1. Comments are notes that make a section of code more understandable to a human reader. The Swift compiler ignores them, does not consider them to be commands or instructions. There are two types of comments in Swift (and most programming languages). The first type begins with “/\*” and ends with “\*/”.
2. When the Swift compiler sees “/\*”, it understands this to mean “ignore everything from now until the “\*/” symbols; comments are between these symbols.
3. The second type of comment indicator is “//”. There is no “//” at the end of this type of comment. When Swift compiler sees “//”, it understands that everything from the symbol, until the end of the line it is written on, is a comment.

/\*

// main.swift

// Ch2Demo

//

// Created by JOE FOY (000037286) on 8/23/15.

// Copyright (c) 2015 JOE FOY (000037286). All rights reserved.

// Revisions:

// 0: 8/23/15 Initial release

// 1: 8/23/15 Updated for school year 2016 - 2017

\*/

import Foundation

import Swift

1. Functions (methods in Java, procedures in some other languages) are exactly like the “function notation” you learned about in Algebra; for example f(x) = 2x +5. In Algebra, we would use “f(x)” in some other equation.
2. Functions are a series of commands, that Swift may consider to be another language command; such as “print”. Somewhere else in your code, you may use “strinput()” or “intinput()” or “doubleinput()” exactly as if these are commands built into Swift. Swift sees the following three function descriptions, and will execute them every time it sees “strinput()” etc.
3. A function may require input information, or not. It may provide an output, or not. Example: the command “go to class”. That is enough to tell you, if you are at lunch, to go to your classroom. Another version of this would be “go to L-day 3rd block class”. This function requires an input; 3rd block class. A final example is “go to L-day 3rd block class and open Canvas”. This has an input; where you are going to go, and it produces an output “open Canvas”.
4. Functions that return a value to the code that uses them, have a “return” statement near their ending.

//

// You should copy the following three functions into your

// future console based programs

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Function strinput returns a String which it reads from the Console

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

func strinput() -> String {

let keyboard = NSFileHandle.fileHandleWithStandardInput()

let inputData = keyboard.availableData

let strData = NSString(data: inputData, encoding: NSUTF8StringEncoding)!

return strData.stringByTrimmingCharactersInSet(NSCharacterSet.newlineCharacterSet())

} // end of function strinput

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// Function intinput returns an integer which it reads from the Console

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

func intinput() -> Int32 {

let keyboard = NSFileHandle.fileHandleWithStandardInput()

let inputData = keyboard.availableData

let strData = NSString(data: inputData, encoding: NSUTF8StringEncoding)!

strData.stringByTrimmingCharactersInSet(NSCharacterSet.newlineCharacterSet())

return strData.intValue

} // end of function intinput

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//

// Function doubleinput returns an integer which it reads from the Console

//

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

func doubleinput() -> Double {

let keyboard = NSFileHandle.fileHandleWithStandardInput()

let inputData = keyboard.availableData

let strData = NSString(data: inputData, encoding: NSUTF8StringEncoding)!

strData.stringByTrimmingCharactersInSet(NSCharacterSet.newlineCharacterSet())

1. In programming languages, we need to specify what kind of variable is being used. In math classes, we simply state x = 3, or x = 3.5 or x = 3/5. But computers are all about memory and instructions. Computers represent an integer, a real number, and a very large real number differently. The difference is in how much memory they need to provide to store each.
2. In Swift, this is accomplished by stating “Int” or “Int32” or “Double”, after declaring a variable. In other languages, the data type is specified before the variable name.
3. “var” and “let”. In Swift, “var” describes a variable whose value will change in the program; “let” describes a variable whose value will not change, once it is defined.

var dvalue: Double = 0

dvalue = strData.doubleValue

return dvalue

} // end of function intinput

1. In Swift, functions are declared by first using the reserved word “func”. This is followed by the name of the function, and then a set of parenthesis. If the parenthesis are empty, then the function does not need any information from the program to do its job. If the function needs something from the program to do its job, then that information will be placed inside the parenthesis. Inside the parenthesis will be a variable name that the function is going to use (probably different than the variable the program is using when it calls the function), and the data types.
2. If a function returns a value, then near its end, there will be a “return” statement with the return value. The variable name used for return may be (and probably is) different than the value used in the program.
3. Notice that the block of instructions that are part of the function, are enclosed in curly brackets; “{“ at the beginning, “}” at the end.

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Exploration 11

// Returns angle in radians, not degrees

//

func Pythagorean(a:Double, b:Double) ->(mag:Double, ang:Double)

{

let magnitude = sqrt(a\*a+b\*b)

let angle = atan(b/a)

return(magnitude, angle)

} // end of function Pythagorean

1. To print an output line, use the command “print” followed by a set of parenthesis, with the item inside the parenthesis to be printed. If it is a string of characters (a sentence, phrase), place the exact text inside a set of quotation marks.
2. When using the “print” command; Swift prints the items, and then advances to the next line in the console.
3. If you don’t want the cursor to move to the beginning of the next line, use “print(“….”) with the reserved word “terminator:” and a pair of quotes. This tells Swift to print the word or phrase, but don’t move the cursor to the beginning of the next line.
4. Notice in the following block that there are “var” statements and how functions “strinput()” and “intinput()” are used. Both return a value, which is “caught” by a variable name. Exact syntax is “variable” = “function name”

print("Grade Calculator")

print("Enter Course Name: ", terminator: "")

var course1: String

course1 = strinput()

print("Enter first grade:\t", terminator: "")

var g1: Int32 = intinput()

print("\n Enter second grade:\t", terminator: "")

var g2: Int32 = intinput()

print("\n Enter last grade:\t", terminator: "")

var g3: Int32 = intinput()

print("Course is \(course1)")

1. Computers distinguish between integers and real numbers. A result of this occurs when we use arithmetic on integers that produce a real number in a math class.
2. In programming languages, when we tell the compiler that a variable will be an integer, then it is going to keep that definition.
3. In the following example, if g1 = 90, g2 = 0 and g3 = 92, we would expect the average to be 272/3 = 90.667. But “average” was defined to be an integer, so its value will be integer 90. There are ways to work around this, we will learn them later.
4. Notice how the value of a variable is displayed in Swift. Inside the print parenthesis, and inside the double quotes, we write a backslash, and then the variable name inside parenthesis. This tells Swift that instead of writing the letters “avg” to the screen, write the value of variable “avg” to the screen
5. Other languages do this differently.
6. Also notice the “\t” in the following code. This is known as an “escape sequence”. The backslash tells the compiler to do something different. At the time, it is printing “A”, “v”,.. but then it understands “t” to mean “insert a tab”, it does this, and continues printing.

var avg: Int32

avg = (g1 + g2 + g3)/3;

print(" Average is:\t \(avg)")

1. All programming languages have an “if..” “else..” capability. Inside the parenthesis is an analysis that will be either true or false. If it is true, then the code inside the following curly brackets is completed. If it is false, then code inside the “else” curly brackets will be completed. You do not have to include an “else” statement.

if (g1 > g2)

{print("Grade 1 \(g1) is larger than Grade 2 \(g2)")}

else

{print("Grade 2 \(g2) is larger than Grade 1 \(g1)")}

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("\(course1) \t \(g1) \t \(g2) \t \(g3)")

var side1: Double = 3.0

var side2: Double = 4.0

var hypotenuse: Double

var angle: Double

25. Notice how function Pythagorean is used; two variables are passed to it, the returned value is assigned to “vector”, which has two sub-values; magnitude and angle.

let vector = Pythagorean(side1, b:side2)

print("Hypotenuse is \(vector.mag), Angle is \(vector.ang)")