

# Introduction to Exception Handling

In this lesson, we will give a brief introduction to exception handling in R.

## We'll cover the following ^

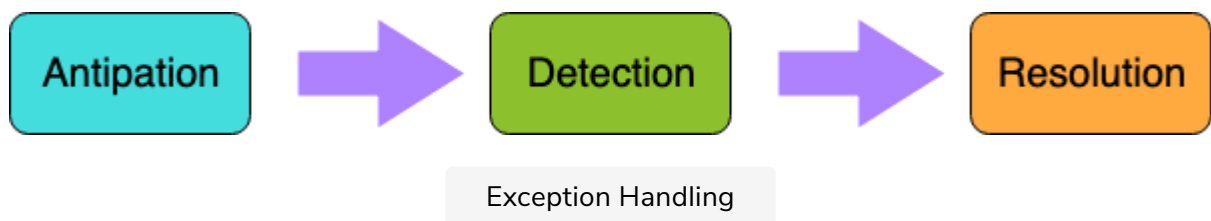
- What is Exception Handling?
  - Errors
  - Warnings

While executing a program many unexpected things can happen. The code might give **errors** or **warnings**. In most cases, we do not want this to hinder the flow of our program. So, what should you do when something goes wrong with your **R code**? In this chapter, we will be learning all the tools that will help us address this problem.

This chapter will teach us how to gracefully handle unanticipated problems. We will learn all the functions that can communicate problems and let us handle them. The term used for such a task is “**Exception Handling**”.

## What is Exception Handling? #

Exception handling involves first anticipating the error, detecting it, and then resolving it in the program.



In the illustration above, anticipation means that as a programmer we know that something might go wrong. We detect the location of that error. We realize where and how such an error can occur and then we try to allow the program to continue running despite the erroneous situation. We find the best method to solve such an issue and not let the compiler throw unexpected errors.

**Error handlers** are specialized programs that forestall errors. They recover from errors when they occur without terminating the program abruptly, or, if all else fails, gracefully ending an effected application.

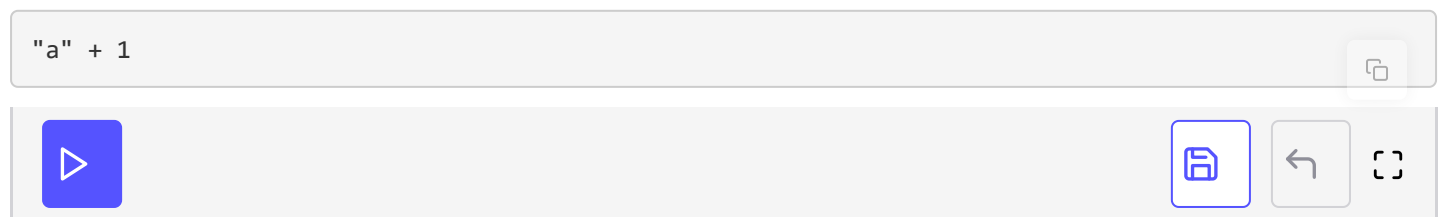
The execution of an R script can be interrupted by the following R signals:

- errors
- warnings
- info messages
- user requested interrupts (by hitting CTRL + C/BRK or ESC)

## Errors #

Error messages occur whenever a programmer tries to do something that the compiler does not allow. You might remember that **arithmetic operations** between different types of objects are not allowed. Therefore, the compiler will throw an error here.

```
"a" + 1
```



Example of an error

Here, the compiler will throw an error.

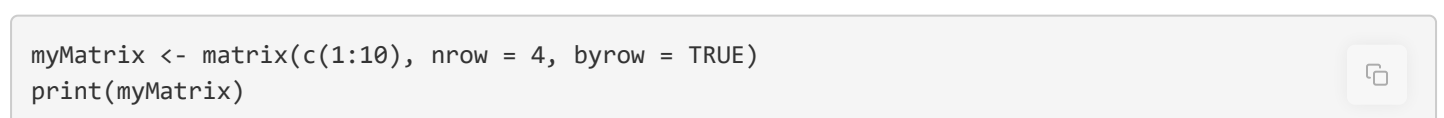
```
Error in "a" + 1 : non-numeric argument to binary operator
```

## Warnings #

Some warning message cautions users without terminating the execution of the program. It means “*although I can and will give you an answer, there might be a problem with your inputs. Thus, the computation could be flawed.*”

For example, while creating a matrix that has a vector of numbers whose length is not a **sub-multiple** or **multiple** of the number of rows.

```
myMatrix <- matrix(c(1:10), nrow = 4, byrow = TRUE)  
print(myMatrix)
```



In the code snippet above, the **length** of the **vector** being passed is 10 but the number of rows is 4. Ten numbers cannot be easily mapped to a **matrix** with 4 rows, however, this is possible. So, the compiler does this for you but displays a **warning** ⚠:

```
Warning message:
```

```
In matrix(c(1:10), nrow = 4, byrow = TRUE) :
```

```
data length [10] is not a sub-multiple or multiple of the number of rows [4]
```

Now that we have an idea of what exception handling is, let's move on to adding this in our code.