Week 5: Loops

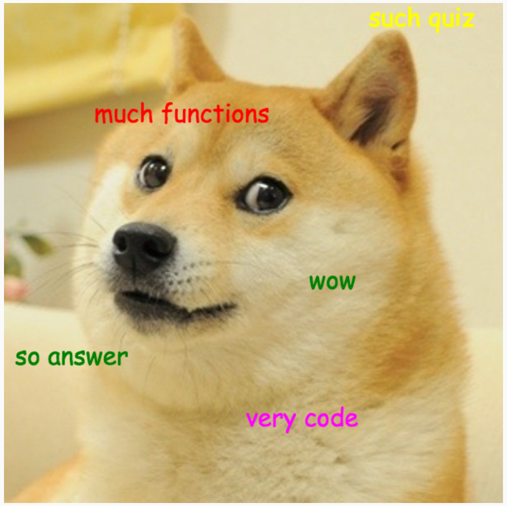
EMSE 6574, Section 11

John Helveston

September 23, 2019

Quiz 2

20 minutes



- No calculators
- No notes
- No books
- No computers
- No phones

Announcements

1) What's with this `+` sign in the console?

2) Do you like to draw? Leave your mark on EMSE 6574 by helping design a hex sticker!

See http://hexb.in/ for dimensions, etc.

"Flow Control"

Flow control is code that alters the otherwise linear flow of operations in a program.

Last week:

- `if` statements
- `else` statements

This week:

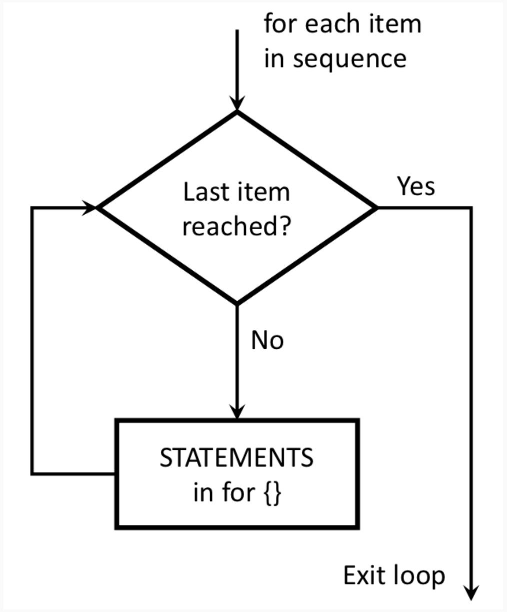
- `for` loops
- `while` loops
- `break` statements
- `next` statements

The `for` loop

Basic format:

```
for (VALUE in SEQUENCE) {
  STATEMENT1
  STATEMENT2
  ETC
}
```

Here's the general idea:



Making a sequence

(Side note: these are vectors...that's next week - read ahead!)

Two ways to make a sequence (there are more):

1. Use the `seq()` function

2. Use the `:` operator

```
seq(1, 10)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
seq(1, 10, 2)
```

```
## [1] 1 3 5 7 9
```

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

Sequences don't have to be integers

```r
1.5:5.5
```

```
## [1] 1.5 2.5 3.5 4.5 5.5
```

```r
seq(1.2, 6, 0.4)
```

```
##  [1] 1.2 1.6 2.0 2.4 2.8 3.2 3.6 4.0 4.4 4.8 5.2 5.6 6.0
```

You can also loop over a vector of strings (more on vectors next week!)

```r
for (i in rep('oh hai!', 3)) {
    cat(i, '\n')
}
```

```
oh hai!
oh hai!
oh hai!
```

Practice: What will this return?

60 seconds - no typing!

```r
for (i in 1:5) {
    if ((i %% 2) == 0) {
        cat('--')
    } else if ((i %% 3) == 0) {
        cat('----')
    }
    cat(i, '\n')
}
```

```
1
--2
----3
--4
5
```

Practice: What will this return?

60 seconds - no typing!

```r
n <- 6
for (i in seq(n)) {
    cat('|')
    for (j in seq(1, n, 2)) {
        cat('*')
    }
    cat('|', '\n')
}
```

```
|***|
|***|
|***|
|***|
|***|
|***|
```

Group practice: sum from `m` to `n`

20 minutes - In groups of 4, write the following functions:

1) `sumFromMToN(m, n)`: Write a function that sums the total of the integers between `m` and `n`.
Challenge: Try solving this without a loop (it's possible - Google it!).

- `sumFromMToN(5, 10) == (5+6+7+8+9+10)`
- `sumFromMToN(1, 1) == 1`

2) `sumEveryKthFromMToN(m, n, k)`: Write a function to sum every kth integer from `m` to `n`.

- `sumEveryKthFromMToN(5, 20, 7) == (5 + 12 + 19)`
- `sumEveryKthFromMToN(1, 10, 2) == (1 + 3 + 5 + 7 + 9)`
- `sumEveryKthFromMToN(0, 0, 1) == 0`

3) `sumOfOddsFromMToN(m, n)`: Write a function that sums every *odd* integer between `m` and `n`.
Challenge: Try solving this without a loop (Hint: use a vector operation...we'll cover this next week!).

- `sumOfOddsFromMToN(4, 10) == (5 + 7 + 9)`
- `sumOfOddsFromMToN(5, 9) == (5 + 7 + 9)`

5 minute break - stand up, move around,

5 minutes

`break` and `next`

`break`

Note: `break` doesn't require `()`

Forces a loop to stop and "break" out of the loop.

```
for (val in 1:5) {
    if (val == 3) {
        break
    }
    cat(val, '\n')
}
```

```
1
2
```

In a nested loop:

```
for (i in 1:3) {
  cat('*')
    for (j in 1:3) {
        if (j == 3) {
            break
        }
        cat(j, '\n')
    }
}
```

```
*1
2
*1
2
*1
2
```

`break` and `next`

`next`

Skips to the *next* iteration of a loop

```
for (val in 1:5) {
    if (val == 3) {
        next
    }
    cat(val, '\n')
}
```

```
1
2
4
5
```

In a nested loop:

```
for (i in 1:3) {
  cat('*')
    for (j in 1:3) {
        if (j == 2) {
            next
        }
        cat(j, '\n')
    }
}
```

```
*1
3
*1
3
*1
3
```

Lame joke time - the `while` loop

A friend calls her programmer roommate and says, "while you're out, buy some milk"...

...her roommate never returned home.

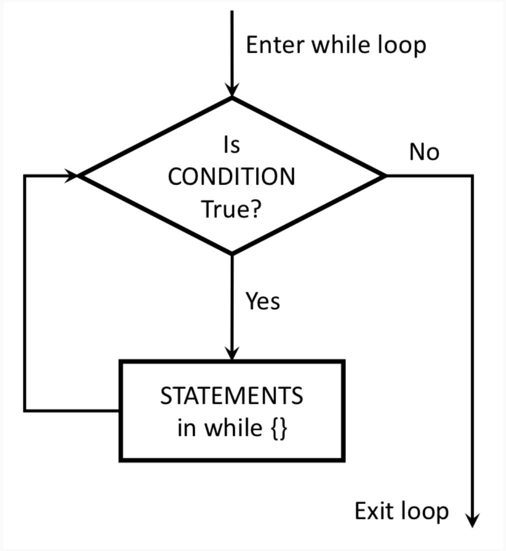
🤣

The `while` loop

Basic format:

```
while (CONDITION) {
  STATEMENT1
  STATEMENT2
  ETC
}
```

Here's the general idea:



`for` vs. `while`

Use `for` loops when there is a *known* number of iterations.

Use `while` loops when there is an *unknown* number of iterations.

Practice: What will this return?

60 seconds - no typing!

```r
f <- function(x) {
    n = 1
    while (n < x) {
        cat(n, '\n')
        n = 2*n
    }
}
```

```r
f(5)
f(10)
f(50)
```

```r
f(5)
```

```
## 1
## 2
## 4
```

```r
f(10)
```

```
## 1
## 2
## 4
## 8
```

```r
f(50)
```

```
## 1
## 2
## 4
## 8
## 16
## 32
```

Group practice

20 minutes - In groups of 4, write the following functions:

1) `isMultipleOf4Or7(n)`

Write a function that returns `TRUE` if `n` is a multiple of 4 or 7 and `FALSE` otherwise. Here's some test cases:

- `isMultipleOf4Or7(0) == FALSE`
- `isMultipleOf4Or7(1) == FALSE`
- `isMultipleOf4Or7(-7) == FALSE`
- `isMultipleOf4Or7(4) == TRUE`
- `isMultipleOf4Or7(7) == TRUE`
- `isMultipleOf4Or7(28) == TRUE`
- `isMultipleOf4Or7('notANumer') == FALSE`

2) `nthMultipleOf4Or7(n)`

Write a function that returns the nth positive integer that is a multiple of either 4 or 7. Hint: use `isMultipleOf4Or7(n)` as a helper function! Here's some test cases:

- `nthMultipleOf4Or7(1) == 4`
- `nthMultipleOf4Or7(2) == 7`
- `nthMultipleOf4Or7(3) == 8`
- `nthMultipleOf4Or7(4) == 12`
- `nthMultipleOf4Or7(5) == 14`
- `nthMultipleOf4Or7(6) == 16`
- `nthMultipleOf4Or7(10) == 28`

Group practice - Prime Numbers

20 minutes - In groups of 4, write the following functions:

1) `isPrime(n)`

Write a function that takes a non-negative integer, `n`, and returns `TRUE` if it is a prime number and `FALSE` otherwise. Here's some test cases:

- `isPrime(1) == FALSE`
- `isPrime(2) == TRUE`
- `isPrime(7) == TRUE`
- `isPrime(13) == TRUE`
- `isPrime(14) == FALSE`

2) `nthPrime(n)`

Write a function that takes a non-negative integer, `n`, and returns the nth prime number, where `nthPrime(1)` returns the first prime number (2). Hint: use `isPrime(n)` as a helper function! Here's some test cases:

- `nthPrime(1) == 2`
- `nthPrime(2) == 3`
- `nthPrime(3) == 5`
- `nthPrime(4) == 7`
- `nthPrime(7) == 17`