Shiny I - Excercises

Excercise 1:

Create an app that greets the user by name. You don't know all the functions you need to do this yet, so I've included some lines of code below. Think about which lines you'll use and then copy and paste them into the right place in a Shiny app.

```
tableOutput("mortgage")
output$greeting <- renderText({
  paste0("Hello ", input$name)
})
numericInput("age", "How old are you?", value = NA)
textInput("name", "What's your name?")
textOutput("greeting")
output$histogram <- renderPlot({
  hist(rnorm(1000))
}, res = 96)
```

Excercise 2:

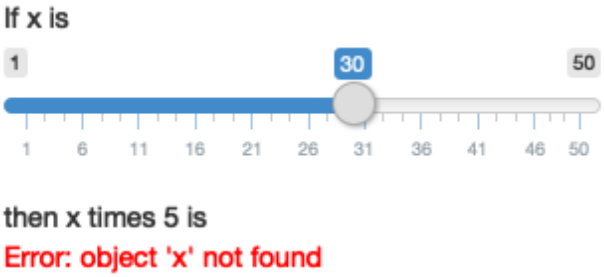
Suppose your friend wants to design an app that allows the user to set a number (x) between 1 and 50, and displays the result of multiplying this number by 5. This is their first attempt:

```
library(shiny)

ui <- fluidPage(
  sliderInput("x", label = "If x is", min = 1, max = 50, value = 30),
  "then x times 5 is",
  textOutput("product")
)

server <- function(input, output, session) {
  output$product <- renderText({
    x * 5
  })
}
shinyApp(ui, server)
```

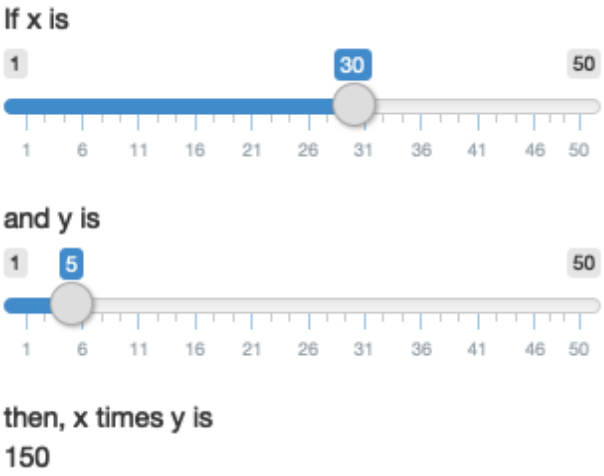
But unfortunately it has an error:



Can you help them find and correct the error?

Excercise 3:

Extend the app from the previous exercise to allow the user to set the value of the multiplier, y , so that the app yields the value of x * y . The final result should look like this:



Excercise 4:

Take the following app which adds some additional functionality to the last app described in the last exercise. What's new? How could you reduce the amount of duplicated code in the app by using a reactive expression.

```
library(shiny)

ui <- fluidPage(
  sliderInput("x", "If x is", min = 1, max = 50, value = 30),
  sliderInput("y", "and y is", min = 1, max = 50, value = 5),
  "then, (x * y) is", textOutput("product"),
  "and, (x * y) + 5 is", textOutput("product_plus5"),
  "and (x * y) + 10 is", textOutput("product_plus10")
)

server <- function(input, output, session) {
  output$product <- renderText({
```

```
    product <- input$x * input$y
    product
  })
  output$product_plus5 <- renderText({
    product <- input$x * input$y
    product + 5
  })
  output$product_plus10 <- renderText({
    product <- input$x * input$y
    product + 10
  })
}

shinyApp(ui, server)
```

Excercise 5:

Consider the following app. You select a dataset from a package (this time we're using the ggplot2 package) and the app prints out a summary and plot of the data. It also follows good practice and makes use of reactive expressions to avoid redundancy of code. However there are three bugs in the code provided below. Can you find and fix them?

```
library(shiny)
library(ggplot2)

datasets <- c("economics", "faithfuld", "seals")

ui <- fluidPage(
  selectInput("dataset", "Dataset", choices = datasets),
  verbatimTextOutput("summary"),
  tableOutput("plot")
)

server <- function(input, output, session) {
  dataset <- reactive({
    get(input$dataset, "package:ggplot2")
  })
  output$summmry <- renderPrint({
    summary(dataset())
  })
  output$plot <- renderPlot({
    plot(dataset)
  }, res = 96)
}
```

Excercise 6:

When space is at a premium, it's useful to label text boxes using a placeholder that appears inside the text entry area. How do you call textInput() to generate the UI below?

Excercise 7:

Carefully read the documentation for sliderInput() to figure out how to create a date slider, as shown below.

Excercise 8:

Create a slider input to select values between 0 and 100 where the interval between each selectable value on the slider is 5. Then, add animation to the input widget so when the user presses play the input widget scrolls through the range automatically.

Excercise 9:

If you have a moderately long list in a selectInput() , it's useful to create sub-headings that break the list up into pieces. Read the documentation to figure out how. (Hint: the underlying HTML is called

Excercise 10:

Which of textOutput() and verbatimTextOutput() should each of the following render functions be paired with?

1. `renderPrint(summary(mtcars))`
2. `renderText("Good morning!")`
3. `renderPrint(t.test(1:5, 2:6))`
4. `renderText(str(lm(mpg ~ wt, data = mtcars)))`

Excercise 11:

Re-create the Shiny app below, setting height and width of the plot to 300px and 700px respectively. Set the plot “alt” text so that a visually impaired user can tell that its a scatterplot of five random numbers.

Hint: plot(1:5)