SWIFT - 2

Ahmed Yossef

Agenda

- More on Variables
- More on Loops
- OOP
  - Classes
  - Objects
  - Methods

More on variables

Data Types

`var x = 5`	Compiler can guess
`var x:Int = 5`	No need to guess
`var x:Int`	This is fine
`var x`	This is **not** fine

Examples

`var x:Int = 5`	Defines an Int
`var x:float = 5.5`	Defines a float
`var x:Bool = true`	Defines a boolean
`var x:String = "Hello"`	Defines a String

Examples

```
let x:Int = 5
```

```
let x:float = 5.5
```

```
let x:Bool = true
```

```
let x:String = "Hello"
```

Same Applies for Constants

Collections

Arrays

```
var x = [2,3,4]
```

Array of integers

```
var x:[Int] = [2,3,4]
```

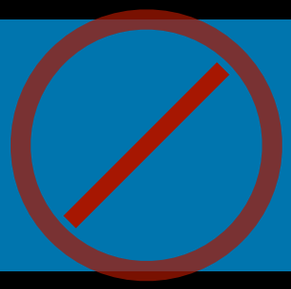
Array of integers

Collections

Arrays

```
var x = [2,"Ahmed",4.5]
```

Mixed Array



```
var x:[Int] = [2,"Ahmed",4.5]
```

It is not fine

Collections

Arrays

```
var x = [2,"Ahmed",4.5]
```

`x[0]`	2
`x[1]`	“Ahmed”
`x[2]`	4.5

Collections

Arrays

```
var x = [2,"Ahmed",4.5]
```

```
x[0] = 8
```

This will change the
first Element

Collections

Arrays

```
let x = [2,"Ahmed",4.5]
```

```
x[0] = 8
```

This is not working

let, defines a constant (Immutable) array !

Collections

Dictionaries

```
var x = ["1st":1,2:4.5,"3rd":"value"]
```

Dictionary [AnyObject: AnyObject]

Dictionary [String: Int]

```
var x:[String:Int] = ["k1":2,"k2":3,"k3":4]
```

Defined Key and Value dictionary

More on Loops

For Loop

```
for i in 1..<5
```

Loop i=1,2,3,4

For Loop

```
for i in 1...5
```

Loop i=1,2,3,4,5

For Loop

```
var drinks = ["coffee", "tea", "juice"]

for i in drinks
```

Loop i="coffee","tea","juice"

For Loop

```
for var i=0; i < 10 ; i++
```

Loop i=1,2,3,4,5,6,7,8,9

It will be deprecated soon !

OOP
Classes, Objects, Methods

Classes

```
class Human{}
```

Defines a new class Human

Classes

```
class Human{
var name = "";
var age = 0;
}
```

Defines a new class with variables

Classes

```
class Human{
func newFunction(){
}
}
```

Defines a new class with one function

Create Object

```
var ahmed:Human = Human();
```

Creates an object of the class Human

Accessing properties

```
var ahmed:Human = Human();
println(ahmed.name);
```

Prints the value of property name inside the human object called ahmed

Methods

```
func myMethod(){}
```

Method with no input nor output

Methods

```
func myMethod(x:Int){
}
```

Method with one parameter x

```
func myMethod(x:Int,y:Int){
}
```

Method with two parameters x,y

This how we define the parameters

Methods

```
func myMethod(x:Int,y:Int) ->Int {
}
```

The famous sum method

This how we define the return type

Methods

```
objectName.functionName(value1,value2)
```

Calling a method with two parameters

Inheritance

Define Parent

```
class Circle:Shape{}
```

We define the parent class using the :

Override

```
override func oldFunction(){}
```

use the word override to indicate overriding

Super

```
super.function()
```

Super is used to call the old implementation