CS 520 Advanced Programming Languages
Fall Semester, 2009
Doc 14 Scala Intro
Nov 5, 2009

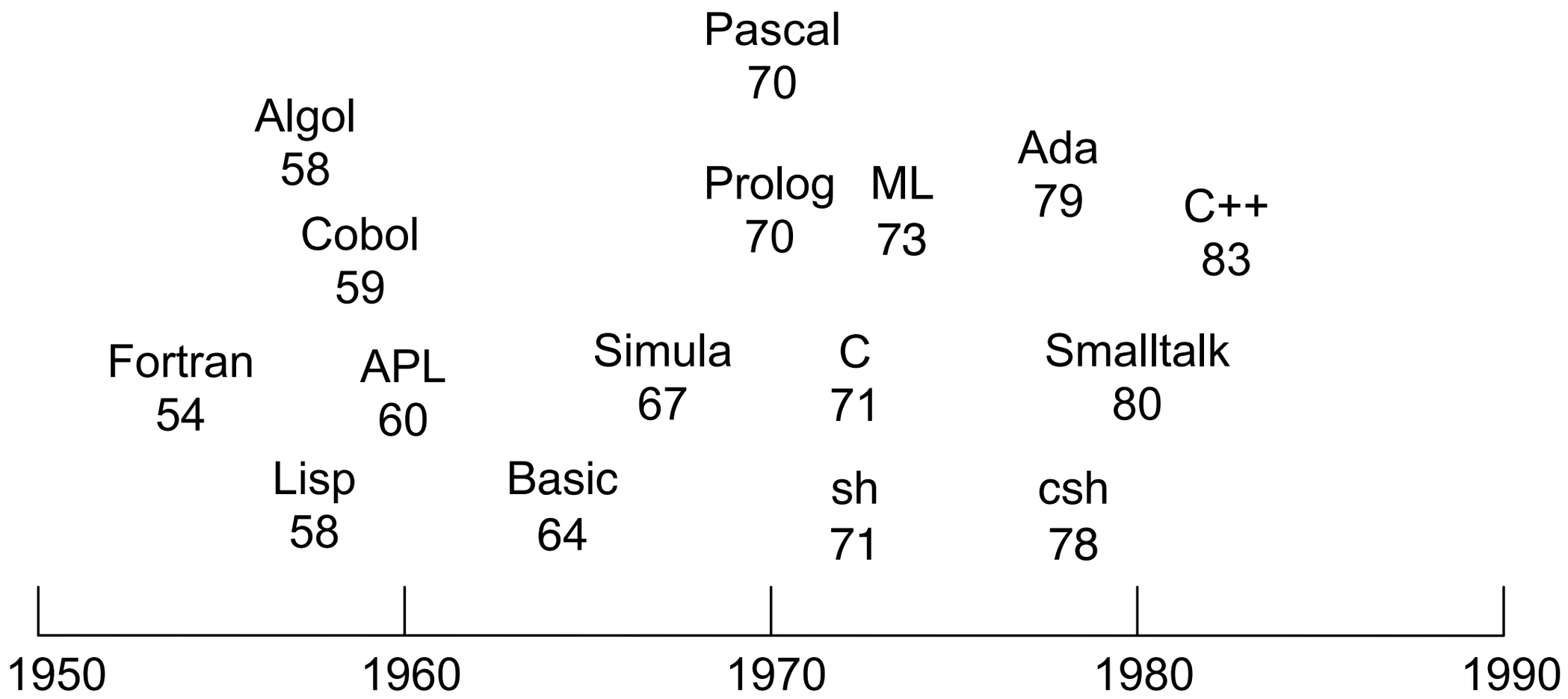
Reference

Programming in Scala, Odersky, Spoon, Venners, Artima Press, 2008

Reading

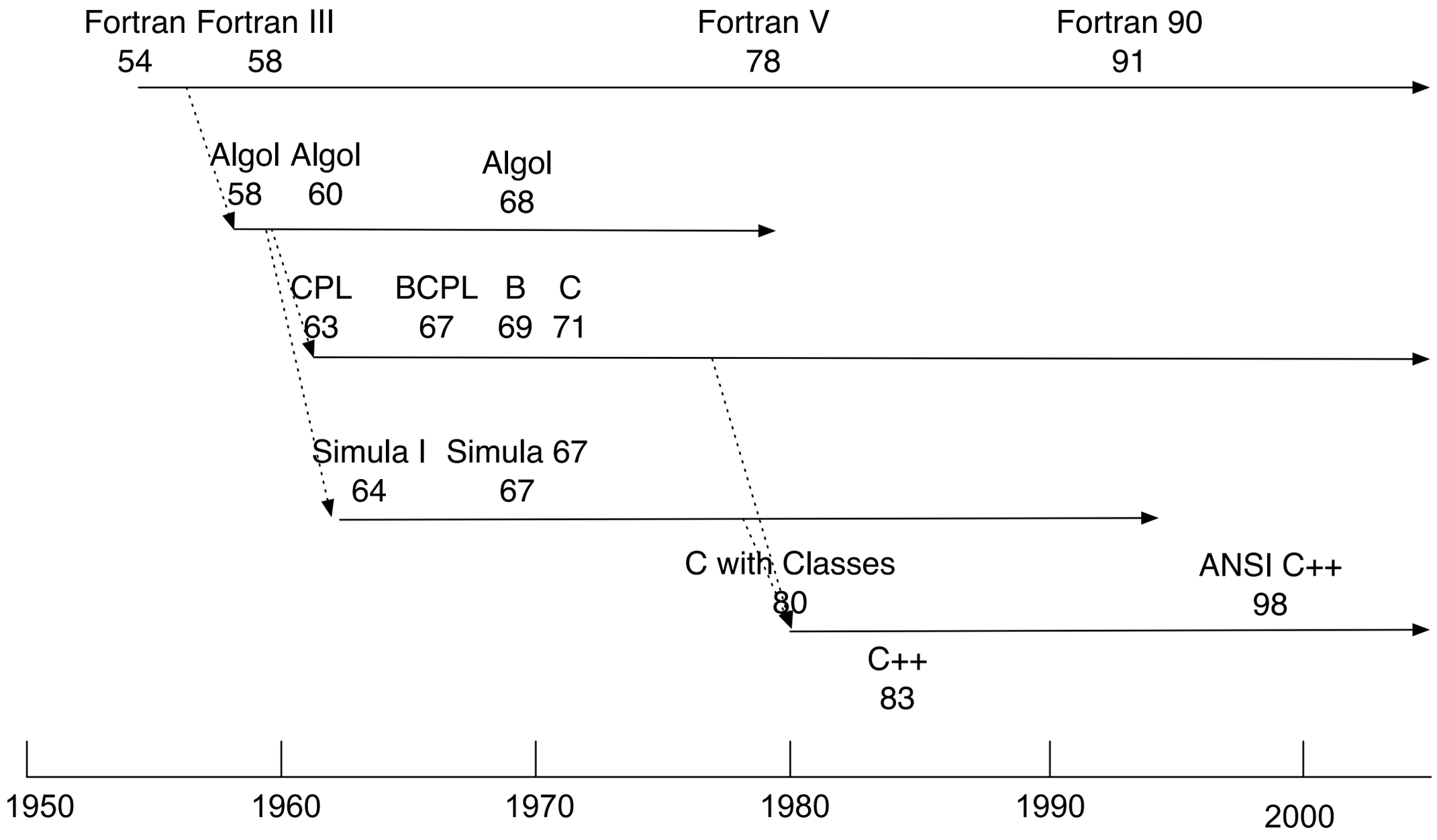
Chapters 2, 3, 5, 7

Some History

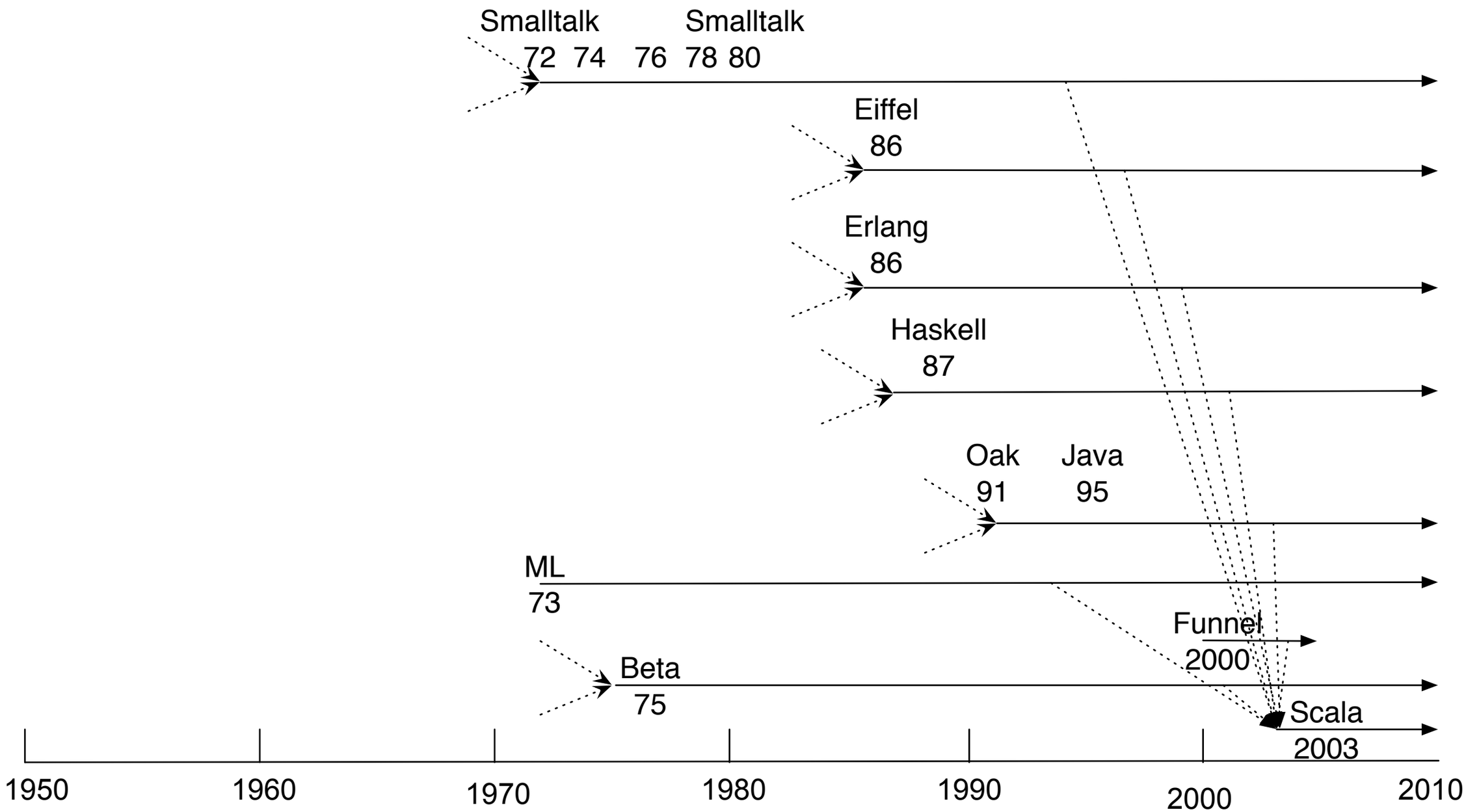


References: http://oreilly.com/news/graphics/prog_lang_poster.pdf, http://www.levenez.com/lang/, http://people.ku.edu/~nkinners/LangList/Extras/langlist.htm

C++ Influences



Scala Influences



Smalltalk – uniform object model; Eiffel – uniform access principle; Erlang – actor-based concurrency; Haskell – implicit parameters; Java – syntax, types, class library, execution model (VM); ML – functional programming, higher order functions; Beta (Algol, Simula, gbeta) – universal nesting; Funnel – don't create minimal languages

Functional Programming

Use functions to compute values

No side-effect allowed

Functions are values

Scala

http://www.scala-lang.org/

Current Version 2.7.7

Functional Programming
Object-Oriented Programming
Scripting
Really strongly typed
Implicit types
Interpreter & compiler

Runs on top of Java
Compiles to Java byte code
Scala can call Java code
Java can call Scala code

Installing

http://www.scala-lang.org/downloads

Requires Java 1.5 or higher

Following instructions on the download page

Sample Interpreter Use

```
AI pro 35->scala
Welcome to Scala version 2.7.7.final (Java HotSpot(TM) Client VM, Java 1.5.0_19).
Type in expressions to have them evaluated.
Type :help for more information.

scala> 1 + 2
res0: Int = 3

scala> println("Hello")
Hello

scala> res0 *5
res2: Int = 15

scala>
```

Semicolons Optional

```
scala> 1 + 2;
res3: Int = 3

scala> 1 + 2; println("hi")
hi

scala> :quit
Al pro 36->
```


Scala infers the end of a statement and adds the semicolon. If you place two statements on the same line you have to separate them with a semicolon

Script

sample.scala

```
var x: Int = 2
var y = 3
val result = x + y
print("We have " + x + " + " + y)
println( " is " + result)
```

Running the Example

```
Al pro 48->scala sample.scala
We have 2 + 3 is 5
```

Type Inference

```
var x: Int = 2
var y = 3
```

var verses val

```
var y = 3
val result = x + y
```

var - variable

mutable

can change

```
var y = 3
y = 5
```

val - value

immutable

can not change

```
val z = 3
x = 5       //Error
```

Loading Files in Interpreter

sample.scala

```
var x: Int = 2
var y = 3
val result = x + y
"We have " + x + " + " + y + " is " + result
```

```
scala> :load sample.scala
Loading sample.scala...
x: Int = 2
y: Int = 3
result: Int = 5
res0: java.lang.String = We have 2 + 3 is 5
```

Scripts Verses Programs

Script

```
var x = 2
var y = 3
println(x + y)
```

Program (application)

```
object RunMe {
  def main(args : Array[String]) : Unit = {
    var x = 2
    var y = 3
    println(x + y)
  }
}
```

Scala & Eclipse

Eclipse runs Scala programs
Create Scala Application

Installing Scala in Eclipse

Install Scala
Set path to inlcude scala bin

Install Scala Eclipse plugin

Defining Functions

```
def sum(x: Int, y: Int): Int = {
        return x + y
}

def min(x: Int, y: Int): Int = {
        if (x > y)
                y
        else
                x
}

def max(x: Int, y: Int) = if (x > y) x else y

println( min(2,3))
println( max(2,3))
println( sum(2,3))
```

factorial

```
def factorial(n: BigInt): BigInt =
      if (n == 0) 1 else n * factorial(n -1)

println( factorial(200))
```

Output

```
788657867364790503552363213932185062295135977687173263294742533244359449963403342920304284011984623904177212138919638830257642790242637105061926624952829931113462857270763317237396988943922445621451664240254033291864131227428294853277524242407573903240321257405579568660226031904170324062351700858796178922227896237038973747200000000000000000000000000000000000000000000000
```

Why type ()?

silly.scala

```
def silly():Int = {   1 +2 }

silly()
silly

12.toString()
12.toString
```

```
scala> :load silly.scala
Loading silly.scala...
silly: ()Int
res4: Int = 3
res5: Int = 3
res6: java.lang.String = 12
res7: java.lang.String = 12
```

Unit

```
def greeting(): Unit = {
      println("hello World")
}
```

like Java/C/C++ void

```
def greeting() {
      println("hello World")
}
```

Basic Types

Boolean
Byte
Char
Double
Float
Int
Long
Short
String

range is same as Java's types

Symbol

BigInt (java.math.BigInteger)
BigDecimal (java.math.BigDecimal)

Literals

```
val hex = 0x5
val oct = 035
val decimal = 35
val long = 35L
var double = 3.5
double = 1.23e2
```

```
val float = 1.234F
val char = 'Q'
val unicode = '\u0044'
val boolean = true
val string = "Hi mom"
val symbol = 'whatIsThis
```

Implicit Conversions

```
scala> var y: Int = 'c'
y: Int = 99

scala> var w:Char = 123
w: Char = {

scala> var x: Long = 23
x: Long = 23
```

```
scala> var w:Char = 123L
<console>:4: error: type mismatch;
 found   : Long(123L)
 required: Char
       var w:Char = 123L
                    ^

scala> var x: Int = 23L
<console>:4: error: type mismatch;
 found   : Long(23L)
 required: Int
       var x: Int = 23L
                    ^
```

==

Java

int x;
int y;
y == x;
Are x & y the same value

Object x;
Object y;
x == y;
Do x & y point to the same object

Scala

int x;
int y;
y == x;
Are x & y the same value

Object x;
Object y;
x == y;
Do the objects x & y have same value

No increment operator

```
var k = 1
k++                    //Compile error
```

Must initialize Variables

```
scala> var test: Int;
<console>:4: error: only classes can have declared but undefined members
(Note that variables need to be initialized to be defined)
       var test: Int;


scala> var test: Int = 5;
test: Int = 5
```

Any

```
var x: Any = 5
x = "cat"
x = 1.23
x = true
x = 'c'
```

```
var y: Int = 5
y = "cat"       //Compile error
```

Arrays

```
val message = new Array[String](3)
message(0) = "This"
message(1) = " is "
message(2) = " not java"

for (k <- 0 to 2)
      print(message(k))
println
```

Output

This is not java

Arrays & foreach

```
val newArray = Array[String]("cat", "dog", "mouse")
val inferredType = Array("rat", "mat")
val both = newArray ++ inferredType

both.foreach((element: String) => println(element))
both.foreach(element => println(element))
both.foreach(println)
```

Arrays & Mutability

Arrays can not change size

Arrays elements can change

Lists

immutable

```
val newList = List[String]("cat", "dog", "mouse")
val inferredType = List("rat", "mat")
val append = newList ++ inferredType
val prepend = newList ::: inferredType

prepend.foreach(println)
```

```
scala> :load sample.scala
Loading sample.scala...
newList: List[String] = List(cat, dog, mouse)
inferredType: List[java.lang.String] = List(rat, mat)
append: List[String] = List(cat, dog, mouse, rat, mat)
prepend: List[java.lang.String] = List(cat, dog, mouse, rat, mat)
cat
dog
mouse
rat
mat
```

Lists, append, prepend

Lists have head and tail

prepend
takes O(1) time

append
takes O(n) time

Prepend

sample.scala

```
val listA = 1 :: 2 :: 3 :: Nil
val listB = 4 :: List(5, 6)
val listC = listA ::: listB
val listD = listA :: listB
```

```
scala> :load sample.scala
Loading sample.scala...
listA: List[Int] = List(1, 2, 3)
listB: List[Int] = List(4, 5, 6)
listC: List[Int] = List(1, 2, 3, 4, 5, 6)
listD: List[Any] = List(List(1, 2, 3), 4, 5, 6)
```

a :: b

prepend a to list b

returns new list

a becomes the head of new list

a ::: b

prepend list a to list b

returns new list

Control Structures

If

```
def example(x: Int, y: Int): Int {
    var min = 0;
    if (x < y)
        min = x
    else
        min = y
    return min
}
```

```
def example(x: Int, y: Int):Int = {
    val min =
        if (x < y)
            x
        else
            y
    return min
}
```

if else Problems

OK

```
val x = 2;
val y = 3;
var min = y;
if (x < y)
      min = x
```

```
if (x < y) min = x else min = y
```

Compile Error

```
val x = 2;
val y = 3;
var min = 0;
if (x < y)
      min = x
else
      min = y
```

Recommended Style

```
val x = 2;
val y = 3;
val min = if (x < y)  x else y
```

While

```
var a = 10
var b = 6;
while (a != 0) {
        val temp = a
        a = b % a
        b = temp
}
println(b)
```

do - while

```
var line = ""
do {
        line = readLine()
        println("read: " + line)
} while (line != "")
```

C/C++/Java while idom does not work

```
var line = ""
while ((line = readLine()) != "")
      println("read: " + line)
```

```
line: java.lang.String =
<console>:6: warning: comparing values of types Unit and java.lang.String
using `!=' will always yield true
      while ((line = readLine()) != "")
```