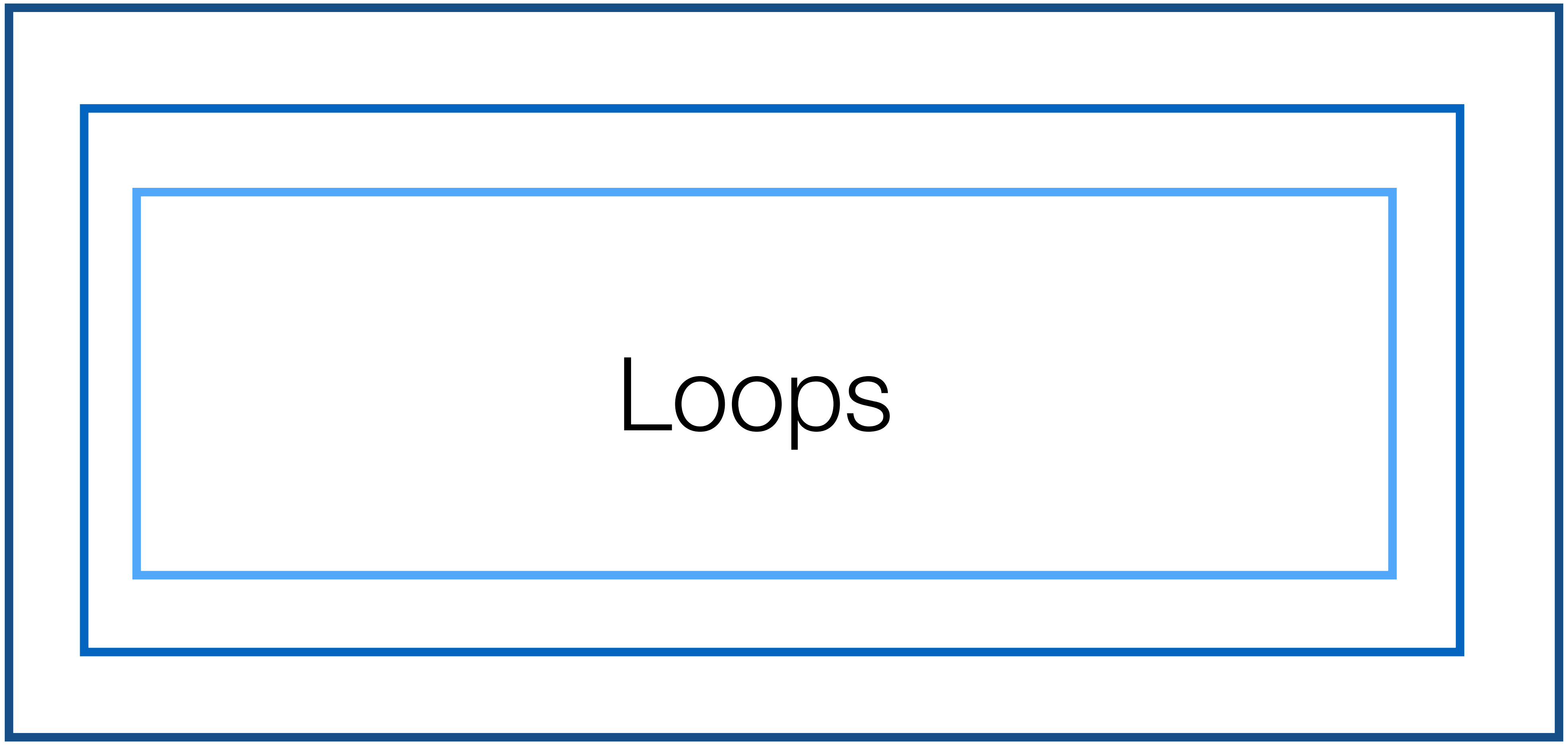
R Bootcamp Part 3

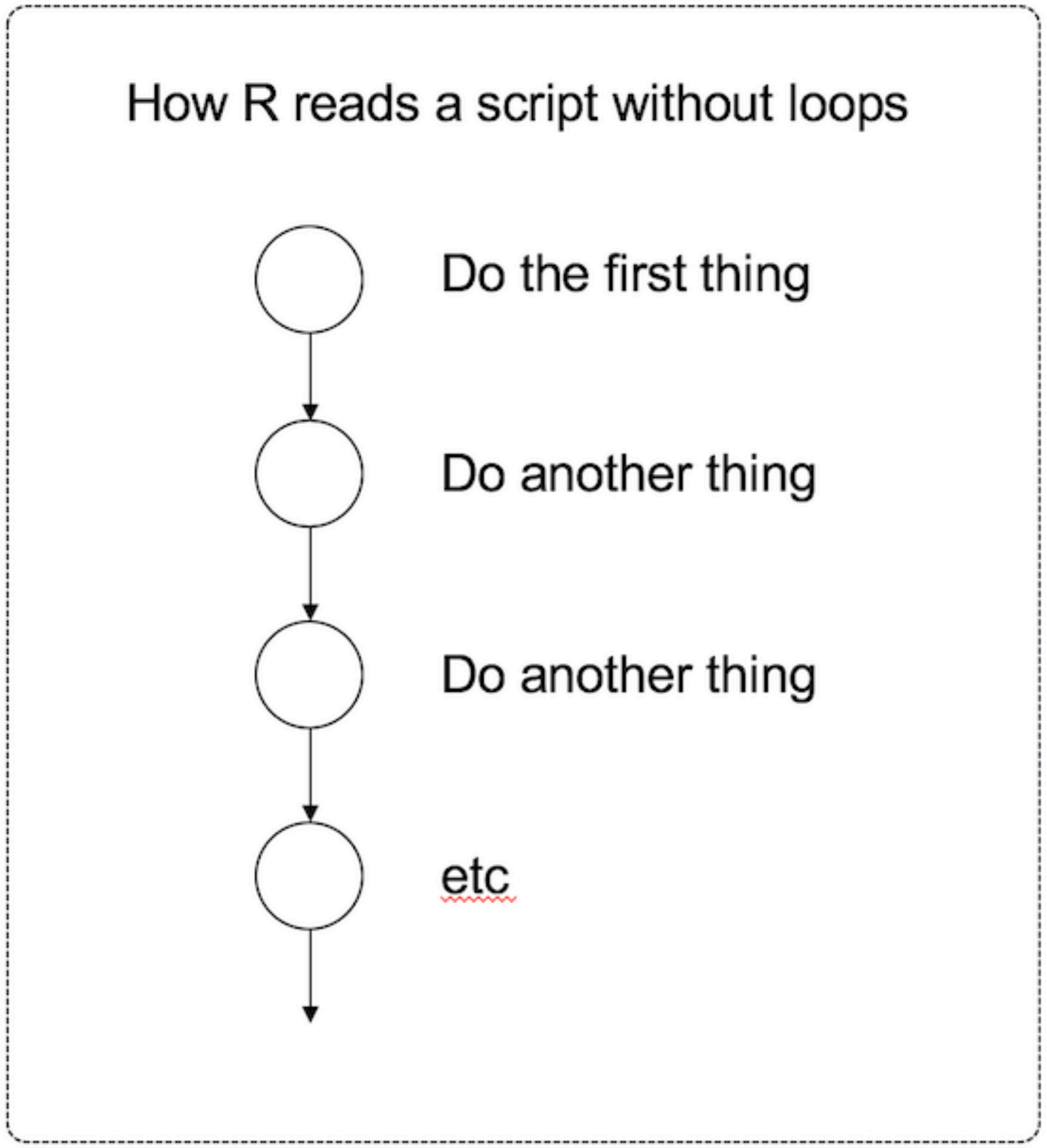
Amy Perfors

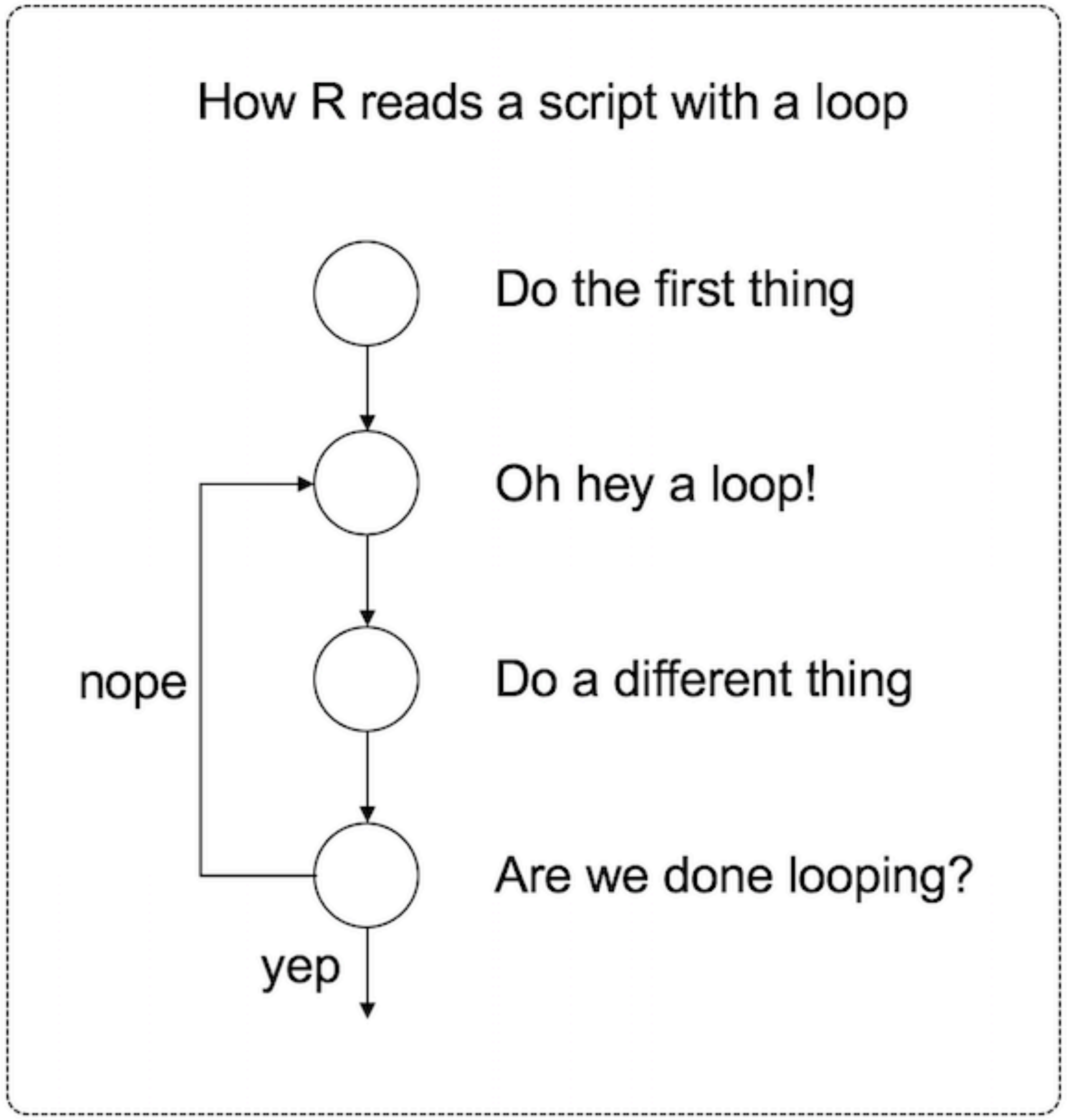
Plan

1. Loops
2. Branches
3. Functions


Loops

The purpose of a loop





While loops

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

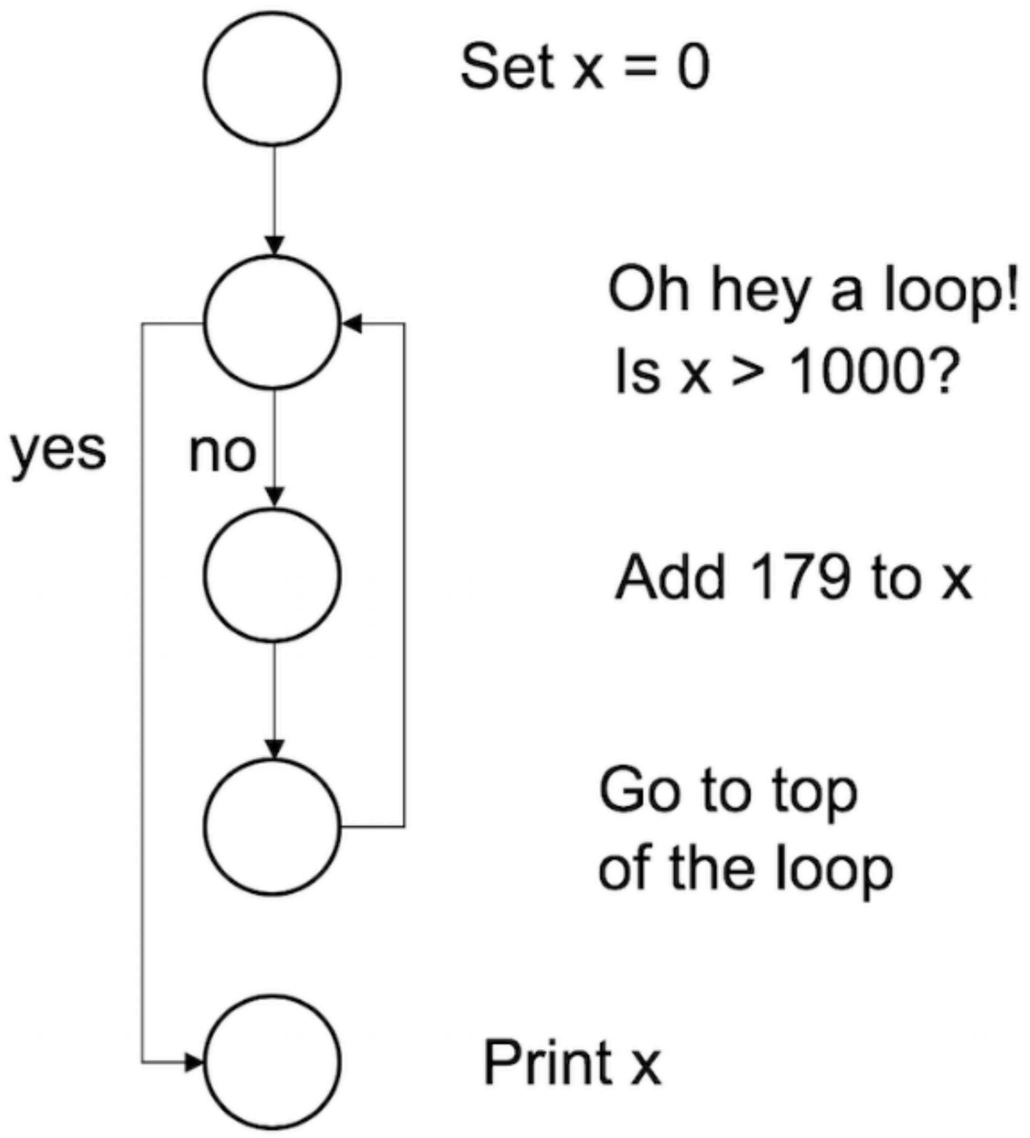
Needs to be a logical (TRUE or FALSE)

Does all of these things as long as the condition is TRUE

While loops

```
x <- 0
while (x < 1000) {
  x <- x + 179
}
print(x)

## [1] 1074
```



Exercise

Count down from 500 by 20s, stopping once the number is negative. Print out the entire sequence (i.e., it should print 500, 480… and stop at 0).

For loops

```
for ( VAR in VECTOR ) {
  STATEMENT1
  STATEMENT2
  ETC
}
```

Counts through each of the things in the vector

Does all of these things as long as the condition is TRUE

For loops

```
for ( value in 1:10 ) {
  answer <- 137*value
  print(answer)
}
```

```
# 137
# 274
# 411
# 548
# 685
# 822
# 959
# 1096
# 1233
# 1370
```

Looping over vectors

```
words <- c(“farewell”,”cruel”,”world”)
for (thisWord in words) {
   nLetters <- nchar(thisWord)
   blockWord <- toupper(thisWord)
   cat(blockWord,”has”,nLetters,”letters\n”)
}
```

```
# FAREWELL has 8 letters
# CRUEL has 5 letters
# WORLD has 5 letters
```

Exercises

1. Use a FOR loop to count down from 500 by 20s, stopping once the number is negative. Print out the entire sequence (i.e., it should print 500, 480… and stop at 0). At the end, also print out how many items total are in the sequence.

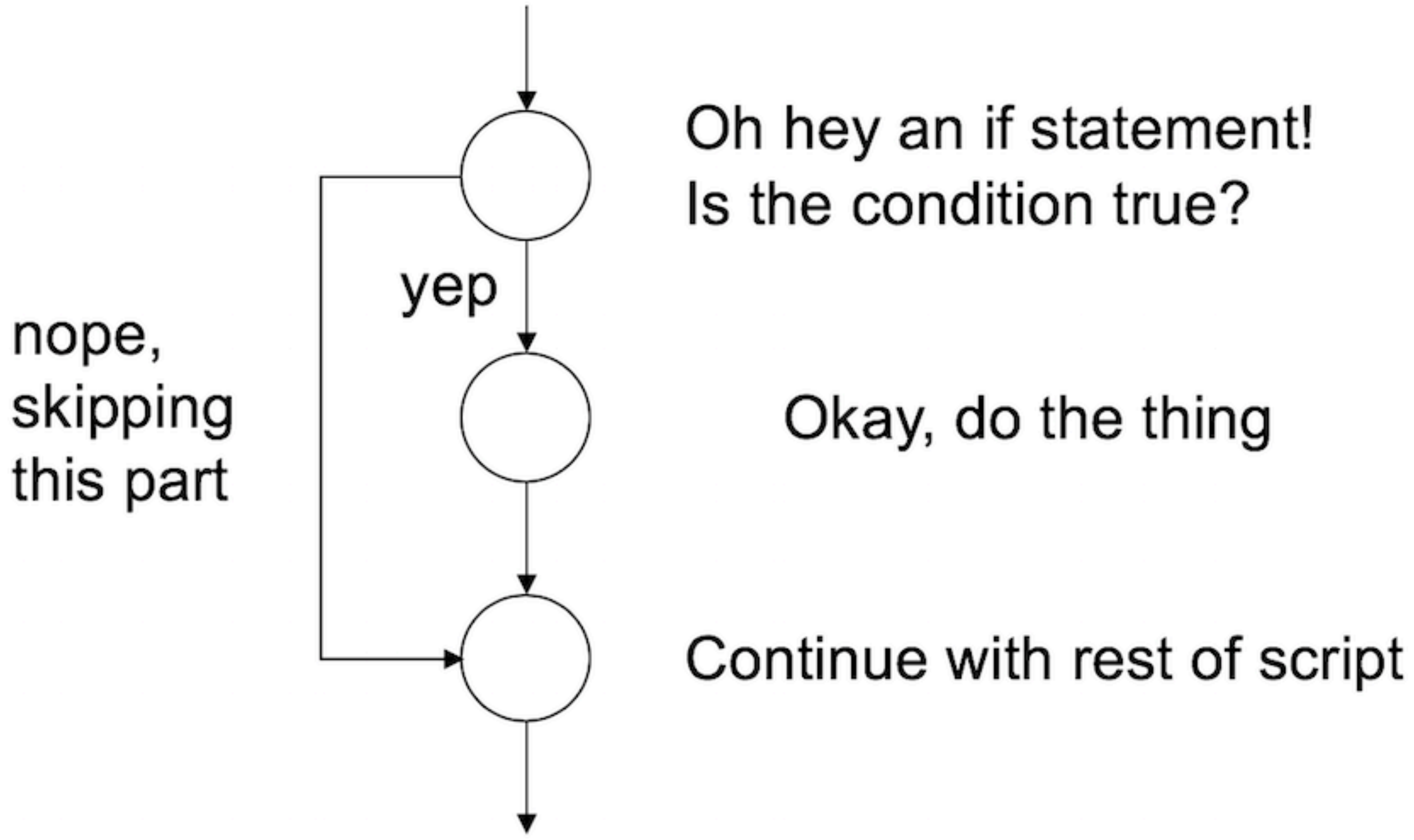
2. Create a data frame with two columns. One is the name of all of the people in your family (or if you prefer your close friends); there should be at least three. The second is their age. For each person it should print out their name next to their age: e.g. “Amy is 41.”

EXTRA CREDIT: At the end it should print out the name of the youngest and oldest person.

Branches

Branches

These let you evaluate conditional statements and do different things depending on the outcome



Branches

These let you evaluate conditional statements and do different things depending on the outcome

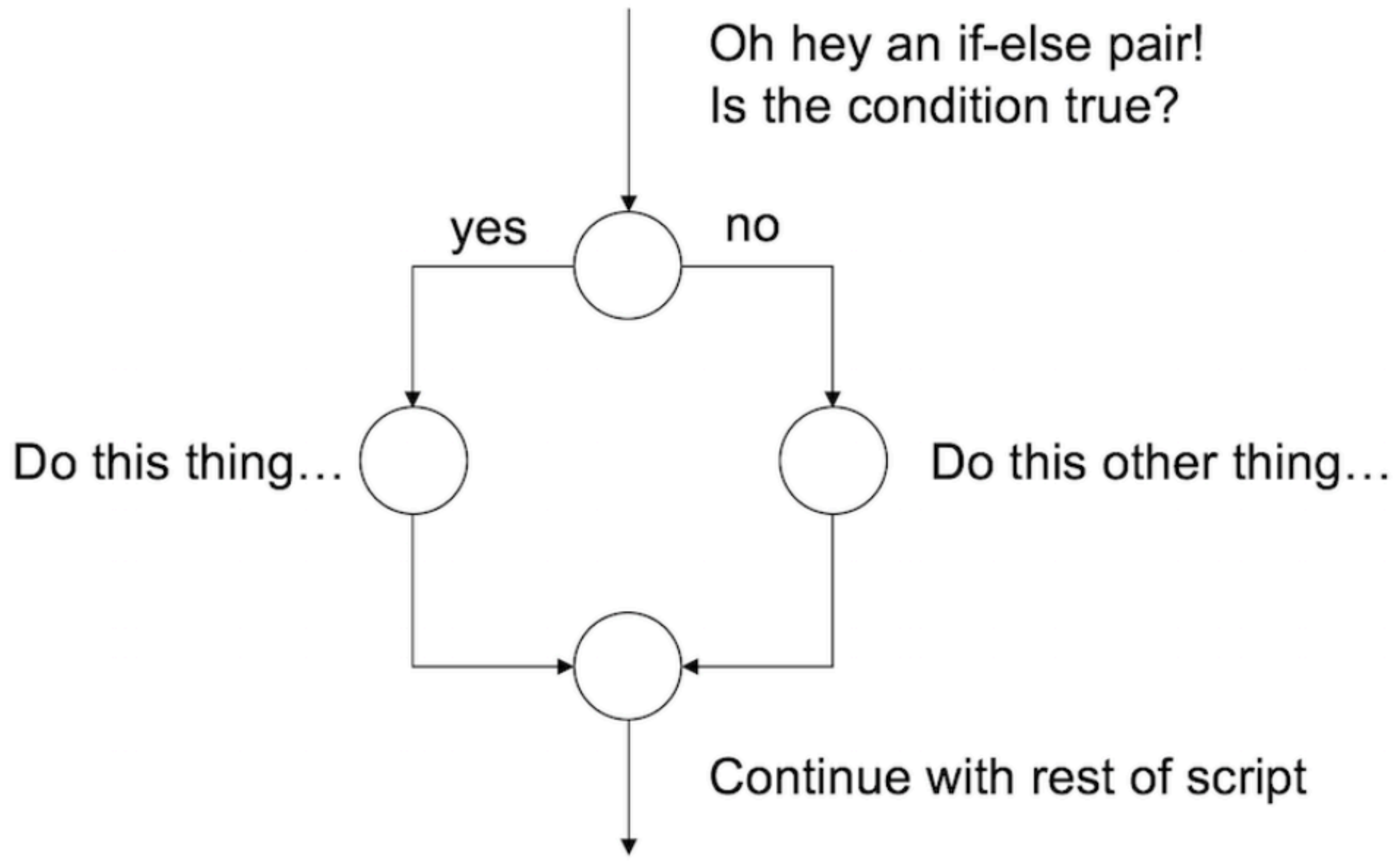
```
If ( CONDITION ) {
   STATEMENT1
   STATEMENT2
   ETC
}
```

Needs to be a logical (TRUE or FALSE)

Does all of these things only if the condition is TRUE

If-Else

```
if ( CONDITION ) {
   STATEMENT1
   STATEMENT2
   ETC
} else{
   STATEMENT3
   STATEMENT4
}
```



If-Else

```
if ( CONDITION ) {
  STATEMENT1
  STATEMENT2
  ETC
} else if ( CONDITION ){
  STATEMENT3
  STATEMENT4
} else {
  STATEMENT5
}
```

Example

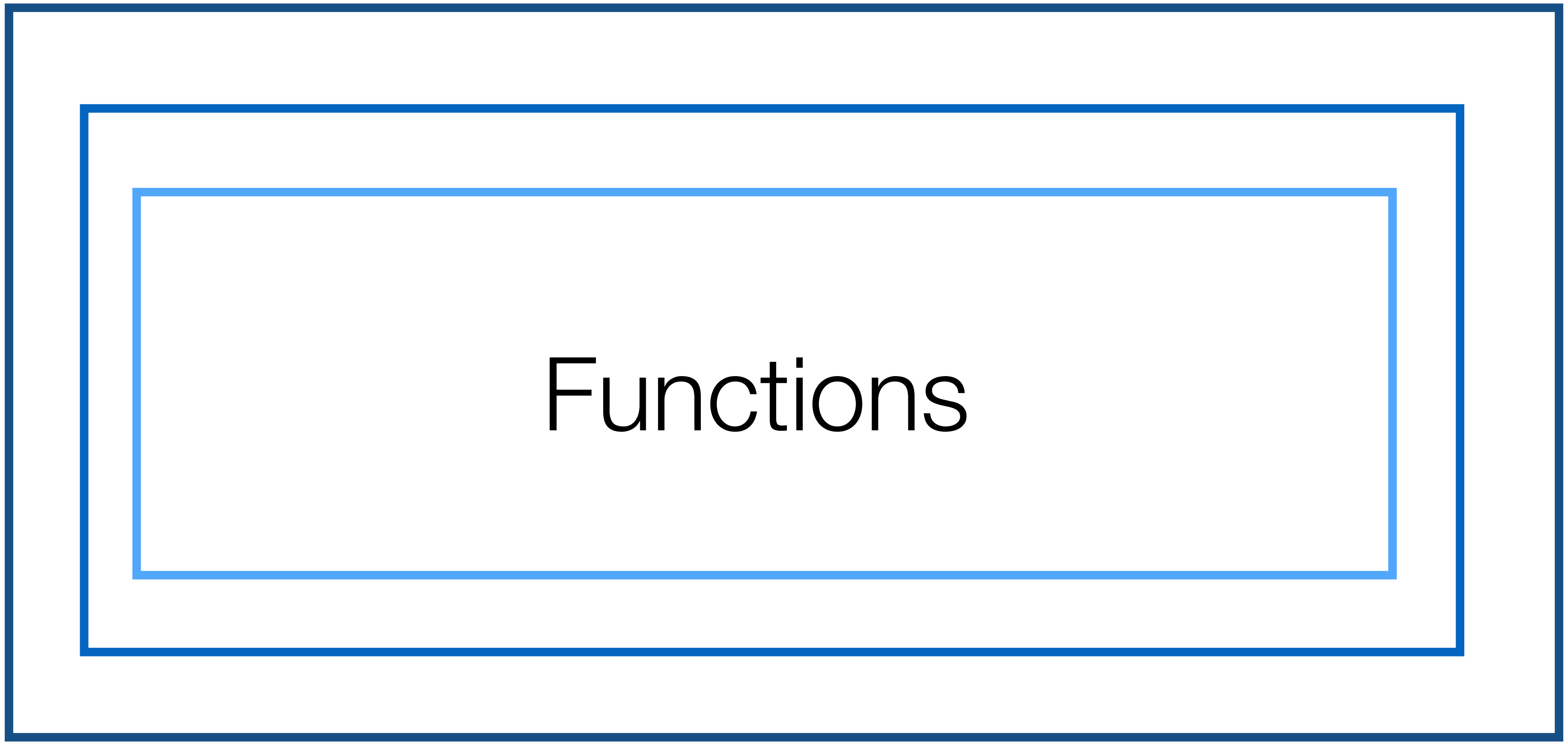
```
if (today==“Saturday”) {
   print(“Yay! Weekend!”)
} else if (today==“Sunday”) {
   print(“Uh oh, Monday is coming”)
} else {
   print(“I need coffee.”)
}
```

Exercises

1. Make a script that uses the `readline()` function to ask the user to enter their name. If it is your name, print out “You are awesome!” If it is the name of a hated enemy, print out “You are terrible!” Otherwise, print out “Hello, [name]!”

2. Write a script that uses `sample()` to randomly generate two integers between 1 and 10, and asks the user to add them together. If the user get it correct, it prints “Good job!” If not, it randomly generates two more integers and keeps going until the user gets it correct.

HINT: This is pretty difficult. Break it down. First make it so that you can generate the integers and get the answer for one problem. Then figure out how to check the answer. Then figure out how to keep going until the user is correct. Note that your final solution will involve *both* if statements and while loops.


Functions

Functions

You can actually create your *own* functions with arguments. Whenever it is called R will execute the statements within it. Creating a function means R creates a temporary environment with it while it's in practice, and only "keeps" the value in the `return()` statement.

```
FNAME <- function (ARG1, ARG2, ARG3, ETC) {
   STATEMENT1
   STATEMENT2
   STATEMENT3
   ETC
   return (VALUE)
}
```

Functions

Here's an example of a function that will square any number.

```
square <- function(x) {
   y <- x*x
   return(y)
}
```

```
> square(4)
# 16
```

Functions

The … argument lets the user enter as many arguments as they would like, as in the example below.

```
doubleMax <- function(...) {
   maxVal <- max(...)
   out <- 2*maxVal
   return(out)
}
```

Exercises

1. Convert your “adding” script from the previous exercise to a function called `askMath()` that returns the number of tries it took the user to get the answer correct.

2. Make a function called `giveFeedback()` that takes a number as an argument. If the number is 1, it should print “You only took one try! Great job!” If it is between 2 and 4 it should print “Pretty good!” If it is greater than 4 it should print “Nice effort, keep working!”

3. Make a function called `playGame()` that calls the previous two functions to play the adding game and give feedback at the end.

Intro to R cheat sheet

1 WHILE loops

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

2 FOR loops

```
for ( VAR in VECTOR ) {
   STATEMENT1
   STATEMENT2
   ETC
}
```

3 IF and ELSE IF

```
if ( CONDITION ) {
   STATEMENT1
   STATEMENT2
   ETC
} else if ( CONDITION ){
   STATEMENT3
   STATEMENT4
} else {
   STATEMENT5
}
```

4 Creating functions

```
FNAME <- function (ARG1, ARG2, ETC) {
    STATEMENT1
    STATEMENT2
    STATEMENT3
    ETC
    return (VALUE)
}
```