Swift

- language used to write applications in iOS, macOS, other Apple devices

- Apple, 2014, open source

- we will focus on Swift language, not app development

Variable

- variable must be declared using var

- type must be specified or inferred

```
- var x: Int              //type specified
  x = 5

- var x: Int = 5          //type specified

- var x = 5               //type inferred

- var x                   //error, type not specified
                          //or inferred
- var x: Int
  print(x)                //error, cannot use variable
                          //without value

- var x: Int              //error, type mismatches are
  x = 3.2                 //not allowed
```

Type

- Int - 35, -650

- Float, Double - 3.7, 2.63

- Bool - true, false

- Character - "p", "5"

- String - "tree", "box"

Constant

- let is used to define a constant

- type is specified or inferred

- value can be assigned only once

- let x: Int = 5 //type specified, value assigned

- let x: Int //type specified, value assigned
  x = 5

- let x = 5 //type inferred, value assigned

- let x = 5 //error, value assigned twice
  x = 3

Identifier

- Identifiers use letters, digits, underscore

- cannot begin in digit

- case sensitive

- sum, number2, max_value, AverageValue

Semicolon

- Semicolons are optional at the end of statements

- Required when there are multiple statements in one line

```
- print("hello");                    //optional

  print("hello"); print("world")   //required
```

Comment

```
//single line comment

/* multiple line
   comment
*/
```

Print

```
- print("hello world")              hello world

- print("hello")                    hello
  print("world")                    world

- var k = 3
  print("value is", k)              value is 3

- var k = 3
  print("value", k, "square", k*k) value 3 square 9

- print("hello", terminator:"")     hello world
  print("world")

- automatically adds blank and newline, unless
  specified otherwise
```

Arithmetic

```
- operators + - * / %

- 2*(3+2) is 10, 15/2 is 7, 15/2.0 is 7.5
```

- no ++, -- operators

- += -= *= /= are available

Boolean

- var x: Bool
  x = true

- var x = false
  print(x)

Relational operations

- > < >= <= == !=

- can be used with numbers or strings

- 5 > 3 true
  "apple" < "box" true
  4 == 5 false

Logical operations

- && || !

- 5 > 3 && 5 < 2 false
  "apple" > "box" || 2 < 5 true
  !(5 > 3 && 5 < 2) true

Switch/case

- similar to Java

- break is not needed, only the first matching case is executed

If else

```
- if (a > b)
  {
     . . .
  }
  else
  {
     . . .
  }
```

```
if (a >= 9)
{
   . . .
}
else if (b == "tree")
{
   . . .
}
else
{
   . . .
}
```

- { } required, ( ) optional

While loop

```
- var k = 1
  while (k <= 10)
  {
      print(k)
      k = k + 1
  }
```

- { } required, ( ) optional

For loop

```
- for k in 1...5      //goes from 1 to 5
  {
     print(k)
  }

- var max = 10
  for k in 1...max    //goes from 1 to max
  {
     print(k)
  }
```

- for k in [2, 9, 5]
  {
     print(k)
  }

- for k in "hello"
  {
     print(k)
  }

- { } required
  for loop variable is not declared, type inferred
  for loop variable cannot be modified

Repeat-while, break, continue

- similar to Java

Function

```
- func f(x: Int, y: Int) -> Int
  {
     var temp: Int
     temp = x + y;
     return temp
  }

  var answer = f(x: 2, y: 3)
```

- parameter name must be specified in function call

- formal and actual parameter names must match

Function

```
- func f()
  {
      print("hello")
  }
```

```
f()
```

-
```
func f(n: Int) -> (square: Int, cube: Int)
{
    var a = n*n
    var b = n*n*n
    return (a, b)
}

var x = f(n: 5)
print(x.square)
print(x.cube)
```

Function

- local variables have local scope

- parameters are passed by value, local copies made

- parameters can be accessed but not modified

- return names cannot be accessed or modified

Function

- function may have local and external parameters, function uses local name, caller uses external name

```
func f(number n: Int)
{
    print(n*n)
}

f(number: 2)
```

- underscore external parameter, call is made without parameter name

```
func f(_ n: Int)
{
    print(n*n)
}

f(3)
```

- inout parameter affects calling variable

```
func f(n: inout Int)
{
    n = 3
}

var value = 2
f(n: &value)
value is 3
```

Optional

- optional type declaration allows a variable to have a value or nil

- `var x: Int?` //x may have a value or nil

-
```
var x: Int?
if (x == nil)
{
   print("no value")  //prints no value
}
else
{
   print(x!)
}
```

-
```
var x: Int?
x = 5
```