Although there may be several cases where you need to create a new environment using new.env(), you will more often create new environments whenever you execute functions. An execution environment is an environment that exists temporarily within the scope of a function that is being executed. For example if we have the following code:

```
x <- 10

my_func <- function(){
  x <- 5
  return(x)
}

my_func()
```

What do you think will be the result of my_func()? Make your guess and then take a look at the executed code below:

```
x <- 10

my_func <- function(){
  x <- 5
  return(x)
}

my_func()
[1] 5
```

So what exactly is happening above? First the name x is being assigned the value 10 in the global environment. Then the namemy_func is being assigned the value of the function function(){x <- 5};return(x)} in the global environment. When my_func() is executed, a new environment is created called the execution environment which only exists while my_func() is running. Inside of the execution environment the name x is assigned the value 5. Whenreturn() is executed it looks first in the execution environment for a value that is assigned to x. Then the value 5 is returned. In contrast to the situation above, take a look at this variation:

```
x <- 10

another_func <- function(){
  return(x)
}

another_func()
[1] 10
```

In this situation the execution environment inside of another_func()does not contain an assignment for the name x, so R looks for an assignment in the parent environment of the execution environment which is the global environment. Since x is assigned the value 10 in the global environment 10 is returned.

After seeing the cases above you may be curious if it's possible for an execution environment to manipulate the global environment. You're already familiar with the assignment operator <-, however you should also be aware that there's another assignment operator called the *complex assignment operator* which looks like <<-. You can use the complex assignment operator to re-assign or even create name-value bindings in the global environment from within an execution environment. In this first example, the function assign1() will change the value associated with the name x:

```
x <- 10
x
[1] 10

assign1 <- function(){
  x <<- "Wow!"
}

```