CSE250 Week 3 Recitations

Program Arguments and Tuples and Max

Kenneth W. Regan and TAs

February 12, 2022
You can give as many arguments to `main` as you wish.

They are automatically stored in an array of strings usually called `args`.

The first one is stored as `args(0)`, and so on if you have more.

The number of arguments given can be read by `args.length`.

On IntelliJ, you can right-click in your code and select “Modify Run Configuration...” Type the arguments you want in order into the “Program Arguments” box.

That has the same effect as e.g. `scala Foo arg0 arg1 arg2` on the UNIX/Linux command line.

For just one argument, `scala MaxWords Hamlet.txt` has the same effect as typing `Hamlet.txt` into your “Program Arguments” box, assuming the file is in your project root folder.
The prescribed lines for the main object are the simplest form of a common idiom (infile is short for inputFile):

```scala
object MaxWords extends App {
  val infile = if (args.length >= 1) args(0) else "words.txt"

  // If you had, say, three items foo1, foo2, foo3, each optional and with default values bar1, bar2, and bar3, then you could use a series of tests:

  val foo3 = if (args.length >= 3) args(2) else bar3
  val foo2 = if (args.length >= 2) args(1) else bar2
  val foo1 = if (args.length >= 1) args(0) else bar1

  // (If you have a lot of program arguments they should be keyword arguments prefaced by one or two - signs and/or a + sign, called "switches." Not a concern in this course.)
```
More About Tuples

- Tuples can give a convenient way to return multiple values from a function.
- Scala has no return keyword, so you just make the tuple be the last “statement” in the body.

```scala
class Rectangle(var len: Double, var width: Double) {
  def area = len*width
}
def bigger(r1: Rectangle, r2: Rectangle) = {
  val bgr = if (r1.area > r2.area) r1 else r2
  val diff = (r1.area - r2.area).abs //absolute value as method
  (bgr, diff) //return diff in area too; Scala infers tuple type
}
val rect1 = new Rectangle(2.0, 7.0)
val rect2 = new Rectangle(5.0, 3.0)
val (r, d) = bigger(rect1, rect2)
```
A reminder that “tupled assignment” doesn’t work like in Python, even with \texttt{var (r,d)}:

\begin{verbatim}
var (r,d) = bigger(rect1,rect2)
val rect3 = new Rectangle(4.0,4.0)
(r,d) = bigger(rect1,rect3)  //error
\end{verbatim}

But OK is to “re-declare” \texttt{var (r,d) = bigger(rect1,rect3)}, and even to re-declare it as \texttt{val (r,d) = ...} instead. But maybe better is to name the tuple:

\begin{verbatim}
var t = bigger(rect1,rect2)
val rect3 = new Rectangle(4.0,4.0)
t = bigger(rect1,rect3)  //fine
println(s"The bigger rectangle is \${t._1}")
\end{verbatim}
Levels of Emptiness

- Suppose we do `import io.Source` and in the code body do
  ```scala
  val ell = Source.fromFile("words.txt").getLines().toList
  ```
- If the file `words.txt` does not exist, we get an error.
- But suppose it exists as an empty file, with zero lines. What then?
  - **Answer:** `val ell: List[String] = List()` This is the *empty list*.
- Now suppose we give it one blank line, so `wc` says 1 line of 0 chars
  - **Answer:** `val ell: List[String] = List("")`
- This is the *list of one element which is an empty string*.
- Now what if we split an empty line on whitespace:
  ```scala
  val arr = ".split("\s+")
  ```
  - We get `val arr: Array[String] = Array("")`
  - Note `arr.length` gives 1. It is a nonempty array.
  - But `arr.drop(1)` gives `Array()`, the empty array.
Edge Cases For Max

- The maximum length of a string in the array `Array("")` is clearly 0, because the empty string is a string in the array and `"".length` equals 0.
- What should be the maximum length of a string in the empty array?
  - **Idea 1**: Use the same default of 0. But could confuse the two cases.
  - **Idea 2**: Use the option type: `Some(0)` in the first case, `None` in the empty-array case. But can make the code clunky.
  - **Idea 3**: Use a different default, such as -1. This is a clearly out-of-bounds value.

( Hey: Scala does not have an unsigned integer type. They rejected a proposal for one. Good—IMPHO, the C++ unsigned integer `size_t` type is an error-fraught boondoggle.)
Besides returning the max value, also consider:

- **argmax**: the element giving the maximum value.
- **indexmax**: the index of that element in the array or list.

```scala
def indexMax(arr: Array[String]): Int = {
  var (max, indexmax) = (-1, -1)
  for (i <- 0 until arr.length) { // until is exclusive
    if (arr(i).length > max) {
      max = arr(i).length;
      indexmax = i
    }
  }
  indexmax
}
```

Doesn’t do all you need, but shows some ideas.