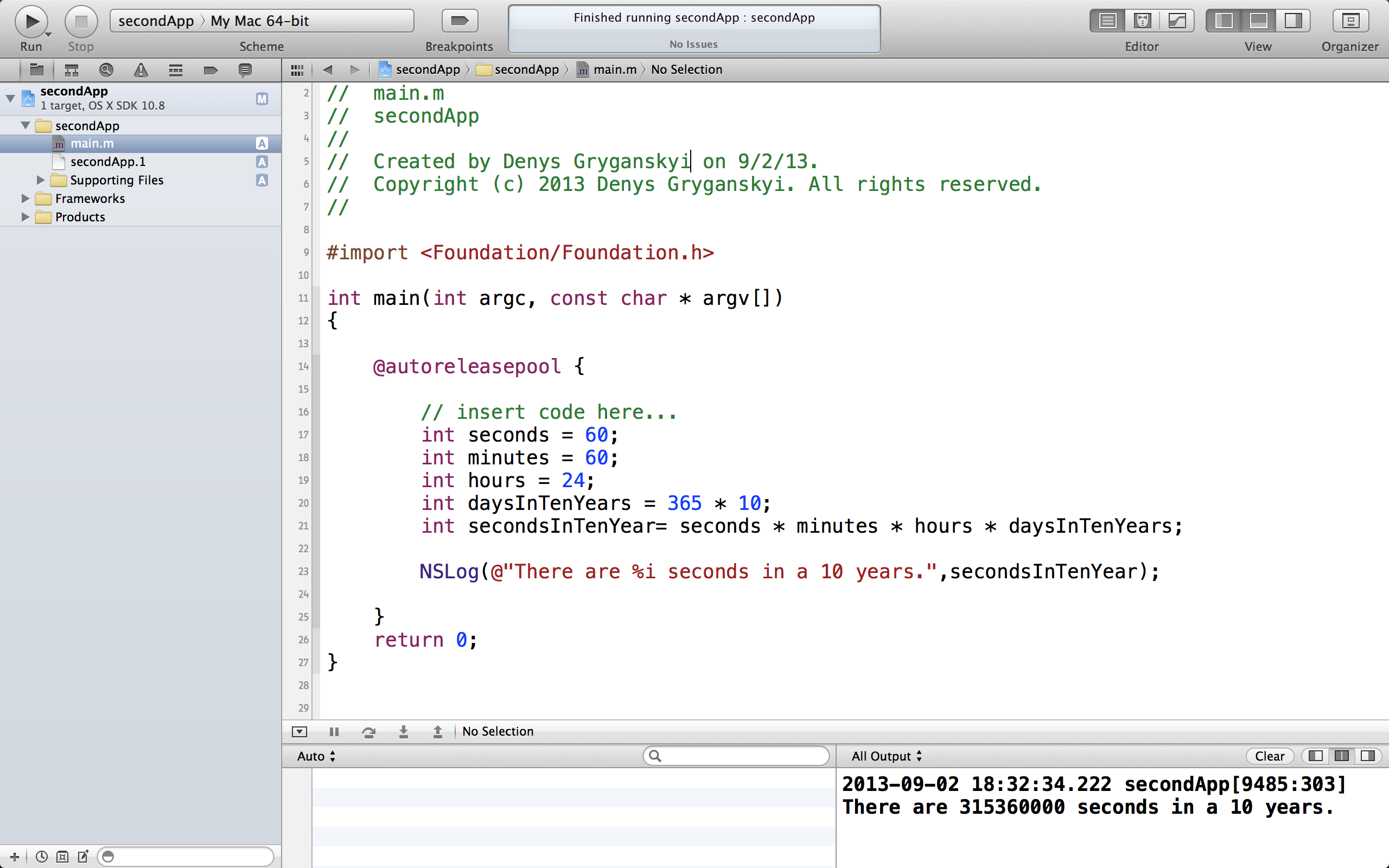
**Q;”Why might you build in one version of iOS but deploy in an older version?**

***A:”*** *Deployment Targets are a powerful concept, that allow you to build applications that take advantage of new features in the latest platform SDK, but can still run on older versions of the operating system. For example, you might want to build your application against the new iOS 7 SDK to take advantage of all the new capabilities, but set a Deployment Target of 6.0 or 5.0 to let users with older devices still use your application – albeit maybe with reduced functionality. Which SDK(s) to build for is controlled by two settings: the* ***Target SDK*** *and the* ***Deployment Target****.*

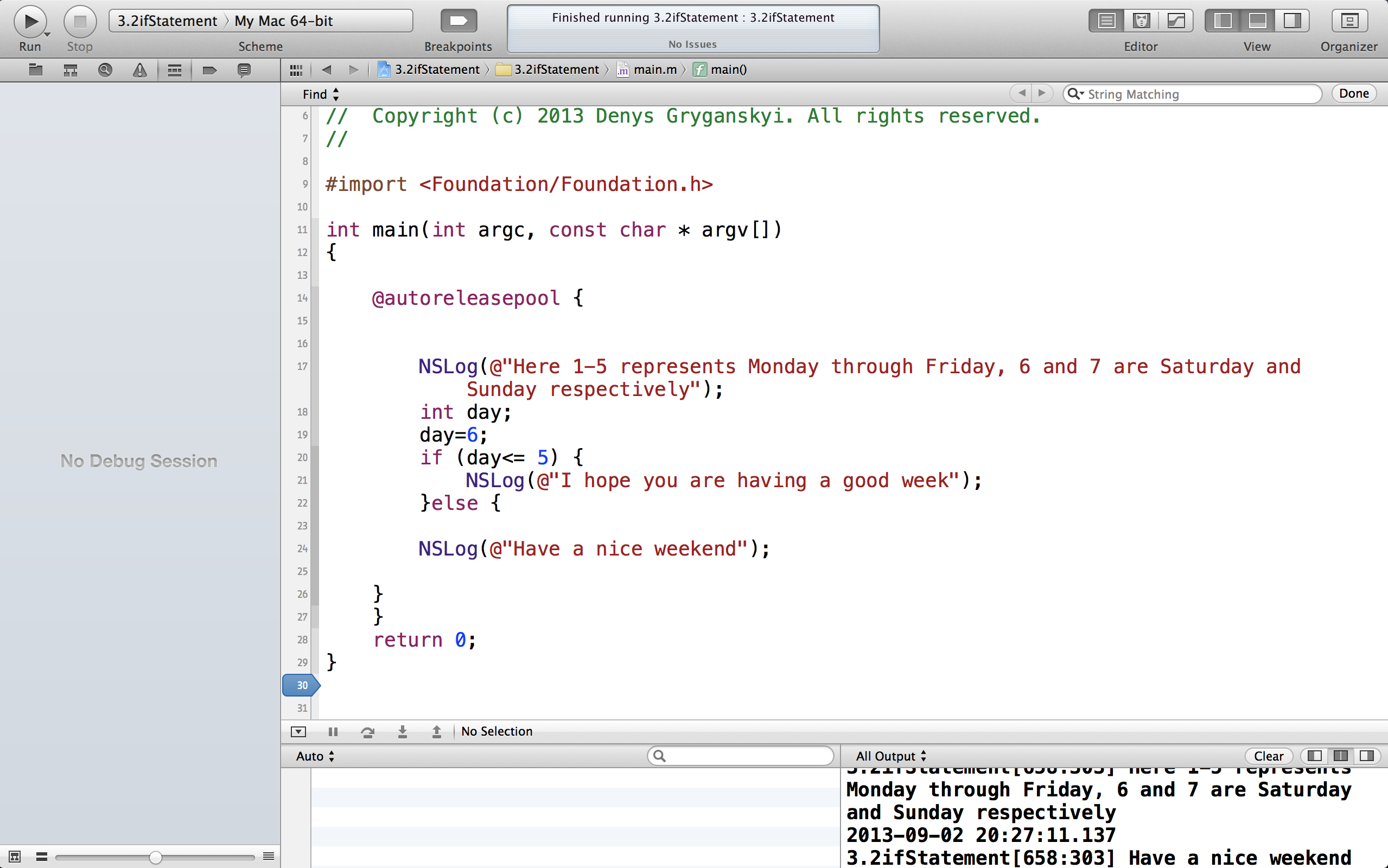
*The* ***Target SDK*** *is the main setting, and tells compiler and linker what version of the SDK you want to build against. The* ***Deployment Target*** *setting does not directly affect what the compiler sees. All it does is to mark your application as being OK to run on versions of the operating system that may be older than the Target SDK. So you might have an up- to- the- minute system, but want to allow your user go back older version.* ***”***

**Section 3 - Program Flow**

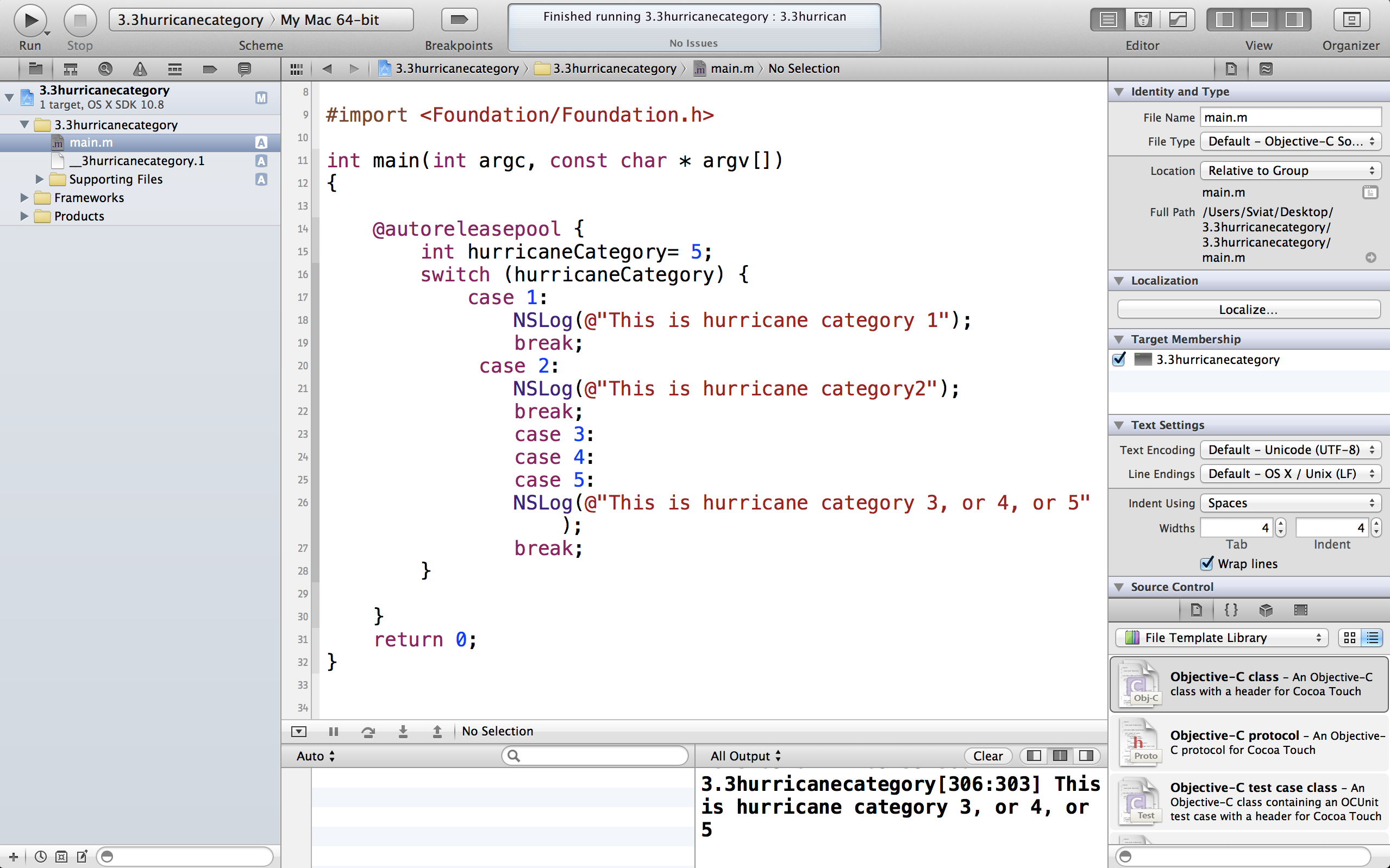
**3.1 Logging messages to the command line 6m 7s**

**Following the example in the video, write a program that calculates and outputs to the console the number of seconds in ten years. Copy and paste your code here.**

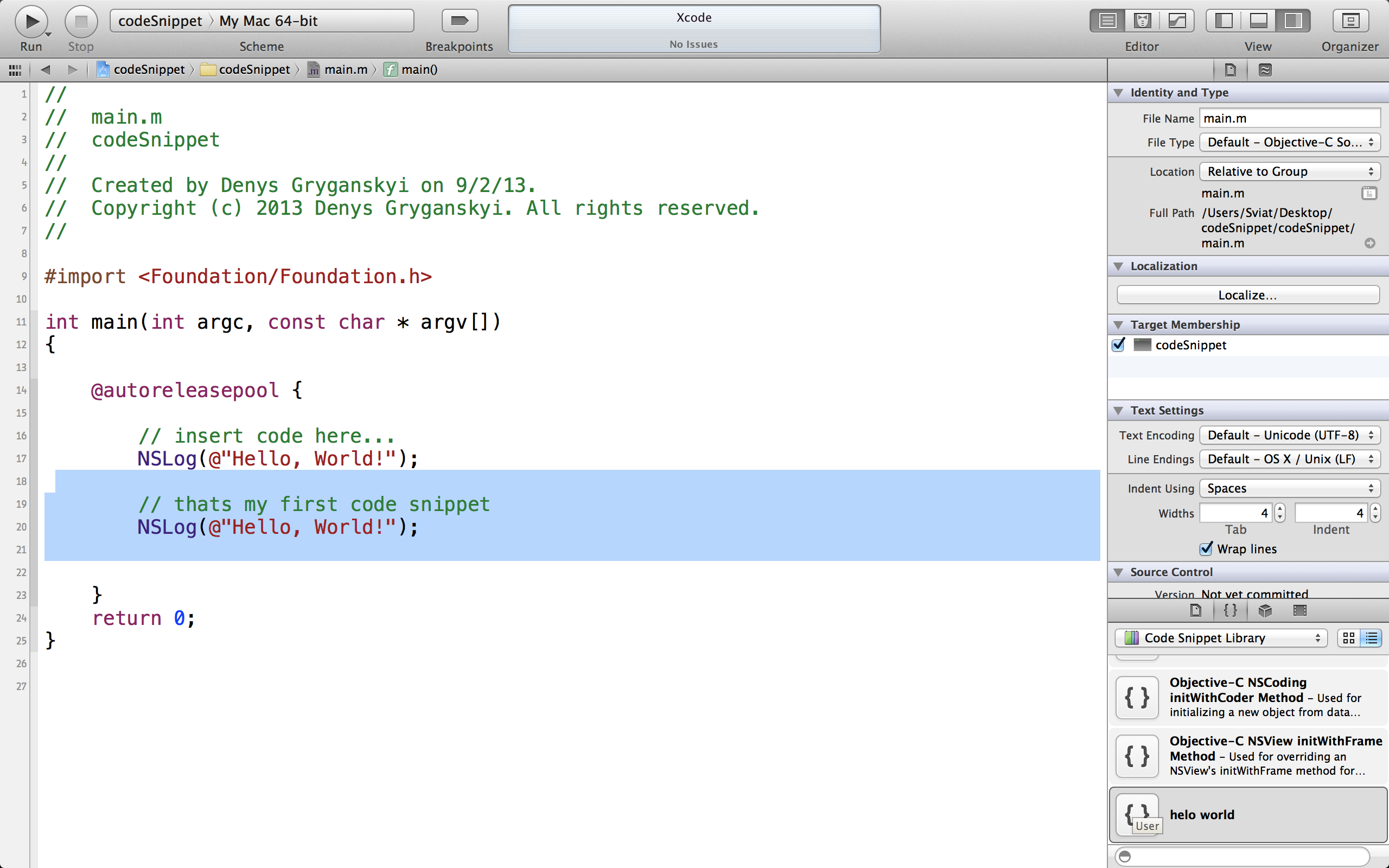
**3.2 Writing conditional code 7m 1s**

**Using Objective-C, create an integer variable called "day" that represents the days of theweek. Write an if statement that checks whether "day" is a weekend day. If the day is a weekend day then have your program print a message saying "Have a nice weekend!" and if it's not, print a message saying "I hope you're having a good week!"**

**3.3 The switch statement**

**Create a variable called "hurricaneCategory" and a switch statement that prints out a message describing a hurricane's category from 1-5.**

**3.4 Code snippets**

**Grab a code snippet, indent it to match the indent of your project, then add comments to it, then select the entire snippet you just modified and save it as your own code snippet. Time yourself and record how many seconds it takes you to do all this.**

**3.5 Operators and expressions**

**List the 6 types of operators described in this video. Provide their name, a description of their meaning, and the syntax you would use to execute them. What code snippet does the ternary operator replace?**

*A:” \*Arithmetic operators:*

*~ “+” addition*

*~ “-” subtraction*

*~ “\*” multiplication*

*~ “/” division*

*These operators have precedence in this order \*, /, +, - which means they are going to be executed in the same order;*

*\*Comparison operators:*

*~ “==” equal to*

*~”!=” not equal to*

*~ “>” more than*

*~ “<” less than*

*~ “>=” more or equal to*

*~ “<=” less or equal to*

*\*Logical operators*

*~ “&&” and*

*~ “||” or;*

*\*Increment or decrement operators*

*~ “i++” increment*

*~ “i--” decrement*

*\*Modulus to calculate reminder*

*\*Ternary*

*~ “condition? true: false ”-will result in one single value. This is like an “if-else statement”, so you can find as an analog in code snippet.”*

**3.6 Loops**

**CHALLENGE: Think of a scenario while using a mobile app that might require you to use a "continue" statement in the middle of a loop.**

*A:” It might be an feedback application, when customer not willing to answer, program will continue to run.”*

**3.7 Functions**

**What is a function? What is a function prototype? What are the purposes of each? What are the rules for when and how you can call a function?**

*A:” Function is a module or block of related code, which deals with a particular task. Making a function is a way of isolation one block of code from other independent blocks of code. But before defining a function we can prototype it, simply saying to provide the basic information about a function, which tells the compiler that the function is used correctly or not. It contains the same information as the function header contains. Prototypes help to ensure that the complier can generate correct code for calling one, as well as allowing the complier to catch certain mistakes you might make. It is consider a good practice to use prototype declaration for all functions that you call. To call the function you have you declare:*

*~ the name of the function*

*~ the parameters which enclosed in parentheses*

*~ and return value.”*

**Section 4 - Variables**

**4.1 Data types**

**What are the primitive data types in Objective-C? Why did Apple add a set of classes to handle other data types?**

A: “ *There are 5 primitive data types in Objective-C :*

~1~ int;

*~2~ float;*

*~3~ double;*

*~4~ char;*

*~5~ BOOL.*

*But strings or date are not built- in data type in Objective- C language. That’s why Apple wrote the code to make it easy to work with strings and dates and images and video and menus and hundred of more data types, which are considered as a combination of primitive types.*

**4.2 Working with numbers**

**Make a table of Objective-C primitive data types. Add numeric data types and their**

**properties to this table.**

*A: “*

|  |  |
| --- | --- |
| *int*  *long int(compile in 64 bits)*  *short int* | *4.3 billion combinatins*  *-2147483648 to 2147483647*  *-9223372036854775808 to …*  *-32767 to 32767* |
| *float* | *4 byte* |
| *double* | *8 byte* |
| *char* | *reflect 1 character(1 byte)* |
| *BOOL* | *reflect YES or NO* |

**4.3 Working with characters**

**Add char and BOOL (the character data types) to your table created above.**

**4.4 Variable scope**

**Describe in your own words what the scope of a variable is in Objective-C**

*A: ” Every variable has a scope, a place where its visible and accessible. By default, variables exist inside the statement block they are defined in, but we can manipulate a little with it. We can call a global variable, which is going to have a spectrum for whole program. We have to be careful by not redefine variables and not giving them different values. And have to remember which part of function variable exist in (main, for loop, other ones)*

**4.5 Enumerations**

**What does "enum" allow you to do?**

A*:”It* *Allows you to create a type that will only take certain, predefined values. This is particularly useful, as it allows the programmer to model real-world collections, such as the months of the year, or the players in a game with a specific number of players.*

*An enumerated type is simply a specific list of integers, where each value is given one (or more) names. By default, the first element in a list is set equal to 0, and each subsequent element is assigned a value one greater.*

**4.6 Using typedef**

**When would you define your own data type versus use an enum?**

*A typedef allows the programmer to define one Objective-C type as another. This drastically improves code readability. As well as simplifies a writing of code by redefining variables, which makes easier to create variables of the type instead of using clunky enum one. To define a new typedef:*

1. *Write the statement as if a variable of the original type was being declared*
2. *Where you would put the name of the variable (after the type), place the new type name.*
3. *Prefix everything with typedef*

*This is particularly useful with enumerated types.*

**4.7 Preprocessor directives**

**Describe the three common preprocessor directives, #import, #define, and #if DEBUG. Come up with one example where you would use each.**

*A: “The preprocessor is a program that is invoked by the compiler to process code before compilation. Commands for that program, known as directives, are lines of the source file beginning with the character #, which distinguishes them from lines of source program text. The effect of each preprocessor directive is a change to the text of the source code, and the result is a new source code file, which does not contain the directives. There are few common of them:*

*~ # import <>, which is going to import whatever information that file in braces has and paste in here. Like <foundation/Foundation.h>*

*~ # define, which creates a unchangeable variable with fixed value. And before your code is compiled it replaces with value. Like Pi=3.14*

*~ # if DEBUG, it tests a constant expression or an identifier to determine which tokens the preprocessor should pass on to the compiler and which tokens should be bypassed during preprocessing.*

**4.8 Working with strings**

**Define the same string using both NSString and C-style string syntax. Describe the purpose behind each part of your definition.**

*A:” C-style sting: char message [6]= “Denys”, in this case we have to calculate each character containing in message, even though “Denys” contains 5 of those but one for closing /0.*

*NSString \*message=@”Denys” contains type(NSSring as apart of Foundation framework. That is pre-written code that we are linking to), name (\*(which is a pointer where the address of the variable holds in memory)”message”), and value(@-which is a clue to the compiler that this is a string and”Denys”-which is actual string)*

**Section 5 - Classes**

**5.1 Introduction to object orientation 7m 36s**

**Create an encapsulated (including generalized attributes and behavior) description of a "mobileMakersParticipant" class. Instantiate a single object representing yourself as a member of this class.**

|  |
| --- |
| **mobileMakersParticipant** |
| **name: string**  **sex: stirng**  **age: int**  **previousExperience: BOOL** |
| **getName ()**  **getSex ()**  **getAge()**  **setPeviousExperience ()** |

**5.2 Using objects and pointers 6m 38s**

**What is the pointer's role in instantiating an object from a class? How is a pointer different than a primitive?**

***A****:” Pointer holds an address ,a reference, a pointer that points to a different area of memory where that object itself exist. Objects are more complex that a primitive types just simply because that hold more memory. That’s why we need more flexible way to reserve that memory and pointers would be really helpful in this case by passing the address of the object around and its pointer to that address somewhere in memory. “*

**5.3 Messages and methods 6m 44s**

**What is the main difference between Objective-C's messages and method calls in other languages? How can this difference be seen as an advantage while programming?**

*A:” Method is basically a function that belongs to a class. Its how you describe, what it is that class and objects of the class can do. The main difference of calling a method would be a syntax, instead of dot syntax we use square brackets, and all action is going to happen inside of that brackets. While having method name that takes multiple argument, you can split method name apart into multiple pieces, that will result in different method name, indeed it makes code more readable. “*

**5.4 Using existing classes in the foundation framework 8m 40s**

**What's the difference between a class method and an instance method? EXPLORE: Try typing "NSD…" into your code window. Use the autofill feature and select a single class name that starts with those three letters. Once the name has been auto-completed, use the handy shortcut (Option + click) and investigate the class whose name just got printed to the screen. Examine the task list for this class. Do this a few more times until you're familiar with the process, or until you've exhausted your curiosity, whichever comes last.**

*A:” Instance methods use an instance of a class, whereas a class method can be used with just the class name. Instance methods operate on an object and has access to its instance variables, while a class method operates on a class as a whole and has no access to a particular instance's variables (unless you pass the instance in as a parameter).*

**Section 6 - Memory Management**

**6.1 What's new with memory management? 1m 45s**

**Let it soak in. No questions for this one.**

**6.2 Memory management in Objective-C 6m 58s**

**What is the relationship between a pointer to an object, a block of memory, and the owning and releasing process. Can you come up with an analogy for this relationship?**

*A:” It is a really closed relationship between them, because pointer points to a block of memory where the object is held, it is an address. When we filling that block in memory we are using reference counting, which says to us by “1” meaning object exists and “0” space is available for new one. So each time you create an object by allocating it in memory we are getting responsibilities for that objet, and whenever we done, we must release it. Otherwise we would have a memory leaking”*

**6.3 Object creation 7m 31s**

**What does the new method do when used to create an object instance of a class? Why do we avoid using this method? How long is an object's lifetime?**

*A: “ New method is going to create an instance, new object by using two stages:*

*~ allocation ( will reserve a memory space big enough for our object);*

*~ initialization ( will check what class consists of and initialize variables internally. And then it will return an address in memory of that object that is what the pointer variable will hold).*

*But in Objective- C we will use alloc instead of new.*

*We can not say exactly the lifetime of the object unless we release it or destroy. But sometime its done automatically for reclaiming unused space in memory.*

**6.4 Using autorelease pools 5m 14s**

**How does the autorelease pool work? How and when can you use it deliberately?**

*A:” Autorelease pool- is a list of objects that need to have release called on them at some point later on. This object will last till you call pool drain, which goes to that list and release each object contained there. So pool might only last for milliseconds before it drained. But using autorelease your objects are lasting longer than they need to, but cases when you write a method that need to create and return an object, and releasing it manually will not work, we have to use autorelease. “*

**6.5 Apple autoreleased objects 3m 39s**

**What does NARC stand for? Why is it important to remember this?**

*A:” N- new*

*A- alloc*

*R- retain*

*C- copy;*

*By using narc anywhere in your code means that you own the object and it is your responsibility to release it, otherwise its done automatically by autorelease.*

**6.6 Introduction to Automatic Reference Counting (ARC) 4m 43s**

**What does ARC save us from having to do? How does it keep us from having to make this extra effort?**

*A: ” ARC- automatic reference counting. It is a complier process. Anytime you build the project, the complier is able to scan your code and determine all the possible path through it, figure out where your objects are being used and where not and synthesize the retain and release call at the best point in your program. So I don’t have to worry about Memory Management Code anymore, its simplified for me, even though the complier would give me an error if I use retain and release calls… AWESOME!!!*

**6.7 What ARC manages 2m 42s**

**What are the differences between ARC and garbage collection? What makes these**

**differences advantageous?**

*A: “****Automatic Reference Counting*** *(ARC) is a memory management enhancement where the burden of keeping track of a object’s reference count is lifted from the programmer to the compiler. Rather than having the runtime look for and dispose unused objects in the background, the compiler will inject code into the executable that keeps track of object reference counts and will release objects as necessary, automatically. In essence, if you were to disassemble an executable compiled with ARC, it would look (conceptually) as if the developer spent a lot of time meticulously keeping track of object life cycles when writing the code — except that all that hard work was done by the compiler.*

***Advantages of Automatic Reference Counting***

*~ Real-time, deterministic destruction of objects as they become unused.*

*~ No background processing, and more efficient on lower-power systems, such as mobile devices.*

*The biggest difference is that iOS 5 doesn't have garbage collection.*

**6.8 The rules of ARC 4m 20s**

**Why can you not release or dealloc memory when working with ARC?**

*A:” Because ARC will prohibit me to do so. It will be done automatically. “*

**Section 7 - Custom Classes**

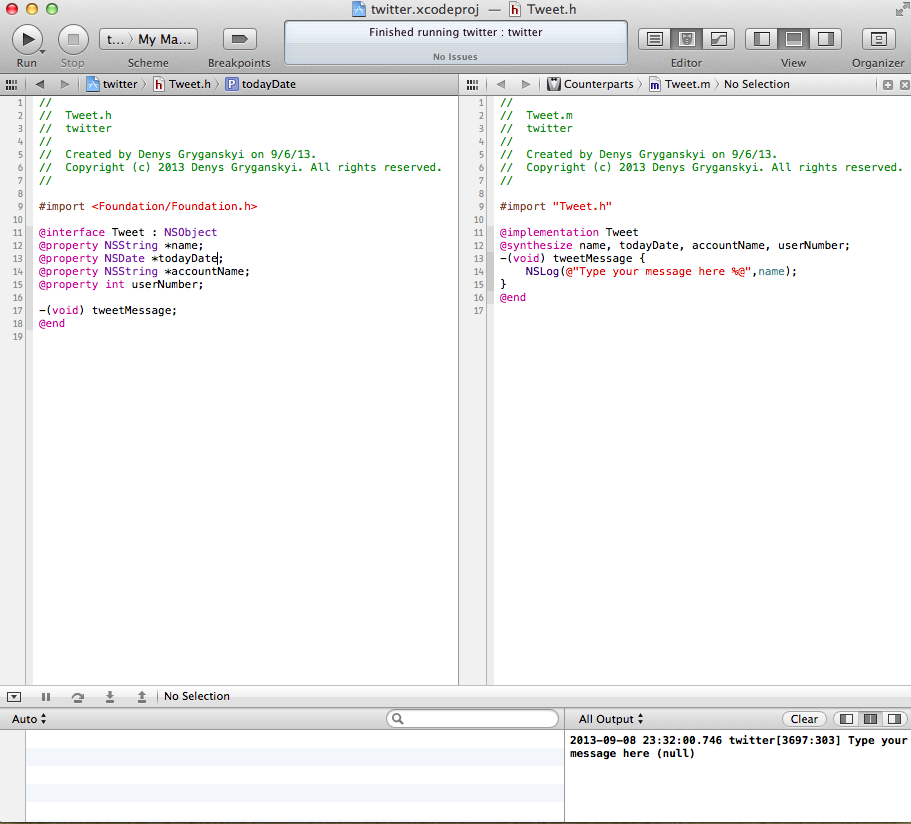
**7.1 Creating your own classes 14m 1s**

**What are the two different sections used to create a class? What do they hold and what files are they placed in? CHALLENGE: Create a Tweet class for a twitter style app.**

*A:” To create a class in Objective- C we use two classes:*

*~ interface, which is the face of the class, which contains of properties which are available, methods it has. But that just a declaration of what we have. Interface just announces those things. And its placed in .h file*

*~ implementation, that’s where the work is done. That’s where we write a code by using interface properties and methods. Its placed in .m file.*

**

**7.2 Defining methods 8m 36s**

**CHALLENGE: Define what should get passed in and what should get returned by each of your methods in your Tweet class above.**

**7.3 Defining properties 7m 21s**

**How did Objective-C programmers handle instance variables before 2012? How are they handled now? What got easier and what got obscured?**

*A:” To define data for your classes, you would define instance variable inside your interface, header file .h file. And those variables would group together inside as a set in curly braces. But the time pass by it wasn’t so convenient, what if I need that instance variables to be used outside of my class, so I would need to create Accessor Methods. That inconvenience was fixed by changing this to @property (this would be stored in interface file) and @synthesize (which generate getter and setter accessor methods in implementation file).”*

**7.4 Defining initializers 12m 30s**

**What are initializers and why do we need to use them? Describe a situations when you can rely on the standard init method and when you have to create your own custom initializer.**

*A:” Initializers exist to initialize new object and make sure it created in valid state. So after an object has been allocated, you send the init message to the new instance so that it can initialize its instance variables to usable values. So alloc creates the space for an object, and init makes the object eady to work. Initializing a variable is only done exactly once when the computer loads your program into memory for execution. That is, all initializations are done before the program starts its execution. Sometimes an object cant be initialized properly without some information from the method that is calling it. You cant do it with init because it has no argument. So I have to create a new initializer- my own custom one. “*

**7.5 Using dealloc 5m 33s**

**Why can we have a dealloc method in a class when using ARC, but we can't call dealloc manually oursevles when using ARC?**

*A:” Dealloc exists for clean up. In case when ARC is turned on we are not allowed to call dealloc method, because this job belongs to ARC. You still can write it, and we need dealloc method in complex applications if our object is holding onto some kind of recourse, but objective- C runtime take care of calling this method for us.”*

**Section 8 - Collections**

**8.1 Working with C-style arrays 7m 12s**

**What are the three constraints when using C-style arrays? Create a C-style array that holds the days of the week.**

*A:” Array is ability to have variable that holds multiple values as the same time. But creating them you have to avoid few constraints:*

*~ no bound checking (reaching memory that we didn’t claim properly);*

*~ fixed size (once created-its fixed);*

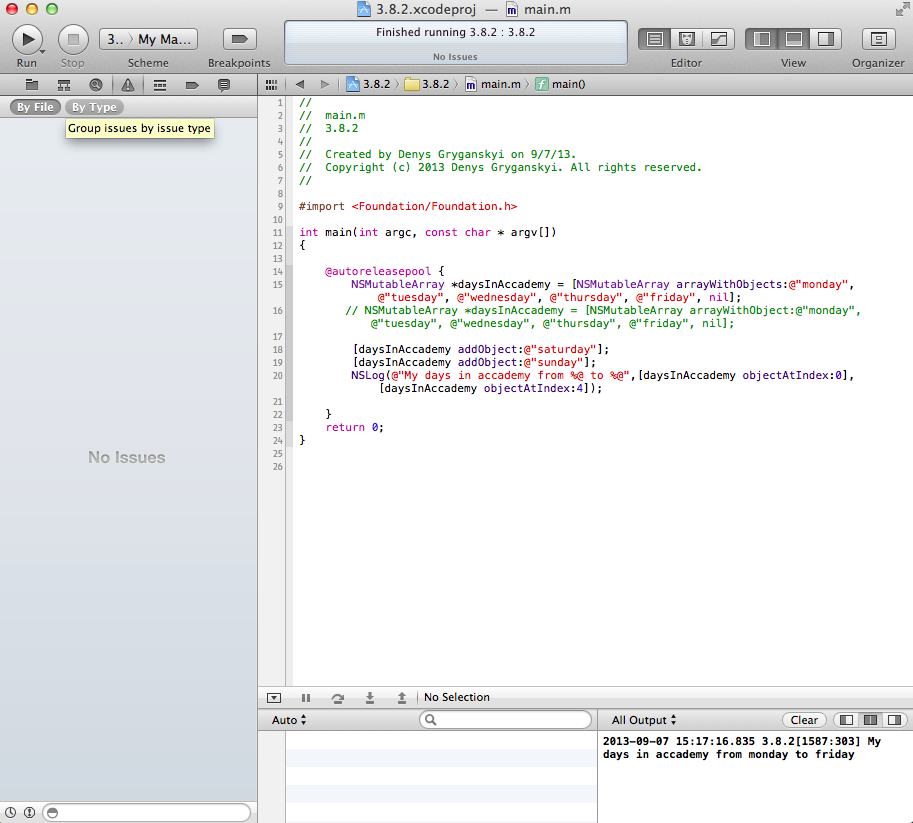
*~ can not mixed types.*

*NSString \*dayOfTheWeek [7]= {@”monday”, @”tuesday”, @”wednesday”, @”thursday”, @”friday”, @”saturday”, @”sunday”};*

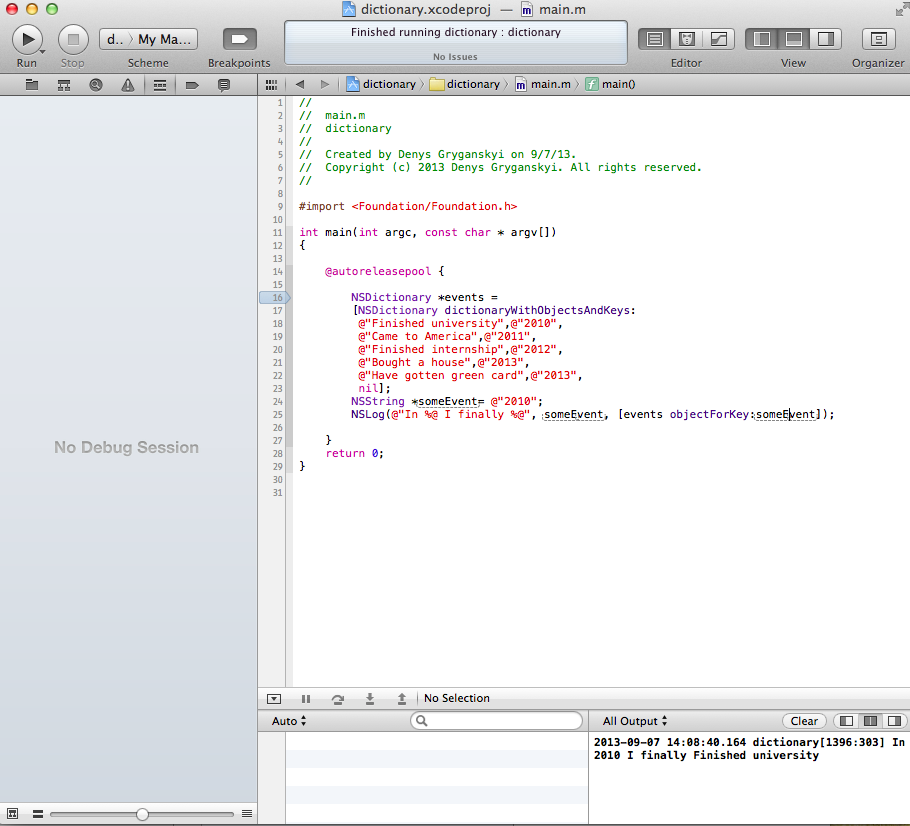
*NSLog (@”Today is %@”, dayOfTheWeek[5]);”*

**8.2 Working with Objective-C array objects 8m 0s**

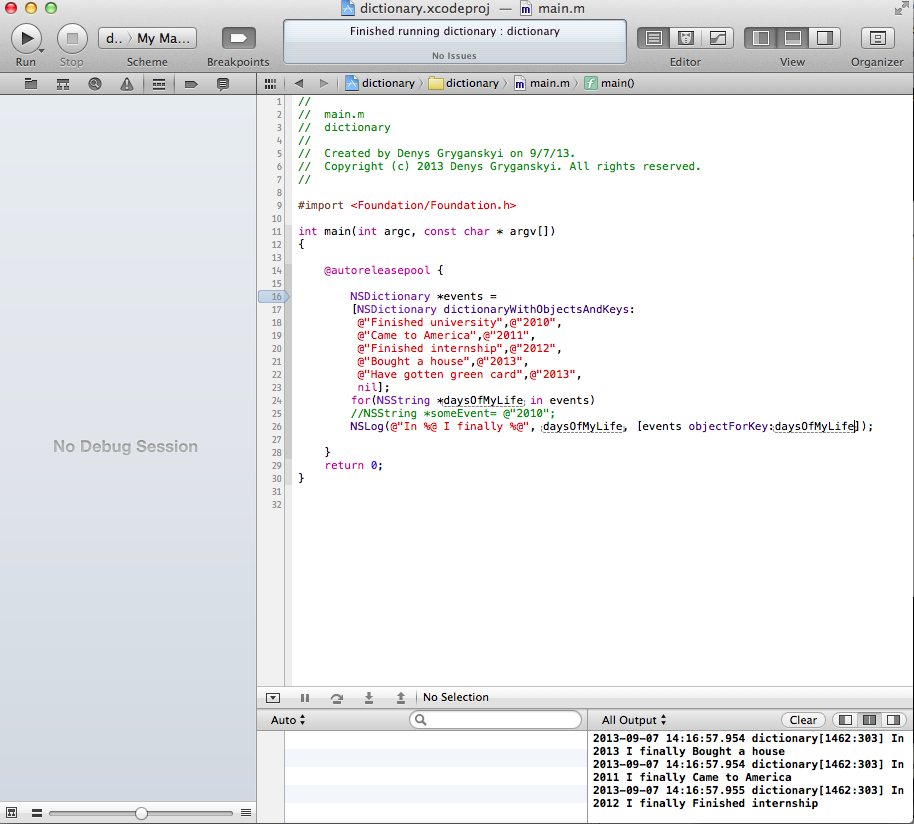
**What is the difference between a mutable and an immutable array? CHALLENGE: Create an immutable array containing the days of the week. Create a mutable array that contains the days of the week that you will be at Mobile Makers. Add the days of the week from the immutable array to the mutable array.**

*A:” NSArray is immutable= unchangeable . once its created you cannot add any object to it, or remove from it. It is fixed size. Unlike Mutable, which is subclass of NSArray, which means it can do all the things parent class can do but has more functionality.*

**8.3 Using dictionaries 5m 55s**

**Create a dictionary that lists five or more events in your life and the accompanying year (or date if you want to get fancy) of the event.**

**8.4 Fast enumeration 3m 27s**

**Use fast enumeration to log the timeline of the life events you described above to theconsole.**

**Section 9 - File Management**

**9.1 Introduction to file management in Objective-C 6m 44s**

**What can you do with files using the methods you are aware of that are available in**

**Objective C's Foundation class.**

*A:” There is a heart of everything called NSFileManager with help if on we can do next:*

*~ save files*

*~ get attributes (NSDictionary will help with this)*

*~ read files*

*~ change name of the file (moveItemAtPath will help with this task)*

*~ copy files ( copyPath)*

*~ read through directories. “*

**9.2 Working with paths and URLs 7m 17s**

**What are the three parts of a URL? What are the advantages to using NSURL?**

*A:” URL consists of*

*~ scheme (http:// or file, or ftp)*

*~ domain (*[*www.denysgryganskyi.com/*](http://www.denysgryganskyi.com/)*)*

*~ path ( /challenge3.docx).*

*NSURL helps t create other objects. And it would be advantageous to use it because:*

*~ faster than using string paths*

*~ better for catching errors*

*~ more powerful as long as it can be used by many classes.*

**9.3 Reading and writing strings 4m 38s**

**What would be a reason you would want to write a string to disk instead of just keeping it memory?**

**!!!! 9.4 Archiving objects 12m 41s**

**Why would you want to archive an object instead of writing the data to disk using the techniques discussed previously?**

*A:” If we have a class with its value and we would like to save it into file system, we would need to write NSString to different attributes of that class which is inconvenient, but instead we can use NSKeyedArchiver, which lets us break apart the different pieces of our object, give them a key, and than store them in separate file system. That what it called encoding.*

*Archives provide a means to convert objects and values into an architecture-independent stream of bytes that preserves the identity of and the relationships between the objects and values.”*

**Section 10 - More Complex Classes**

**10.1 Inheritance and NSObject 8m 13s**

**How can you determine what methods you're inheriting from a super class? How do you overide a method inherited from a super class?**

*A:” Methods which are inherited from superclass marked with [super] like [super init] or [super dealloc] etc.*

*To override a method inherited from a superclass you have to write a method in your new class implementation file with the same signature as a superclass but with no parameter’s changes, return type changes. Because it will be first executed from this file and if not founded, program will go to superclass method.*

**10.2 Extending classes with categories 6m 31s**

**What is the difference between a category and an inheritance? What are the limitations of using a category?**

*A:” Category is not the same as an inheritance, because it doesn’t inherit anything, it simply allows us to add our own methods to an existing class without subclassing it. The creation of category is similar to creation of custom class using interface and implementation files, but the restriction would be that we can’t add variables, only methods!!*

**10.3 Defining protocols**

**How are protocols useful?**

*A:” Protocols are used to declare a list of methods that are used or may be implemented in any other class.*

*Protocols are declared with the @protocol directive. They have no curly brackets with variables since they cannot have any variables associated with them. Protocol method can be marked as required to implement, or as optional to implement. Required methods are marked with the @required keyword, and optional methods are marked with the @optional keyword. If there is no keyword specified, the default is @required. Protocols are useful in many situations, e.g. to declare methods for others to implement, especially when a project is divided among many programmers who are all expected to implement the same group of methods, but may have different implementations. Protocols are also useful in situations where the class of an object isn’t known, or needs to stay hidden.*

**10.4 Dynamic typing**

**What are the advantages and disadvantages to dynamic typing?**

*A:” A variable is dynamically typed when the type of the object it points to is not checked at compile time. Objective-C uses the id data type to represent a variable that is an object without specifying what sort of object it is. This is referred to as dynamic typing. Dynamic typing contrasts with static typing, in which the system explicitly identifies the class to which an object belongs at compile time. Static type checking at compile time may ensure stricter data integrity, but in exchange for that integrity, dynamic typing gives your program much greater flexibility. And through object introspection (for example, asking a dynamically typed, anonymous object what its class is), you can still verify the type of an object at runtime and thus validate its suitability for a particular operation.”*