2. Loops (through a sequence, 0 or more times, or 1 or more times)

Iterate through a sequence (vector or list)

This `for` loop runs `EXPRESSION` for each value `VARIABLE` in `SEQUENCE`. (`UPPER.CASE` text is a placeholder for R code.)

```
for (VARIABLE in SEQUENCE) {
  EXPRESSION
}
```

e.g.

```
for (file.name in list.files()) {
  cat("file.name =", file.name, "\n") # ... then do something with the file
}

n = 3 # also try 0
for (i in seq_len(n)) { # SEQUENCE is 1:n (or, for n=0, integer(0))
  even.or.odd = ifelse(i %% 2 == 0, "even", "odd")
  cat(i, "is", even.or.odd, "\n")
}
```

Here are canonical sum and product loops. (But use `sum()` and `prod()` when you can.)

```
x = c(2, 3, 5)
total = 0
for (x.i in x) { # loop through values
  total = total + x.i
}
```

```
product = 1
for (i in seq_len(length(x))) { # or indices
  product = product * x[i]
}
```

Loop zero or more times

This `while` loop runs `EXPRESSION` as long as `CONDITION` is true.

```
while (CONDITION) {
  EXPRESSION
}
```

e.g.

```
x = 1
while (x < 10) {
  cat("x =", x, "\n")
  x = 2*x
}
```

Loop one or more times

This loop runs `EXPRESSION` once and then repeats until `CONDITION` is true.

```
repeat {
  EXPRESSION
  if (CONDITION) {
    break
  }
}
```

e.g. Prompt for user input until user cooperates:

```
repeat {
  cat("Please answer 'yes' or 'no':")
  decision = scan(what=character(), n=1, quiet=TRUE) # ?scan
  if ((decision == "yes") | (decision == "no")) {
    break
  }
}
```

Background:

- Loop forever (usually a bad idea):

  ```
  repeat {
     EXPRESSION
  }
  ```

- Break out of a loop with `break`, usually guarded by a condition:

  ```
  if (CONDITION) {
    break
  }
  ```

- Skip to the bottom of a loop (still inside it) with `next`:

  ```
  if (CONDITION) {
    next
  }
  ```

Code formatting tips

- "`{`" does not get a new line
- "`}`" is on a line by itself, indented like the line containing the corresponding "`{`"
- code inside braces is indented two spaces
- In RStudio, use "Code > Reindent Lines"