Challenge-5

Phua Zong Yao

2023-09-11

Questions

Question-1: Local Variable Shadowing

Create an R function that defines a global variable called `x` with a value of 5. Inside the function, declare a local variable also named `x` with a value of 10. Print the value of `x` both inside and outside the function to demonstrate shadowing.

Solutions:

```
# Enter code here

x <- 5

gg <- function(x){
 # local variable
   x<- 10
  return(x)
}

# Global variable x=5
x
```

```
## [1] 5
```

```
gg(x)
```

```
## [1] 10
```

Question-2: Modify Global Variable

Create an R function that takes an argument and adds it to a global variable called `total`. Call the function multiple times with different arguments to accumulate the values in `total`.

Solutions:

```
# Enter code here

total <- 0
add_to_total <- function(value){
total <<- total + value
}

add_to_total(5)
add_to_total(15)
add_to_total(25)
add_to_total(30)

total
```

```
## [1] 75
```

Question-3: Global and Local Interaction

Write an R program that includes a global variable `total` with an initial value of 100. Create a function that takes an argument, adds it to `total`, and returns the updated `total`. Demonstrate how this function interacts with the global variable.

Solutions:

```
# Enter code here

total <- 100
add_total <- function(value){
total <<- total + value
return(total)
}

new_total <- add_total(50)
new_total <- add_total(30)

total
```

```
## [1] 180
```

Question-4: Nested Functions

Define a function `outer_function` that declares a local variable `x` with a value of 5. Inside `outer_function`, define another function `inner_function` that prints the value of `x`. Call both functions to show how the inner function accesses the variable from the outer function's scope.

Solutions:

```
# Enter code here

# define outer function
outer_function <- function() {
  x = 5

  # define inner function
  inner_function <- function() {
    print(x)
  }

  inner_function()

}

# call the outer function
outer_function()
```

```
## [1] 5
```

Question-5: Meme Generator Function

Create a function that takes a text input and generates a humorous meme with the text overlaid on an image of your choice. You can use the `magick` package for image manipulation. You can find more details about the commands offered by the package, with some examples of annotating images here: https://cran.r-project.org/web/packages/magick/vignettes/intro.html (https://cran.r-project.org/web/packages/magick/vignettes/intro.html)

Solutions:

```
# Enter code here

library(magick)
```

```
## Linking to ImageMagick 6.9.12.93
## Enabled features: cairo, freetype, fftw, ghostscript, heic, lcms, pango, raw, rsvg, webp
## Disabled features: fontconfig, x11
```

```
generate_meme <- function(input_text, image_path){
  # Load the base image
  base_image <- image_read(image_path)
  image_annotate(base_image,input_text)
}

input_text <- "That moment when you realize\n you forgot your umbrella!"
image_path <- "valorantsage.jpg"

generate_meme(input_text, image_path)
```

Question-6: Text Analysis Game

Develop a text analysis game in which the user inputs a sentence, and the R function provides statistics like the number of words, characters, and average word length. Reward the user with a “communication skill level” based on their input.

Solutions:

```
# Enter code here

library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.3     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to be
come errors
```

```
play_game <- function(user_input){
  # user_input <- readline(prompt = "Enter a sentence:")


# calculate the number of characters

num_char <- nchar(user_input)

# calculate the number of words

num_words <- strsplit(user_input," ")[[1]] %>% length()

# calculate the average word length

avg_word_length <- num_char / num_words

# calculate the communication skill level

communication_skill_level <- case_when(
  avg_word_length >= 1 & num_words >= 5 ~ "zai kia",
  avg_word_length >= 1 & num_words >= 4 ~ "nice one",
  avg_word_length >= 1 & num_words >= 3 ~ "not bad",
  TRUE ~ "Get Better"
)
return(list(
  num_words = num_words,
  num_char = num_char,
  avg_word_length = avg_word_length,
  communication_skill_level = communication_skill_level
))


}

play_game("have a nice day")
```

```
## $num_words
## [1] 4
##
## $num_char
## [1] 15
##
## $avg_word_length
## [1] 3.75
##
## $communication_skill_level
## [1] "nice one"
```