Scala CheetSheet

Using an Actor is a lot like using a Thread in Java at a first glance –

```
■ import scala.actors._
  class MyActor extends Actor {        This is a Trait!
     def act {  // similar to run in a Java Thread
        // do something time consuming
     }
  }

■ object TestActors1 {
     def main(args: Array[String]) = {
        println("start it")
        val myActor = new MyActor()
        myActor.start()
        println("it ain't over til it's over…")
     }
  }

  val actor1 = actor {
     // do something
  }
```

Futures

```
val f: Future[String] =
                Future { Thread.sleep(1000); "Hello world!"; }
f.onComplete {
   case Success(value) => println("Callback processing: " + value);
    case Failure(value) => println("Failure"); }
```

Set up callback *(onSuccess or on Failure also available)*

```
while (!f.isCompleted) { Thread.sleep(2000) }
```

Wait for completion

Message passing in actors:
receiver ! message

```
val me = self        //self is like this in Java
me ! "Hello"         //asynchronous, fire and forget

receive{case x => print(x)}   // x matches everything

loop {                 // a special while(true) for Actors
  receive {
    case "Friday" => println("Thank God it's Friday!")
    case "Saturday" => exit
    case x => println("It's "+x+" and I'm working hard.")
  }
}
```