```
#The httr library
#httr is a R library that allows you to build and send HTTP requests, as well as process
HTTP requests easily. We can import the package as follows (may take less than minute to
import):
library(httr)

#You can make a GET request via the method get to www.ibm.com:
url<-'https://www.ibm.com/'
response<-GET(url)
response

#We have the response object response , this has information about the response, like the
status of the request. We can view the status code using the attribute status
response$status

#You can also check the headers of the response
response_headers <- headers(response)
response_headers

#We can obtain the date the request was sent using the key Date
response_headers['date']

#Content-Type indicates the type of data:
response_headers['content-type']

#To obtain the original request, you can view it via response object:
response$request$headers

#coding Exercise: in the code cell below, find the content-length attribute in the
response header
# Write your code below. Don't forget to press Shift+Enter to execute the cell
response_headers['content-length']

#Now, let's get the content of HTTP response
content(response)

#which is the IBM home page (in fact, HTML page which you will learn later in this course)
#You can load other types of data for non-text requests like images, consider the URL of
the following image:
image_url<-'https://gitlab.com/ibm/skills-
network/courses/placeholder101/-/raw/master/labs/module%201/images/IDSNlogo.png'

#We can make a get request:
image_response<-GET(image_url)

#We can look at the response header:
image_headers <- headers(image_response)

#We can we can see the 'Content-Type', which is an image
image_headers['content-type']

#An image is a response object that contains the image as a bytes-like object. As a
result, we must save it using a file object. First, we specify the file path and name
image <- content(image_response, "raw")
writeBin(image, "logo.png")

#Suppose we have a simple GET API with base URL for http://httpbin.org/
url_get <- 'http://httpbin.org/get'
#and we want to add some URL parameters to above GET API. To do so, we simply create a
named list with parameter names and values:
query_params <- list(name = "Yan", ID = "123")
#Then passing the list query_params to the query argument of the  GET() function.
#It basically tells the GET API I only want to get resources with name equals Yan and id
```

```
equals 123.
#OK, let's make the GET request to 'http://httpbin.org/get' with the two arameters
response <- GET(url_get, query=query_params)
#We can print out the updated URL and see the attached URL parameters.
response$request$url
#After the base URL http://httpbin.org/get, you can see the URL parameters name=Yan&ID=123
are seperated by ?
#The attribute args of the response had the name and values:
content(response)$args

#Post Requests
#Like a GET request a POST is used to send data to a server in a request body. In order to
send the Post Request in Python in the URL we change the route to POST:
url_post <- 'http://httpbin.org/post'
#This endpoint will expect data as a file or as a form, a from is convenient way to
configure an HTTP request to send data to a server.
#To make a POST request we use the POST() function, the list body is passed to the
parameter  body :
body<- list(course_name='Introduction to R', instructor='Yan')
response<-POST('http://httpbin.org/post', body = body)
response
#We can see POST request has a body stored in fields attribute
response$request$fields
```