Swift Fundamentals II
Loops / Arrays

CS112 Unit 3
Max Luttrell, Fall 2016

while loop

- like an **`if`** statement, but **repeats** executing statement(s) as long as expression is true

```
while expression {
  statement
}
```

```
while expression {
  statement
  statement
  ...
}
```

while loop example

- Does the following print anything out?

```
let burritoCost = 6.95
var myMoney = 10.00

while myMoney < burritoCost {
    print("Darn, not enough money for a burrito today")
}
```

while loop example

- Does the following print anything out?

```
let burritoCost = 6.95
var myMoney = 5.00

while myMoney < burritoCost {
    print("Darn, not enough money for a burrito today")
}
```

while loop example

- Does this loop execute? What does it print out?

```
let burritoCost = 6.95
let salary = 1.00
var myMoney = 5.00

while myMoney < burritoCost {
    print("Darn, not enough money for a burrito today")
    myMoney += salary
}

print("yum!")
```

Sample Debug Output

Darn, not enough money for a burrito today
Darn, not enough money for a burrito today
yum!

another while loop example

```
var balance = 10000.00
var yearsPassed = 0
var yearsToInvest = 7
let interestRate = 0.05

while yearsPassed < yearsToInvest {
    balance += balance * interestRate
    yearsPassed+=1
    print("After \(yearsPassed) years: $\(balance)")
}
```

Sample Debug Output

```
After 1 years: $10500.0
After 2 years: $11025.0
After 3 years: $11576.25
After 4 years: $12155.0625
After 5 years: $12762.815625
After 6 years: $13400.95640625
After 7 years: $14071.0042265625
```

nicer formatting for our money

```
var balance = 10000.00
var yearsPassed = 0
var yearsToInvest = 7
let interestRate = 0.05

while yearsPassed < yearsToInvest {
    balance += balance * interestRate
    yearsPassed+=1
    print("After \(yearsPassed) years: $\(String(format:"%.2f",balance))")
}
```

Sample Debug Output

```
After 1 years: $10500.00
After 2 years: $11025.00
After 3 years: $11576.25
After 4 years: $12155.06
After 5 years: $12762.82
After 6 years: $13400.96
After 7 years: $14071.00
```

Exercise 3A

- in a new playground, create a constant **numHellos** and give it the value 5

- now, write a while loop that outputs "hello" to the user as many times as specified in the constant **numHellos**

- once you have that working, create a second while loop which alternates between capital and lower-case hellos, such as the below:

```
hello
HELLO
hello
HELLO
hello
```

for-in loop

- similar to a while loop, but iterates over a sequence, for example a range of numbers

```
for index in 1...5 {
    print("hello")
}
```

Sample Debug Output

```
hello
hello
hello
hello
hello
```

for-in loop

- here, index is a **constant** which is automatically set at the beginning of each loop iteration
- we don't need to declare it, that's done under the hood for us
- we can use index if we like!

```
for index in 1...5 {
    print("\(index)")
}
```

Sample Debug Output

```
1
2
3
4
5
```

for-in example

```
for index in 1...5 {
    print("The square of \(index) is \(index*index)")
}
```

Sample Debug Output

```
The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
The square of 4 is 16
The square of 5 is 25
```

arrays

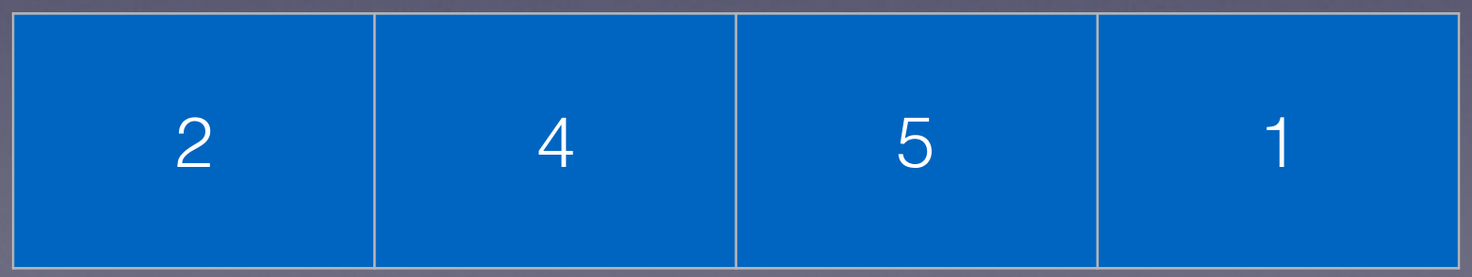
- Our data types so far have held one value, whether `Int`, `Double`, `String`, etc.

```
var card1 = 7
```


```
var card2 = 4
```

- An **array** is an ordered list of the same data type. Here, we declare an example array of 4 `Ints`. Its name is `cards`.

```
var cards = [2, 4, 5, 1]
```



array operations

- Examples of array declarations

```
// create an array with 4 Ints
var cards = [2, 4, 5, 1]

// create an empty array of type Int
var dealerCards = [Int]()

// create an array of type Double, with three
// elements, initialized to zeros
var threeDoubles = [Double](count: 3, repeatedValue: 0.0)
```

- We can access individual elements using square brackets

```
print("card in position 0 is \(cards[0])")
print("card in position 3 is \(cards[3])")
```

Sample Debug Output

```
card in position 0 is 2
card in position 3 is 1
```

array operations

- We can add elements to the **end** of the array using **append**

```
dealerCards.append(2)
dealerCards.append(8)
```

- We can get the number of items in an array by using **count**

```
print("cards has \(cards.count) elements")
print("dealerCards has \(dealerCards.count) elements")
```

Sample Debug Output

```
cards has 4 elements
dealerCards has 2 elements
```

array operations

- We can use a for-in loop to iterate over all elements in the array

```
print("our cards:")
for card in cards {
    print(card, terminator: " ")
}
print("")
```

Sample Debug Output

```
our cards:
2 4 5 1
```

```
cards[0] = 3
cards[3] = 8
print("our cards:")
for card in cards {
    print(card, terminator: " ")
}
print("")
```

Sample Debug Output

```
Sample Debug Output
our cards:
3 4 5 8
```

Exercise 3B

A. in your playground, create an array of four integers with values 1, 2, 3, and 4

B. change the value of the element in position 0 to 4

C. change the value of the element in position 3 to 1

D. try to change the value of the element in position 4. what happens?

E. append the value 11 to the end of your array

F. print each element in the array to the debug area

G. print the double of each element in the array

H. print the total of the all the values in the array

I. print the average (mean) value of all the values in the array (note: the answer has decimal places!)