Swiftian

LEARN TO CODE

SeesawApps.com

CHAPTER 1

First Steps

A Good First Program

The first program you write and run is the “Hello World” program. All it does is just print “Hello World” when you run it.

```
print(“Hello World”)
```

Notice that the print command shows what to print on the screen, not on the printer!

How to Fix Errors

What if we misspell print as Print? Note the capitalization.

```
Print("Hello World")
```

In this case, a syntax error occurs. To fix this, just replace the first character with lowercase p.

```
print("Hello World")
```

SECTION 3

Printing Multiple Arguments

If you want to print multiple values, just pass them inside the parentheses separated by commas.

```
print("Hi", "there!")
```

```
Hi there!
```

SECTION 4

User Input

Let's take input from the user and then print some results back.

The input command prints a string and waits for input from the user.

```
input("What is your name?")

handleInput {
    print("Hi!", userInput)
}
```

The handleInput block is called when you enter text. Your text is passed in as user input. So the program will say hi to you.

```
What is your name?
Sam
Hi! Sam
```

Code Blocks

The curly braces { } represents a code block - often for conditionals, function definitions, and control flow structures.

```
for i in 0..<5 {
    print(i)
}
```

We'll see more about this in later. For now, just notice the print command is inside the block. The block enumerates the value of i from 0 to 4.

```
0
1
2
3
4
```

CHAPTER 2

Basics

Comments

Comments are ignored when the program runs. They are added with the purpose of making the code easier for humans to understand. Comments are any text to the right of the //.

```
print("Hello World") // My First Program !!!
```

Notice that "My First Program !!!" is not printed.

```
Hello World
```

Multiline Comments

Multiline comments start with /* and end with */.

Everything in between them is ignored. It's for your eyes only.

```
/* This is a multiline comment.
Second line,
Third line,
...
*/

print("Hello World")
```

```
Hello World
```

Variables dsfasdffdsa

Using variables you can store information. The var is the keyword that indicates a new variable declaration.

```
var num = 0
num = 3
print(num)
```

For example, this is how we create a variable called num, which contains an integer value of zero.

Once you declared a variable, you can assign a new value to the variable.

```
3
```

Practice

If you try to use a variable which hasn't been declared yet, a syntax error occurs.

```
print(sum)
```

To resolve this error, declare the variable first.

```
var sum = 2
print(sum)
```

Assignment

An assignment is to set a value to a variable name. The single equal sign is the assignment operator. Here are a few examples:

```
var total = 2
total = total + 1
print(total)
```

```
3
```

In the second line ‘total = total + 1’ might look a bit strange if we were to interpret ‘=’ as a mathematical equals sign - this statement is assigning a new value to the variable. The new value is equal to the old value plus one.

More Practice

Why do I get an error?

```
var a = 3
4 = a
print(a)
```

The left side of the assignment must be a variable name. Remember, you need to assign a value to a variable not vice versa.

CHAPTER 3

Data Types

Integers and Floats

Integers and floats are two different kinds of data. An integer is a number without a decimal point. A float is a number that has a decimal place.

```
var i = 2
var f = 3.5
```

Strings

A string is a sequence of characters. You can specify strings using double quotes such as “This”. You can concatenate strings with plus operator.

```
var s = “Hi!” + “Tom”
print(s)
```

```
Hi! Tom
```

Practice

Spot the difference between strings and numbers.

```
var s = "2" + "3"
print(s)
var i =  2 + 3
print(i)
```

```
23
5
```

CHAPTER 4

Operators

Arithmetic Operators

Addition	Subtraction	Multiplication	Division	Modulus
+	-	*	/	%

The modulus operator % finds the remainder after division.

```
print(2 * 3)
print(6 / 2)
print(10 % 3)
```

```
6
3
1
```

Comparison Operators

Equal-to operator:

In programming == sign or double equal sign means we are comparing right side with left side. And this comparison returns a boolean data type that has two possible values: it is either true, or false.

```
var a = 3
var b = 2
print(a==b)
```

```
false
```

Not-equal-to operator:

Denoted by !=, this does the exact opposite of the equal-to operator.

More Comparison Operators

These operators compare the values on either sides and decide the relation among them.

Operator	Description
<	Returns whether x is less than y. 5 < 3 gives false and 3 < 5 gives true.
>	Returns whether x is greater than y. 5 > 3 returns true.
<=	Returns whether x is less than or equal to y. 3 <= 6 returns true.
>=	Returns whether x is greater than or equal to y. 4 >= 3 returns true.

```
var a = 3
var b = 2
print(a < b)
```

```
false
```

Logical Operators

There are following logical operators:

Operator	Description
&&	Logical AND If both sides are true then true
||	Logical OR If any of the two sides are true then true
!	Logical NOT User to reverse the state

```
var a = (3 > 2)
var b = (3 == 2)
print(a || b)
```

```
true
```

CHAPTER 5

The if Statement

if

Run this example first. If your name is Jay, your name will be welcome.

```
input("What's your name?")

handleInput {
    if userInput == "Jay" {
        print("Jay, ")
    }
    print("Welcome!")
}
```

```
What's your name?
Jay
Jay, Welcome!
```

If the userInput is equal to "Jay", the if block is executed.

SECTION 2

if...else

Let's make a guess game. In this program we take guesses from the user and check if it is the number that we have. We set the variable number to 23. Then, we take the user's guess using the input command.

```
var secret = 23
input("Guess the number!")

handleInput {
    var guess = Int(userInput)
    if guess == secret {
        print("Congrats!")
    } else {
        input("No!")
    }
}
```

```
Guess the number!
7
No!
23
Congrats!
```

Random Number

The rand function generates a random number from 0 to the given number. For example, rand(100) returns a value between 0 and 100 (starting zero, and up to, but not including 100).

```
var secret = rand(100) + 1
input("Guess the number (1~100)!")

handleInput {
    var guess = Int(userInput)
    if guess == secret {
        print("Congrats, you guessed it")
    } else if guess < secret {
        input("No, it's higher than that.")
    } else {
        input("No, it's lower than that.")
    }
}
```

```
Guess the number (1~100)!
```

CHAPTER 6

The for Loop

Iteration

The for loop is to specify iteration, which allows code to be executed repeatedly. Take a look at the example code. The i is a variable name you define. Inside the block you can read the variable.

```
for i in 0..<5 {
    print(i)
}
```

```
0
1
2
3
4
```

And 0..<5 means that the value of i will iterate from 0 to 5 but not including 5.

Range Operators

Notice the difference between the two range operators.

0..<5 means 0 1 2 3 4.

0…5 means 0 1 2 3 4 5.

Here's the code example:

```
for i in 1...5 {
    print(i)
}
```

```
1
2
3
4
5
```

Iteration over an Array

0..<5 is equivalent to an array [0, 1, 2, 3, 4]. Arrays will be explained right after.

```
var numbers = [0, 1, 2, 3, 4]
for number in numbers {
    print(number)
}
```

```
0
1
2
3
4
```

CHAPTER 7

Arrays

What is an Array?

The array of items is enclosed in square brackets.

If you declare a variable abc which is an array:

```
var names = [“a”, “b”, “c”]

names.append(“d”)
print(“Names are”, arr)

names.removeLast
print(“Now names are”, arr)
```

```
Names are [“a”, “b”, “c”, “d”]
Now names are [“a”, “b”, “c”]
```

Array Index

You can read items from an array by using the item index. You can access any array element this way.

```
var names = [“a”, “b”, “c”]
var x = names[0]
var y = names[1]
var z = names[2]
print(x,y,z)
```

```
a b c
```

To get the number of elements in an array, use “count” function on the array.

```
print(names.count)
```

```
3
```

Iteration with Index

With the number of elements in an array, we can use the for loop to iterate through the items of the array.

```
var names = [“a”, “b”, “c”]
var num = names.count
for i in 0..<num {
    print(names[i])
}
```

```
a
b
c
```

Iteration without Index

We can also use the standard approach to iterate through the items of the array.

```
var names = [“a”, “b”, “c”]

for name in names {
    print(name)
}
```

```
a
b
c
```

CHAPTER 8

Functions

sayHello

You can reuse code by defining functions. Functions are defined using the func keyword. An example will show that this is actually simple:

```
func sayHello {
    print("Hello, nice to meet you! I'm so pleased!")
}

sayHello
sayHello
sayHello
```

```
Hello, nice to meet you! I'm so pleased!
Hello, nice to meet you! I'm so pleased!
Hello, nice to meet you! I'm so pleased!
```

We define a function sayHello and call the function three times which means we do not have to write the same code again.

Function Parameters

While sayHello in the previous example takes no parameters, functions can take parameters which are values you supply to the function. For example, if you write printMax function you need two parameters a and b from which you will choose the max. Let's see the example:

```
func printMax(a, b) {
    if a > b {
        print(a)
    } else {
        print(b)
    }
}

printMax(23, 78)
```

```
78
```

Variable Scope

If you declare a variable inside a function, outside the function you cannot see the variable.

Say if you declared a variable x inside a function, outside the function you cannot access the variable x.

```
func abc {
    var x = “San Francisco 49ers”
}
print(x)
abc
```

To fix the error, move the print(x) statement inside the func block.

```
func abc {
    var x = “San Francisco 49ers”
    print(x)
}

abc
```

```
San Francisco 49ers
```

The return Statement

The return statement is used to return from a function. We can optionally return a value from the function as well. The inchToMeter function converts inches to meters. In this case

```
func inchToMeter(inch) {
    var meter = inch * 0.0254
    return meter
}

var what = inchToMeter(100)
print("100 inches equal to", what, "meters")
```

```
100 inches equal to 2.54 meters
```

inches supplied to the function and then it calculates meters.

The import Statement

If you want to reuse a number of functions in other files, import that file.

```
import “8.4 The return Statement”
print(“100 inches equal to”, what, “meters”)
```

```
100 inches equal to 2.54 meters
```

CHAPTER 9

Drawing Class

The Drawing Class

Generally, you create an instance of a class to use it.

Type Drawing to create an instance of this class.

After pressing the run button, you can draw lines that follow your touch.

```
Drawing
```

How to Change Colors

A Drawing class has its own properties such as lineColor and lineWidth.

Let's create a variable of the Drawing class. You can change the line color and the line width by specifying values to the properties which follow by a dot notation.

```
var d = Drawing
d.lineWidth = 10
d.lineColor = [255,0,0,1] // red
```

Color is an array of four numbers, Red, Green, Blue, and Opacity.

Red, Green, and Blue are colors in the range of 0 to 255.

Opacity is a number between 0 and 1.

Combine red, green, and blue to make any color. For instance, yellow is a mix of red and green.

[255,255,0,1] // yellow

SECTION 3

Polygons

To draw a polygon on the screen:

Specify a start point, add as many lines as you need, and close the path.

```
var d = Drawing
d.start(0,0)
d.addLine(width, height)
d.addLine(300,200)
d.close
```

The coordinates of a point are given inside the parentheses. They are a pair of numbers that define the location on (x, y) plane. Its origin or (0, 0) is at the upper left of the screen and (width, height) is the bottom right.

CHAPTER 10

Image Class

Loading Images

There are pre-saved images such as “optimus”. You can load the image by creating an Image instance.

```
Image(“optimus”)
```

Changing Sizes

You can apply a mode to your images. The “center” mode centers the image on the screen. The “fit” will enlarge or shrink the image according to your screen’s width to get a proper fit. Choosing the “fill” mode will stretch the image and fit it on the screen but may distort it.

How to Move Images

If you want to move your image into 200, 300, set the center to 200, 300.

```
card.center = [200, 300]
```

or use moveTo method.

```
card.moveTo(200, 300)
```

The moveTo method can have a third argument that is the animation duration in seconds.

```
card.moveTo(200, 300, 0.5)
```

To enable the image to recognize a tap gesture, use addTap method. When the image is tapped, handleTap block is called. The self keyword is the image itself tapped. Let's create flying card effects when the card is tapped.

```
card.addTap

hadleTap {
    self.moveTo(100, -200, 0.50
}
```

Images from Camera Roll

To bring an image from the camera roll, use pick method. The brought image will be saved as the filename for later use.

```
Image.pick("filename")
```

SECTION 5

Saved Images

To see the saved images, print saved images. Saved images can be reached by Image.files.

```
print(Image.files)
```

If you want to delete an image, use Image.delete(“filename”).

```
Image.delete(“filename”)
```

CHAPTER 11

Label Class

What's a Label

When you want to position text on the screen for example to display game scores, use a Label class. It draws text as an image in a rectangle are you specify on the screen. For the rectangle area, the origin is in the upper-left corner and the rectangle extends towards the lower-right corner. Create a label with the origin coordinates, width and height.

```
Label(x, y, w, h)
```

Label Properties

To specify the text and color of the label, set them as what you want. For example, when you create a label, define a variable to get the reference of it.

```
var lbl = Label(x, y, w, h)
```

Now, set the text of the label to “Hello Label” and set the color to red.

```
lbl.text = “Hello Label”
lbl.color = [255,0,0,1]
```

Tap Gestures

The addTap method enables the label to recognize the tap gesture. When you tap the label, the handleTap block is called. Let's change its text when it is tapped.

```
var lbl = Label(50,100,80,40)
lbl.text = "Tap me!"
lbl.addTap

handleTap {
    self.text = "tapped"
}
```

CHAPTER 12

Map Class

How to Display a Map

Just type the Map keyword. Run this extra simple command and you'll have a fully functional map showing your current location.

```
Map
```

Eiffel Tower

You can center the map where to display, giving a latitude and longitude. The loc property is the center point.

```
var m = Map
m.loc = [48.858093, 2.294694]
```

The radius property determines the size of region to display in meters. The default map type is a standard type (“std”). Use the type property to change map type to satellite (“sat”) or flyover (“fly”).

```
m.type = “sat”
```

Locations

Find your current GPS location using the loc property.

```
var m = Map
print(m.loc)
```

Yet you cannot see the text. Because when you declare the map view it is on top of your default view. The map view is blocking the text, so the text cannot be seen. In this case you have to change the order of views. Send the map view below the default view.

```
m.sendToBack
```

The opposite method is bringToFront.

CHAPTER 13

Speech Kit

Text to Speech

Have you ever thought about converting text to speech in your own program? The Speech kit does the magic. Just add the speak method. It converts the text into natural voice. You can change the voice to almost any language and local using the setVoice method. For example, If you want to speak British English, enter “en-GB” as an argument in the setVoice method.

```
var str = “The water literally washes the anxiety away.”
Speech.setVoice(“en-GB”)
Speech.speak(str)
```

SECTION 2

Speech Recognition

Now, it's time to implement speech recognition and test how British your accent is. The recognize method does the magic. First, set the voice setting to British English. Second, add speech recognition. Finally, add the handleInput block. Whenever you say something, the transcription is passed in as userInput. There you go! Just clear the screen and print it.

```
Speech.setVoice("en-GB")
Speech.recognize
handleInput {
    clear
    print(userInput)
}
```