```
> library(swirl)

| Hi! I see that you have some variables saved in your workspace. To keep things
| running smoothly, I recommend you clean up before starting swirl.

| Type ls() to see a list of the variables in your workspace. Then, type
| rm(list=ls()) to clear your workspace.

| Type swirl() when you are ready to begin.

> swirl()

| Welcome to swirl! Please sign in. If you've been here before, use the same name
| as you did then. If you are new, call yourself something unique.

What shall I call you? mesbah

| Please choose a course, or type 0 to exit swirl.

1: R Programming
2: Take me to the swirl course repository!

Selection: 1

| Please choose a lesson, or type 0 to return to course menu.

 1: Basic Building Blocks      2: Workspace and Files
 3: Sequences of Numbers       4: Vectors
 5: Missing Values             6: Subsetting Vectors
 7: Matrices and Data Frames   8: Logic
 9: Functions                 10: lapply and sapply
11: vapply and tapply         12: Looking at Data
13: Simulation                14: Dates and Times
15: Base Graphics

Selection: 14

  |                                                                      |   0%

| R has a special way of representing dates and times, which can be helpful if
| you're working with data that show how something changes over time (i.e.
| time-series data) or if your data contain some other temporal information, like
| dates of birth.

...

  |==                                                                    |   3%
| Dates are represented by the 'Date' class and times are represented by the
| 'POSIXct' and 'POSIXlt' classes. Internally, dates are stored as the number of
| days since 1970-01-01 and times are stored as either the number of seconds since
| 1970-01-01 (for 'POSIXct') or a list of seconds, minutes, hours, etc. (for
| 'POSIXlt').

...

  |====                                                  |   6%
| Let's start by using d1 <- Sys.Date() to get the current date and store it in
| the variable d1. (That's the letter 'd' and the number 1.)

> d1 <- Sys.Date()

| You are really on a roll!

  |======                                                |   8%
| Use the class() function to confirm d1 is a Date object.

> class(d1)
[1] "Date"

| You got it!

  |========                                              |  11%
| We can use the unclass() function to see what d1 looks like internally. Try it
| out.

> unclass(d1)
[1] 18534

| Your dedication is inspiring!

  |==========                                            |  14%
| That's the exact number of days since 1970-01-01!

...

  |============                                          |  17%
| However, if you print d1 to the console, you'll get today's date --
| YEAR-MONTH-DAY. Give it a try.

> d1
[1] "2020-09-29"

| That's the answer I was looking for.

  |==============                                        |  19%
| What if we need to reference a date prior to 1970-01-01? Create a variable d2
| containing as.Date("1969-01-01").
```

```
> d2 <- as.Date("1969-01-01")

| You are really on a roll!

  |================                                             |  22%
| Now use unclass() again to see what d2 looks like internally.

> unclass(d2)
[1] -365

| Keep working like that and you'll get there!

  |==================                                           |  25%
| As you may have anticipated, you get a negative number. In this case, it's -365,
| since 1969-01-01 is exactly one calendar year (i.e. 365 days) BEFORE 1970-01-01.

...

  |====================                                         |  28%
| Now, let's take a look at how R stores times. You can access the current date
| and time using the Sys.time() function with no arguments. Do this and store the
| result in a variable called t1.

> t1 < Sys.time()
Error: object 't1' not found
> t1 <- Sys.time()

| Perseverance, that's the answer.

  |======================                                       |  31%
| View the contents of t1.

> t1
[1] "2020-09-29 11:57:11 +06"

| All that hard work is paying off!

  |========================                                     |  33%
| And check the class() of t1.

> class(t1)
[1] "POSIXct" "POSIXt"

| That's correct!

  |==========================                                   |  36%
| As mentioned earlier, POSIXct is just one of two ways that R represents time
| information. (You can ignore the second value above, POSIXt, which just
| functions as a common language between POSIXct and POSIXlt.) Use unclass() to
| see what t1 looks like internally -- the (large) number of seconds since the
| beginning of 1970.

> unclass(t1)
[1] 1601359031

| Excellent work!

  |============================                                        |  39%
| By default, Sys.time() returns an object of class POSIXct, but we can coerce the
| result to POSIXlt with as.POSIXlt(Sys.time()). Give it a try and store the
| result in t2.

> t2 <- as.POSIXlt(Sys.time())

| You nailed it! Good job!

  |==============================                                      |  42%
| Check the class of t2.

> class(t2)
[1] "POSIXlt" "POSIXt"

| Keep working like that and you'll get there!

  |================================                                    |  44%
| Now view its contents.

> t2
[1] "2020-09-29 11:59:29 +06"

| Your dedication is inspiring!

  |==================================                                  |  47%
| The printed format of t2 is identical to that of t1. Now unclass() t2 to see how
| it is different internally.

> unclass(t2)
$sec
[1] 29.63121

$min
[1] 59

$hour
[1] 11

$mday
```

```
[1] 29

$mon
[1] 8

$year
[1] 120

$wday
[1] 2

$yday
[1] 272

$isdst
[1] 0

$zone
[1] "+06"

$gmtoff
[1] 21600

attr(,"tzone")
[1] ""    "+06" "+06"

| Perseverance, that's the answer.

  |===================================                               |  50%
| t2, like all POSIXlt objects, is just a list of values that make up the date and
| time. Use str(unclass(t2)) to have a more compact view.

> str(unclass(t2))
List of 11
 $ sec   : num 29.6
 $ min   : int 59
 $ hour  : int 11
 $ mday  : int 29
 $ mon   : int 8
 $ year  : int 120
 $ wday  : int 2
 $ yday  : int 272
 $ isdst : int 0
 $ zone  : chr "+06"
 $ gmtoff: int 21600
 - attr(*, "tzone")= chr [1:3] "" "+06" "+06"

| That's correct!
```

```
  |========================================                       |  53%
| If, for example, we want just the minutes from the time stored in t2, we can
| access them with t2$min. Give it a try.

> t2$min
[1] 59

| That's the answer I was looking for.

  |==========================================                     |  56%
| Now that we have explored all three types of date and time objects, let's look
| at a few functions that extract useful information from any of these objects --
| weekdays(), months(), and quarters().

...

  |============================================                   |  58%
| The weekdays() function will return the day of week from any date or time
| object. Try it out on d1, which is the Date object that contains today's date.

> weekdays(t1)
[1] "Tuesday"

| Not exactly. Give it another go. Or, type info() for more options.

| Try weekdays(d1) to get the day of the week.

> weekdays(d1)
[1] "Tuesday"

| You nailed it! Good job!

  |==============================================                 |  61%
| The months() function also works on any date or time object. Try it on t1, which
| is the POSIXct object that contains the current time (well, it was the current
| time when you created it).

> months(t1)
[1] "September"

| You are quite good my friend!

  |================================================               |  64%
| The quarters() function returns the quarter of the year (Q1-Q4) from any date or
| time object. Try it on t2, which is the POSIXlt object that contains the time at
| which you created it.

> quarters(t2)
[1] "Q3"
```

```
| All that hard work is paying off!

  |===============================================                        |  67%
| Often, the dates and times in a dataset will be in a format that R does not
| recognize. The strptime() function can be helpful in this situation.

...

  |=================================================                      |  69%
| strptime() converts character vectors to POSIXlt. In that sense, it is similar
| to as.POSIXlt(), except that the input doesn't have to be in a particular format
| (YYYY-MM-DD).

...

  |===================================================                    |  72%
| To see how it works, store the following character string in a variable called
| t3: "October 17, 1986 08:24" (with the quotes).

> t3 <- "October 17, 1986 08:24"

| You got it right!

  |=====================================================                  |  75%
| Now, use strptime(t3, "%B %d, %Y %H:%M") to help R convert our date/time object
| to a format that it understands. Assign the result to a new variable called t4.
| (You should pull up the documentation for strptime() if you'd like to know more
| about how it works.)

> strptime(t3, "%B %d, %Y %H:%M")
[1] "1986-10-17 08:24:00 +06"

| You almost had it, but not quite. Try again. Or, type info() for more options.

| t4 <- strptime(t3, "%B %d, %Y %H:%M") will convert our date/time object to a
| format that R understands.

> t4 <- strptime(t3, "%B %d, %Y %H:%M")

| You are amazing!

  |=======================================================                |  78%
| Print the contents of t4.

> t4
[1] "1986-10-17 08:24:00 +06"

| You got it right!
```

```
  |============================================================         |  81%
| That's the format we've come to expect. Now, let's check its class().

> class(t4)
[1] "POSIXlt" "POSIXt"

| You nailed it! Good job!

  |==============================================================       |  83%
| Finally, there are a number of operations that you can perform on dates and
| times, including arithmetic operations (+ and -) and comparisons (<, ==, etc.)

...

  |================================================================     |  86%
| The variable t1 contains the time at which you created it (recall you used
| Sys.time()). Confirm that some time has passed since you created t1 by using the
| 'greater than' operator to compare it to the current time: Sys.time() > t1

> Sys.time() > t1
[1] TRUE

| Keep working like that and you'll get there!

  |==================================================================   |
89%
| So we know that some time has passed, but how much? Try subtracting t1 from the
| current time using Sys.time() - t1. Don't forget the parentheses at the end of
| Sys.time(), since it is a function.

> Sys.time() - t1
Time difference of 14.71975 mins

| That's a job well done!

  |====================================================================  |
92%
| The same line of thinking applies to addition and the other comparison
| operators. If you want more control over the units when finding the above
| difference in times, you can use difftime(), which allows you to specify a
| 'units' parameter.

...

  |====================================================================== |
94%
| Use difftime(Sys.time(), t1, units = 'days') to find the amount of time in DAYS
| that has passed since you created t1.
```

```
> difftime(Sys.time(), t1, units = 'days')
Time difference of 0.01156042 days

| That's the answer I was looking for.

  |=============================================================================  |
97%
| In this lesson, you learned how to work with dates and times in R. While it is
| important to understand the basics, if you find yourself working with dates and
| times often, you may want to check out the lubridate package by Hadley Wickham.

...

  |
================================================================================|
100%
| Would you like to receive credit for completing this course on Coursera.org?

1: Yes
2: No

Selection: 1
What is your email address? ahmed.mesbahuddin.md@outlook.com
What is your assignment token? beRKnWdVb7OtYORn
Grade submission succeeded!

| You're the best!

| You've reached the end of this lesson! Returning to the main menu...

| Please choose a course, or type 0 to exit swirl.

1: R Programming
2: Take me to the swirl course repository!

Selection: save.image("~/Downloads/test.RData")
Enter an item from the menu, or 0 to exit
Selection: save.image("~/Downloads/test.R.RData")
Enter an item from the menu, or 0 to exit
Selection:
```