2. Conditionals

Introduction

Sometimes you want to run some code only **if** some conditions are met. For example:

```
var numberOfOranges = 1
var numberOfApples = 5

if numberOfApples > numberOfOranges {
    print("You have more apples than oranges!")
}
```

You can compare numbers using these operators:

- `<` Less than
- `<=` Less than or equal
- `>` Greater than
- `>=` Greater than or equal
- `==` Equal
- `!=` Not equal

```
1 != 2 // true
1 == 2 // false
1 < 2 // true
1 > 2 // false
1 <= 2 // true
3 >= 3 // true
```

Anatomy of an if statement

An `if` statement has the following form:

```
if CONDITION {
    STATEMENT
    STATEMENT
    ...
    STATEMENT
}
```

The statements between curly braces (`{}`) will only be executed if the given condition is `true` ! The statements that follow after the curly brace ends will be executed independently of the condition.

An if statement can also have an `else` branch:

```
if CONDITION {
    STATEMENT
    STATEMENT
    ...
    STATEMENT
} else {
    STATEMENT
    STATEMENT
    ...
    STATEMENT
}
```

The statements in the `else` branch i.e. between `else {` and `}` will only be executed if the condition is false.

Consider the following code as an example:

```
var money = 20 // you have 20$
var burgerPrice = 10 // you ate a good burger

// if you have enough money pay for the burger
if money >=  burgerPrice {
    print("pay burger")
    money -= burgerPrice
} else {
// otherwise you will need to go wash dishes to pay for your meal
// hopefully this will not be the case
    print("wash dishes")
}

// if you have some money left order desert
if money > 0 {
    print("order desert")
}
```

Nesting conditions

If statements can be nested inside other if statements.

```
if CONDITION {
    STATEMENT

    if CONDITION2 {
        STATEMENT
        STATEMENT
        ...
        STATEMENT
    }

    STATEMENT
```

```
    ...
    STATEMENT
}
```

For example lets say we have two variables `age` and `money`. We'll write some code to determine if you can buy a car that costs 20000. For this you'll need at least 20000 money and at least an age of 18:

```
var age = 23
var money = 25000

if age >= 18 {
    if money >= 20000 {
        print("Getting a new car, baby!")
    } else {
        print("Sorry, you don't have enough money.")
    }
} else {
    print("Sorry, you're not old enough.")
}
```

Multiple conditions

Mutiple conditions can be chained together using the `&&` (AND) operator and the `||` (OR) operator

The `&&` (AND) operator is used to check if two conditions are simultaneously true. For example consider we have the age of a person stored in a variable `age` and want to determine if the person is a teenager (age is between 13 and 19). We have to check that the age is greater than or equal to 13 **AND** less than or equal to 19. This is accomplished by the code below:

```
var age = 18
if age >= 13 && age <= 19 {
    print("Teenager")
}
```

This is equivalent to the following code:

```
var age = 18
if age >= 13 {
    if age <= 19 {
        print("Teenager")
    }
}
```

The `||` (OR) operator is used to check that at least one of two conditions is true.

Consider again that we have the age of a person stored in a variable `age`. We want to print a warning if

the age is less than or equal to 0 **OR** the age is greater than or equal to 100. This is accomplished by the code below:

```
var age = 123
if age <= 0 || age >= 100 {
    print("Warning age is probably incorrect!")
}
```

Note: The **OR** in programming is not equivalent to the or in everyday language. If someone asks you if you want beef or chicken that means that you can have only one of two. In programming an or statement is also `true` when both conditions are `true` at the same time. For example:

```
var numberOfSisters = 1
var numberOfBrothers = 2

if numberOfSisters > 0 || numberOfBrothers > 0 {
    print("Has siblings")
}
```

To get a better understanding of how **AND**(`&&`) and **OR**(`||`) behave have a look at the truth tables below:

```
// AND
true && true // true
true && false // false
false && true // false
false && false // false

// OR
true || true // true
true || false // true
false || true // true
false || false // false
```

Negating a condition

You can negate a condition using the `!` operator. A negated condition has oposite value to the original condition. i.e. if the initial condition was `true` than it's negation is `false`. If the initial condition is `false` than it's negation is `true`.

For example if we wanted to check if an age is **NOT** the age of a teenager we could use the following code

```
var age = 18
if !(age >= 13 && age <= 19) {
    print("Not a teenager!")
}
```

Note:

```
if condition {
    // DO SOMETHING WHEN CONDITION IS TRUE
} else {
    // DO SOMETHING WHEN CONDITION IS FALSE
}
```

is equivalent of :

```
if !condition {
    // DO SOMETHING WHEN CONDITION IS FALSE
} else {
    // DO SOMETHING WHEN CONDITION IS TRUE
}
```

Note: If you have an if statement with an else branch than it's not recommended to negate the condition.

The below table shows the values of negating some conditions:

```
!true // false
!false // true
!(true && true) // false
!(true || false) // false
!(false || false) // true
```