Practical 4

Jumping Rivers

In this practical we're going to have a go at building a function to automatically create the directory structure we've just talked about in chapter 4.

Question 1

a) We can use the `dir.create()` function to create a directory in R. Here's a starter function that will create a project directory depending on the users input

```
create_workflow = function(project_name) {
    dir.create(project_name)
}
```

b) Now we need to create the directories `input`, `R`, `graphics` and `output` within the main project directory. To do this we'll need to create the filepath for each. `file.path()` is a handy function that will help.

```
file.path("project", "directory")
```

c) Now we need to create the R scripts `load.R`, `clean.R`, `func.R`, `do.R` and `graphics.R` within the `R` directory. The `file.create()` function can create files of any extension. So to create an R file we could do

```
file.create("load.R")
```

and this will create an empty R script called `load.R`. Hint: You can do this with a for loop. Remember your file paths!

Question 2 - Harder

This question is much harder than the first, you have been warned! It would be ideal if we could insert the source commands into the R scripts as well. You can append lines of text to a file using the `writeLines()`, `file()` and `close()` functions. For instance,
The contents of each file should look like so:

- `load.R` - empty
- `clean.R` - One line of code: `source("project_name/R/load.R")`
- `func.R` - One line of code: `source("project_name/R/clean.R")`
- `do.R` - One line of code: `source("project_name/R/func.R")`

- `graphics.R` - One line of code: `source("project_name/R/do.R")`

The idea being that when you call `source("do.R")` in `graphics.R`, it will run all 4 previous files.