Lecture 09 - Variables, functions, data structures, and input/output

Scripting and commenting

Remember we are *scripting* here, so we should be working in scripts!

Scripts also allow us to do a lot of commenting to help our collaborators and our future selves.

Building blocks of a scripting language

- variables
- functions

What is a variable?

What is a function? What are the two things you need to know to use a function?

What's a variable?

- a name that holds a value
- attributes of a variable:
  - name
  - mode or type
  - value
  - scope

- had variables in shell too
- in fact Exercise 4 uses some variables. Anyone struggling with that?

variable modes or types

Our definition of a variable is very flexible, but a scripting language has to define a base set of variable types or modes.

Data modes in R:

- numeric
- complex
- character
- logical
- factors

Data types in Python:

- integer
- float
- complex
- string
- boolean

Assignment of a value to a variable

Assignment of a value to a variable can be accomplished with = in both `Python` and `R`. Some like to use <- in `R`, but they both work.

The simplest example of this is assigning a single numeric or character value to a variable name, but the value returned from a function call can also be assigned to a variable.

```
x=5
name<-"Stuart"
curTime=date()
```

```
x=5
name="Stuart"

from datetime import datetime
curTime=datetime.now().
strftime('%Y-%m-%d %H:%M:%S')
```

Calling a function

- still have function names and arguments
- back to learning syntax. . .

Where do we learn about functions?

Calling a function

- still have function names and arguments
- back to learning syntax. . .

Where do we learn about functions?

`?function_name` `help(function_name)`

Calling a function

In unix we would call a function using it's name with arugment following separated by spaces. This is a bit different in `R` and `Python`.

`R` functions are called with their name followed by an open and close parenthese. Arguments are passed within the parentheses and separated by commas.

```
x = c(1,2,3,4)
y=mean(x,na.rm=TRUE)
```

Base functions are called as `variable.function()` with additional arguments separated by commas within the parentheses. Other functions are called with the variable passed as an argument.

```
x = [1,2,3]
x.append(4)
import numpy
y=numpy.mean(x)
```

Data structures

We usually want to work with more than one numeric or character value at a time and so a variety of data structures exist to hold multiple values in a single variable.

- vector: ordered 1D, contains single data mode
- matrix/array: 2D/nD, contains single data mode
- list: sequence, contains multiple data modes and other data structures
- dataframe: 2D, hybrid of list and matrix

- list: ordered, mutable; `[]`
- tuple: ordered, immutable; `()`
- dictionary: unordered key-value pairs, mutable; `{}`
- set: unordered, immutable, only unique entries
- matrix and arrays in `Numpy` (only single data type here)

Reading and writing files

The power of scripting languages is their ability to process large amounts of data, so it makes sense that we can access the contents of text files.

To read tabular data:

```
read.table(file='filename'
,sep=',',header=TRUE)
```

To read text data:

```
scan(file='filename')
```

To read tabular data:

```
import numpy
numpy.loadtxt(fname=
'filename',delimiter=',')
```

To read text data:

```
with open(filename) as f:
   content = f.readlines()
```

Indexing from data structures

Indexing is a means by which we can access a subset of an array, matrix, vector, etc.

We can use square brackets `[]` in both languages to subset many data structures.

Always remember: rows before columns

A major difference between the two languages is the initial index.

+ `` `Python` starts with zero ``

+ `` `R` starts with one ``

We can also index multiple rows or columns using `:`

Indexing unordered structures

Lists in `R` and dictionaries in `Python` are unorderd and must be indexed by their *keys*.

To get an element in an `R` list, use dollarsign notation:
`list$name1`

`R` lists actually aren't unordered and so you can also use double brackets:
`list[[1]]`

To get an element in a `Python` dictionary we use square brackets:

```
dict = {'key':'value'}

dict['key']
```