Applied Statistical Programming - Environments

Cassandra Custis, Berta Diaz, Zion Little, Alma Velazquez

2/14/2022

Write the R code to answer the following questions. Write the code, and then show what the computer returns when that code is run. Thoroughly comment your solutions.

You have until the beginning of class 2/16 at 10:00am to answer all of the questions below. You may use R, but not any online documentation. Submit the Rmarkdown and the knitted PDF to Canvas. Have one group member submit the activity with all group members listed at the top.

The Sorting Hat

The Hogwarts School of Witchcraft and Wizardry has hired you as an R programming wizard to replace its now-dilapidated sorting hat. In this activity you will create a sorting hat program to decide whether a given student belongs in Gryffindor, Slytherin, Ravenclaw,or Hufflepuff.

Students

Make a function that will output an S3 object of the class "student." The function will take in an argument "name." Each student should hold four values called, name, courage, ambition, intelligence, and effort. The function should *randomly* assign integer values to these traits ranging from 1-100.

```
# Code
make_student <- function(name) {
    new_student <- list(name = name, courage = sample(100, 1), ambition = sample(100,
        1), intelligence = sample(100, 1), effort = sample(100, 1))

    class(new_student) <- "student"
    return(new_student)

}

zion <- make_student("Zion")
class(zion)
```

```
## [1] "student"
```

Sorter

Create method for the generic `sort` that takes in as arguments:

- An object of the class `student`
- A matrix with four columns and four rows (X).

Let a be the vector of values for the attributes (courage, ambition, intelligence, effort). The sort method should perform the following calculations.

1. Calculate $X^T a$, which should result in a vector of length four.
2. If the first element of the resulting vector is largest, return "GRYFFINDOR!", if the second element is largest return "SLYTHERIN!", if the third is largest return "RAVENCLAW!", if the fourth is largest return "HUFFLEPUFF!"

```
# Code


rand_matrix <- matrix(c(sample(50, 16)), ncol = 4, nrow = 4)
rand_matrix
```

```
##      [,1] [,2] [,3] [,4]
## [1,]   30   15   16    8
## [2,]    6   47   34    5
## [3,]    4   45   48   28
## [4,]   35   23   14   10
```

```
sort <- function(x, ...) {
    UseMethod("sort")
}

sort.student <- function(student, x) {
    a <- c(student$courage, student$ambition, student$intelligence, student$effort)
    sort <- t(x) %*% a
    # print(sort)
    sort_result <- ifelse(which(sort == max(sort)) == 1, "GRYFFINDOR", ifelse(which(sort ==
        max(sort)) == 2, "SLYTHERIN", ifelse(which(sort == max(sort)) == 3, "RAVENCLAW",
        ifelse(which(sort == max(sort)) == 4, "HUFFLEPUFF", "ERROR"))))

    return(sort_result)
}

sort(zion, rand_matrix)
```

```
## [1] "SLYTHERIN"
```

Modifications

Alter the sort function the student included in the call is changed in the global environment such that student is assigned a second class (e.g., “Gryffindor”). Note that the student will now have two class labels.

```
# Code
sort.student <- function(student, x) {
    a <- c(student$courage, student$ambition, student$intelligence, student$effort)
    sort <- t(x) %*% a
    # print(sort)
    sort_result <- ifelse(which(sort == max(sort)) == 1, "GRYFFINDOR", ifelse(which(sort ==
        max(sort)) == 2, "SLYTHERIN", ifelse(which(sort == max(sort)) == 3, "RAVENCLAW",
        ifelse(which(sort == max(sort)) == 4, "HUFFLEPUFF", "ERROR"))))

    housed_student <- student
    class(housed_student) <- c(sort_result, class(student))

    assign(deparse(substitute(student)), housed_student, envir = .GlobalEnv)

    return(sort_result)
```

```
}

sort(zion, rand_matrix)
```

```
## [1] "SLYTHERIN"
```

```
class(zion)
```

```
## [1] "SLYTHERIN" "student"
```

Curfew

Create four new environments called, “Gryffindor_Tower”, “Black_Lake”, “Ravenclaw_Tower”, and “Basement”. These are the dormitories for the Gryffindor, Slytherin, Ravenclaw, and Hufflepuff students respectively.

Create a generic function called “curfew”, and then create curfew methods for each house that that takes a student as input and changes their environment to their appropriate dorm.

```
library(rlang)


Black_Lake <- env()
Ravenclaw_Tower <- env()
Basement <- env()
Gryffindor_Tower <- env()


curfew <- function(student) {

    house <- class(student)[1]

    if (house == "GRYFFINDOR") {
        assign(deparse(substitute(student)), student, envir = Gryffindor_Tower)
    }

    if (house == "SLYTHERIN") {
        assign(deparse(substitute(student)), student, envir = Black_Lake)
    }

    if (house == "RAVENCLAW") {
        assign(deparse(substitute(student)), student, envir = Ravenclaw_Tower)

    }

    if (house == "HUFFLEPUFF") {
        assign(deparse(substitute(student)), student, envir = Basement)

    }

}


curfew(zion)

env_print(Black_Lake)
```

```
## <environment: 0x7fbd44182f58>
## Parent: <environment: global>
## Bindings:
## * zion: <SLYTHERI>
```

```
# Without if statements

curfew <- function(student){
  common_rooms <- list("GRYFFINDOR"=Gryffindor_Tower,
                       "RAVENCLAW"=Ravenclaw_Tower,
                       "HUFFLEPUFF"=Basement,
                       "SLYTHERIN"=Black_Lake)


  house <- class(student)[1]
  targetEnv <- common_rooms[which(names(common_rooms)==house)][[1]]

    assign(deparse(substitute(student)), student, envir=targetEnv)

}


curfew(zion)

env_print(Black_Lake)
```

```
## <environment: 0x7fbd44182f58>
## Parent: <environment: global>
## Bindings:
## * zion: <SLYTHERI>
```