Variables and functions

```
var x = 10 //mutable
val y = 10 //immutable
def z = 10 //immutable but lazily evaluated

x = 15 //Yes
//y = 12 //compiler error
//z = 11 //compiler error
```

Conditions - everything is an expression

Control Shift P

```
if (x > 0) 10 else 100

val someInt = if (x > 0) 10 else 100

someInt
```

Defining functions

```
def addTwo(int1: Int, int2: Int): Int = {
  println("Int 1 " + int1)
  println(s"Int2 $int2")
  int1 + int2
}


addTwo(2, 3)
```

Add two - multiple function parameter lists - currying

```
def addTwoC(int1: Int)(int2: Int): Int = {
  println("Int 1 " + int1)
  println(s"Int2 $int2")
  int1 + int2
}

addTwoC(2)(3)
```

Loops - Generally frowned upon

//While loops

```
var l = 10
while (l < 20) {
  println(l)
  l = l + 1
}
```

Lazy values

```
lazy val lazyVal = {
  println("evaluating lazyVal")
  10
}

println(lazyVal)
println(lazyVal)


def defValue = {
  println("evaluating defVal")
  10
}

println(defValue)
println(defValue)
```

Exception handling

There's a better way to do this, which we'll talk about in the next discussion

```
def tryFun() {
  try {
    println("Trying hard here")
    throw new Exception("No luck though")
  }
  catch{
    case e:Exception => println (s"Gobbling up exception: ${e.getMessage}")
  }
  finally{
    println("Whatever man. Noone cares about me these days")
  }

}
```