2. Conditional Expressions

- `EXPRESSION` runs only if `CONDITION` is `TRUE`. (Here `UPPER.CASE` is a placeholder for R code.)

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

```
x = -3 # example: set x to |x|
if (x < 0) {
  x = -x
}
x
```

- One of `TRUE.EXPRESSION` and `FALSE.EXPRESSION` runs:

```
if (CONDITION) {
  TRUE.EXPRESSION
} else {
  FALSE.EXPRESSION
}
```

"`} else`" must be on single line (for Console, source()).

```
x = 3 # example
if ((x %% 2) == 0) {
  parity = "even"
} else {
  parity = "odd"
}
parity
```

- The first true `CONDITION`'s `EXPRESSION` runs; or, if none is true, `DEFAULT_EXPRESSION` runs:

```
if (CONDITION_1) {
  EXPRESSION_1
} else if (CONDITION_2) { # optional "else if" clauses
  EXPRESSION_2            # ...
} ... {
  ...
} else {                  # optional "else" clause
  DEFAULT_EXPRESSION
}
```

```
water.state = function(temperature) { # example within a function
  if (temperature < 32) {
    state = "frozen"
  } else if (temperature > 212) {
    state = "boiling"
  } else {
    state = "liquid"
  }
  return(state)
}
water.state(20); water.state(60); water.state(220)
```

- None of the three constructs above works if its `CONDITION` has length greater than one. However, `x = ifelse(test, yes, no)` sets `x` to be a vector with the same length as `test` filled with elements from `yes` or `no` depending on the logical values in `test`. e.g.

```
x = 1:3; parity = ifelse((x %% 2) == 0, "even", "odd"); parity
```

More examples

```
is.heads = function() { # Return TRUE or FALSE to simulate a coin flip.
  r = rnorm(1)
  if (r < 0) {
    return(TRUE)
  } else {
    return(FALSE)
  }
}
x=is.heads() # test cases
N=100; coins = replicate(N, is.heads()); mean(coins)

is.heads = function() { # shorter version
  r = rnorm(1)
  return(r < 0)
}

is.heads = function() { # even shorter
  return(rnorm(1) < 0)
}


# Return maximum of a and b. # (Implements part of R's max() function;
baby.max = function(a, b) {  #  BUG: fails for vector input.)
  if (a > b) {
    return(a)
  } else {
    return(b)
  }
}
stopifnot(isTRUE(all.equal(4, baby.max(3,  4))))
stopifnot(isTRUE(all.equal(3, baby.max(3, -4))))
x = c(3, 3); y = c(4, -4); m = baby.max(x, y)
stopifnot(isTRUE(all.equal(c(4, 3), m))) # BUG revealed.

baby.max = function(a, b) {  # Improved to work for vector input.
  return(ifelse(test=(a > b), yes=a, no=b))
}
stopifnot(isTRUE(all.equal(4, baby.max(3,  4))))
stopifnot(isTRUE(all.equal(3, baby.max(3, -4))))
x = c(3, 3); y = c(4, -4); m = baby.max(x, y)
stopifnot(isTRUE(all.equal(c(4, 3), m)))
```