Creating a vector:

Lesson:
Vectors are one dimension arrays that can contain numbers, character strings, or logical data. A vector could contain the same or different data types depending on usage.
In R, you can create a vector by using the combine function(c()). Essentially you enter the collection of data within the parenthesis. Here's an example:

```
fruits <- c("apple", "orange", "kiwi", "loquat")
prime_numbers <- c(2, 5, 139, 193, 71)
boolean <- c(FALSE, TRUE, TRUE, FALSE)
```

take note that each data is separated by a comma.

To do:

1. Create and display the variable "vector" such that it contains the following elements: 1, "hello" and FALSE (in that order)
2. Create two variables named "odd" and "even" respectively and fill them with odd and even numbers from 1 to 10.

Naming a vector:

Lesson:
Vectors can be used to collect data and transform them into more meaningful information especially when your collection grows larger. One way to achieve this is to give a name to elements of the vector with the uses of the **names( )** function.

Refer to the example below:

```
meal_plan <- c ("Baked Salmon", "Spaghetti", "Roast Chicken", "Braised Beef", "Shepherd's Pie")
```

This is a vector of a planned meal from Monday to Friday. We can use this to assign the names of the day to the data. Think of it as adding headers. Similar to a calendar. To do this we will add the following:

```
names(meal_plan) <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday")
```

When you display the values of meal plan it will be presented as:

Monday	Tuesday	Wednesday	Thursday	Friday
"Baked Salmon"	"Spaghetti"	"Roast Chicken"	"Braised Beef"	"Shepherd's Pie"

To do:

1. Create a vector containing the names of people you know and their profession. And then add the names "Name" and "Profession" respectively.
2. Display your created vector

Two become one:

Lesson:
You are able to perform arithmetic operations with and within vectors that contain numbers. It is important to know that if you sum two vectors in R, it takes the element-wise sum. For example, the following three statements are completely equivalent:

```
c(1, 2, 3) + c(4, 5, 6)
c(1 + 4, 2 + 5, 3 + 6)
c(5, 7, 9)
```

You are also able to assign vectors to variables to achieve the same result. For example

```
a <- c(2, 4, 6)
b <- c(8, 10, 12)
c <- a + b
```

To do:

1. Add the following two vectors and assign them to a variable name "total":
   Vector1 <- c(10 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120)
   Vector2 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12)
2. Display the value of total.

Computing Vectors:

Lesson:
There are a number of kinds of functions depending on the need or purpose. One kind of function is called **Arithmetic Functions** these functions may be used to carry out arithmetic operations within a vector.
For example, In order to get the total value of all numerical data within a vector, one may use the sum function. Used as **sum()**, you insert the variable within the parenthesis to add all the numbers in the vector.

To do:

1. Get the sum of the following vector using the sum function:
   Vector1 <- c(2, 5, 9, 3)

Comparing Vectors:
Lesson:
Logical Operators are used to two values in R. When used, you will be provided with an answer that will consist of TRUE or FALSE. Consider the following basic logical operators:

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to

To Do:

1. Copy the following code to your editor then Highlight the first two lines and click run.
   Vector1 <- c(48, 66, 99)
   Vector2 <- c(79, 90, 81)

2. Then copy the following to the editor and analyze each line. Try to predict the output. **Highlight the line and click Run to test them individually.**

```
sum(Vector1) <  sum(Vector2)
sum(Vector2) <= sum(Vector1)
sum(Vector2) > sum(Vector1)
sum(Vector1) >= sum(Vector2)
sum(Vector1) == sum(Vector2)
sum(Vector1) != sum(Vector2)
```

Vector Indexing:

Lesson:
As Vector contain multiple data there are instances where you only need to select one out of the entire collection. This is called indexing. Index is a numeral representation of the position or location of an element inside a vector.

For example:
Dwarves <- c("Doc", "Grumpy", "Happy", "Sleepy", "Dopey", "Bashful", "Sneezy")
[1] [2] [3] [4] [5] [6] [7]

Given the vector Dwarves, should you wish to select “Happy”, you just enter **Dwarves[3].** Additionally, if you want to select all of the data except one, you may use a **negative index**. For example, entering **Dwarves[-7]** will display all of the dwarves **except “Sneezy”**.

If you want to select more than one value, you may use **c()** and indicating the indexes you want serparated by a comma. For example, **Dwarves[c(1, 6)]** will provide you with “Doc” “ Bashful” respectively. The same thing can be said with negative indexes.

It is also possible to select ranges in a vector. Instead of using Dwarves[c(2, 3, 4)] you can substitute it as **Dwarves[2:4]** to achieve the same result.

To do:

1. Copy the following commented lines and provide the code to display what is asked. Use the same vector as above.
   #Select Grumpy
   #Select all except Dopey
   #Select Sleepy
   #Select all except Happy
   #Select Grumpy, Happy, Sleepy and Dopey
   #Exclude Sneezy, Dopey and Happy
   #Select Grumpy, Bashful and Doc
   #Select Dopey, Bashful Sneezy
   #Select Doc, Grumpy, Happy, Bashful and Sneezy