Lecture 10 - Making choices and some practice

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 and ends non-inclusive

+ `R` starts with one and ends inclusive
```

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']
```

Control structures

Our programs can become much more flexible and powerful if the process can proceed in multiple directions depending on characteristics of the input data or other arguments.

Code that allows progression of the program among one of multiple possible paths are often called **control structures**.

We have seen one control structure when we used *for loops* in Unix.

If-Else statements, what SWC refers to as "making choices" is another example of a control structure.

Making choices with If-Else statements

If-Else statements use logic tests to make context-dependent decisions.

General syntax is:

```
if (logic test)
    do some stuff
else
    do some other stuff
```

Making choices with If-Else statements

What about if we have three conditions?

What if we have four conditions?

What if we have a decision tree where three conditions are nested within each of two higher level conditions?

What if we only care about the case where the logic test is true?

Remind me of the syntax for If-Else statements. . .

Remind me of the syntax for If-Else statements. . .

Both languages follow the general syntax:

```
if(x==1){
  print("x is 1")
}else if(x==2){
  print("x is 2")
}else{
  print("x isn't 1 or 2")
}
```

```
if x==1:
  print("x is 1")
elif x==2:
  print("x is 2")
else:
  print("x isn't 1 or 2")
```

Logic tests

```
x==1

x!=1

x>1

x<1

x>=1

x<=1
```

Python uses `and`/`or` to indicate more complex logic tests, but R uses `&`/|

If-Else statements are most useful in loops

A loop allows us to use an If-Else statement to consider many different scenarios and evaluate the condition for each scenario.

Remind me how loops worked in Unix. . .

For loops in Python and R

For loops are conceptually similar regardless of the language, so you already know how to use them.

You only need to know the specific syntax for your language!

```
for(i in 1:10){
  do some stuff
}
```

```
for i in xrange(0,10,1):
  do some stuff
```

Practice with control structures

Challenge: Use a `for loop` and `if-else` statement to calculate the sum of wages for males and females in `wages.csv`.

Practice with control structures

Challenge: Use a `for loop` and `if-else` statement to calculate the sum of wages for males and females in `wages.csv`.

Challenge: Adjust your code to use the `for loop` and `if-else` statement to calculate the sum and the mean of wages for males and females in `wages.csv`.