Problem Set # 1

Programming and Data Management

Overview

This challenge will test your knowlege of variable assignment, functions, vectors, lists and more that we have covered in part I of the course. This challenge will not just ask you to remember syntax you have learned but will in may cases require you to use what you have learned to solve problems. You will have to think more deeply about the concepts you learned and how to apply them. Please post in teams or email me if you do not understand something, however, YOU ARE TO DO THIS ASSIGNMENT ON YOUR OWN, and cannot ask for help in Teams from your fellow students. However you can use the videos, books, data camp, the internet and any other resources you wish. Do the best you can, this challenge is meant to strengthen your skills and you should think of it that way rather than as an exam. I will post the answers to this challenge after the due date.

Challenge #1 (10 points)

Part I (5 points)

The entry point into any language is to print 'Hello, World!' to the console. First write a function called `hello_world` that takes no arguments and prints 'Hello, World!' to the console. You can use the `print()` function or the `cat()` function in R to print to the console inside of a function.

Part II (5 points)

Now write a function called `greeting` which takes one argument, a person's name as a string, and returns a string that says 'Hello,' followed by the name that is passed in. For example if I pass in 'Robert', the function would return 'Hello, Robert!' (make sure you include the !). You can use the `paste()` function in R to combine strings together, see the example code below.

```
paste('My', 'name', 'is', 'Robert', ',', 'Welcome', 'to my', 'R course!')
```

```
## [1] "My name is Robert , Welcome to my R course!"
```

Challenge #2 (15 points)

Below is the equation for calculating the probability of observing some value within a normal distribution.

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}$$

Please write a funciton that takes three arguments. The first argument should be some value for which we want to calulate the probability (x). The second argument should be a default paramenter representing μ with a default of 0, and the third argument should be a default parameter representing σ with a defalut value of 1.

Hint: You should break the equation into pieces within the function using assignment so it is easier to calculate. The e in the equation can be calculated in R using the `exp()` function and you can use the `sqrt()` function to calculate the sqaure root and you can use the keyword `pi` for π

Also you CANNOT use the `dnorm()` function inside your function, however you can use it to check that your function returns the correct result

Challenge #3 (20 points)

There is a function in R called `unique()` which will return only the unique values of a vector. For example in the code snippet below we can see that while there are many numbers the only unique numbers in the vector are 1, 2, and 3 which `unique(my_vector)` returns.

```
my_vector <- c(1, 1, 2, 3, 3, 3, 2, 2, 2, 1)
unique(my_vector)
```

```
## [1] 1 2 3
```

If the vector contains missing values (NA) like in the code snippet below, the `unique()` function will return NA as one of the unique values.

```
my_vector_na <- c(1, 1, 2, 3, 3, 3, 2, 2, 2, 1, NA, NA, NA)
unique(my_vector_na)
```

```
## [1]  1  2  3 NA
```

Your callenge is to write a function called `unique_nona()` which takes in a vector and returns the unique values of the vector without any NA values. For example if I pass `c(1, 1, 2, 2, NA)`, I will get back `1 2` as a result NOT `1 2 NA`.

Hint: This will involve subsetting using a logical vector and testing whether there are NA values with the `is.na()` function. You will also need to know that the `!` means 'not' in R. That is if `is.na(x)` returns `TRUE` then `!is.na(x)` will return `FALSE`. We will learn more about 'logical operators' like this in the next section of the course but you only need to know about this one to complete this assignment.

```
x <- NA
is.na(x)
```

```
## [1] TRUE
```

```
!is.na(x)
```

```
## [1] FALSE
```

Challenge # 4 (30 points)

Part I (15 points)

Write a function called `append_to_vector` that will take two arguments, the first argument is a vector and the second argument is an item to add to the end of the vector. For the purpose of this challenge, we will assume that the original vector is a vector of numbers and the item to add is also a number. Your function should add the item to the end of the vector (i.e. append it). For example if I pass the vector `c(1, 2, 3)` and the item `4`, the function should return `1 2 3 4` like in the code below. Make sure to call the function to test it. **DO NOT USE THE BUILT-IN `append()` FUNCTION**

```
vector_1 <- c(1, 2, 3)
append_to_vector(vector_1, 4)
```

```
## [1] 1 2 3 4
```

Part II (15 points)

Write a function called `append_to_list` that will take two arguments, the first argument is a list, and the second argument is an item we want to append to the list similar to what we did with the `append_to_vector` function. If I pass the list `list(1, 2, 3)` and the item `4`, the function should return a list of 1, 2, 3, 4 like in the code below. But it should also work on more complex items like matricies or even other lists. Make sure to call the function to test it. **DO NOT USE THE BUILT-IN `append()` FUNCTION**

```
list_1 <- list(1, 2, 3)
append_to_list(list_1, 4)
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] 3
##
## [[4]]
## [1] 4
```

```
append_to_list(list_1, matrix(c(1,2,3,4), nrow = 2))
```

```
## [[1]]
## [1] 1
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] 3
##
## [[4]]
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

Challenge # 5 (15 points)

Use the code I provided below to create `my_vector`

```
my_vector <- sample(1:100, 2065, T)
```

- write code to find the length of my_vector
- what is the value of the 1000th element of my_vector
- create a new vector called my_new_vector which is the 100th though 200th elements of my_vector, use the `:` to do this subsetting
- in R, booleans can be converted to a number using `as.integer()`, and `TRUE` will convert to `1` and `FALSE` will convert to `0`. In addtion, in R we can use the `sum()` function to add up a bunch of number for us. Please find the number of NA values in `my_vector` using the `is.na()` function and the information above

Challenge #6 (10 points)

- Write code to install the `tibble` package
- Write code to load the `tibble` package into your R session
- load the `mtcars` dataset into R with this code `data("mtcars")` and then print the `mtcars` dataset
- convert `mtcars` to a tibble with `as_tibble()` and save it in a new binding called `df`
- subset `df` where we only want rows 6-10 and the columns 'mpg', 'hp', and 'gear'.