Swift Fundamentals I
Data Types / Control Flow

CS112 Unit 2

Max Luttrell, Fall 2016

identifiers

- **identifiers** are case-sensitive names for variables, constants, functions, classes etc.

- we cannot use reserved words (e.g. we can't call something **let** because that is a reserved word in Swift)

- make it **descriptive**! e.g. **numStudents,** not **x**

identifiers example

```
let apples = 3
let oranges = 5
let appleSummary = "I have \(apples) apples."
let fruitSummary = "I have \(apples + oranges) pieces of fruit."
```

```
print(appleSummary)
print(fruitSummary)
```

Sample Debug Output

```
I have 3 apples.
I have 8 pieces of fruit.
```

data types

- Swift provides several data types, depending on the data being stored, including:
  - integer - whole numbers `(Int)`
  - floating point - decimal numbers `(Float, Double)`
  - single character `(Character)`
  - string `(String)`
  - boolean - true or false `(Bool)`

data types

- Swift does not require us to provide the data type in order to create a variable or constant, it will determine the type for us if we provide an initial value:

```
var apples = 3
```

- however, we can specify the type if we want:

```
var oranges: Int = 5


var bananas: Int

bananas = 8
```

floating point data types

- A **Float** can be used for real numbers (e.g. dollar amounts, a ratio, etc.).
  - `var taxRate: Float = 0.08`
- A **Double** can be used for real numbers when you need more precision.
  - `var pi: Double = 3.14159265358979`

character and string

- A **Character** can be used for a single character
  - `var answer: Character = "n"`
- A **String** can be used for a string of characters
  - `var like: String = "I like CS112?"`

operators

operator	meaning	comments
+	addition	
–	subtraction	
*	multiplication	
/	division	watch out for integer division -- truncates!
%	modulo	remainder from integer division
=	assignment	stores right side to a variable on left side

math example

```
let numStudents = 30
let numMacs = 20

print("We have \(numStudents/numMacs) students per mac")
```

Sample Debug Output

```
We have 1 students per mac
```

type cast

- type cast: for this line, convert a variable to a different type
- note: variable remains its original type

```
let numStudents = 30
let numMacs = 20

print("We have \(Float(numStudents)/Float(numMacs)) students per mac")
```

Sample Debug Output

We have 1.5 students per mac

modulo

- the modulo operator (%) gives us the remainder after integer division

```
let largeNumber = 12345
let lastTwoDigits = largeNumber % 100
print("The last 2 digits are: \(lastTwoDigits)”)
```

Sample Debug Output

```
The last 2 digits are: 45
```

Exercise 2A

- You're planning a wedding, and you need to transport all your wedding guests from the ceremony to the reception.

- I have begun a Swift playground for you by creating two constants below. Create a new playground, and copy this into the beginning of your own playground in Xcode

- Add some Swift code to print out the number of buses needed to the debug area, and also the number of extra people you could carry with those buses

- You should be able to change numGuests to a different value, and your playground should compute the correct number of buses and extra capacity

```
import UIKit

let numGuests = 73
let busCapacity = 40
```

control flow

- **control flow** allows us to change the behavior of our program
  - **decision**: execute some code only if a condition is true
  - **loop**: execute some code multiple times

relational operators and expressions

- Assume x is 5, y is 8, and z is 5. Are the following true or false?
  - x < y
  - x > y
  - x <= y
  - x >= y
  - x != y
  - x == z
  - x < 7
  - x < (y - 4)

if

```
var score = 2

if (score == 0) {
  print("We had a shutout!")
}

if (score > 0) {
  print("We had a score!")
}

// what is wrong with below?
if (score = 0) {
  print("We had a shutout!")
}
```

Sample Debug Output

We had a score!

if-else

```
if (score == 0) {
    print("We had a shutout today!")
} else {
    print("No shutout today")
    print("The score was: \(score)")
}
```

else if

```
var age = 15

if(age < 13) {
  print("child");
} else if (age < 18) {
  print("teen");
} else {
  print("adult");
}
```

else if

```
var age = 15
var workingAge = false

if(age < 13) {
    print("child")
    workingAge = false
} else if (age < 18) {
    print("teen")
    workingAge = false
} else if (age < 65) {
    print("adult")
    workingAge = true
} else {
    print("retired")
    workingAge = false
}

if (workingAge) {
    print("Want a job?")
}
```

Exercise 2B

- I have begun a Swift playground for you by creating two constants below. jillAge is Jill's age, johnAge is John's age. Copy it into the your own playground in Xcode

- Add some Swift code to print out the ages, who is older, Jill or John, or if they are the same age, all to the debug area

- You should be able to change johnAge and jillAge to different values, and your playground should print the correct result

```
import UIKit

let jillAge = 73
let johnAge = 40
```