Challenge-5

Daniel Tan

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:

```r
# Enter code here

x <- 5

shadow_demo <- function() {
  x <- 10
  cat("Inside the function: x =", x, "\n")
}

shadow_demo()
```

```
## Inside the function: x = 10
```

```r
cat("Outside the function: x =", x, "\n")
```

```
## Outside the function: x = 5
```

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:

```r
# Enter code here
total <- 0

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

add_to_total(5)
add_to_total(10)
add_to_total(7)

cat("Accumulated total:", total, "\n")
```

```
## Accumulated total: 22
```

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:

```r
# Enter code here
total <- 100

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

cat("Initial total:", total, "\n")
```

```
## Initial total: 100
```

```r
new_total <- add_to_total(25)
cat("Updated total after adding 25:", new_total, "\n")
```

```
## Updated total after adding 25: 125
```

```r
new_total <- add_to_total(50)
cat("Updated total after adding 50:", new_total, "\n")
```

```
## Updated total after adding 50: 175
```

```r
new_total <- add_to_total(75)
cat("Updated total after adding 75:", new_total, "\n")
```

```
## Updated total after adding 75: 250
```

```r
cat("Final total:", total, "\n")
```

```
## Final total: 250
```

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:

```r
# Enter code here
outer_function <- function() {
  x <- 5

  inner_function <- function() {
    cat("Value of 'x' from inner_function:", x, "\n")
  }

  inner_function()

  cat("Message from outer_function\n")
}

outer_function()
```

```
## Value of 'x' from inner_function: 5
## Message from outer_function
```

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

Solutions:

```r
# Enter code here
library(magick)
```

```
## Warning: package 'magick' was built under R version 4.2.3
```

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

```r
generate_meme <- function(text, image_path, output_path) {

  img <- image_read(image_path)

  text_color <- "white"
  text_size <- 40
  text_font <- "Arial-Bold"

  img_with_text <- img %>%
    image_annotate(text,
                   color = text_color,
                   size = text_size,
                   font = text_font,
                   gravity = "center",
                   location = "+0+120")

  image_write(img_with_text, path = output_path)

  img_with_text
}

text_to_add <- "It's a doggy dog world..."
input_image_path <- "dog.jpg"
output_image_path <- "output_meme.jpg"

# Generate the meme
generate_meme(text_to_add, input_image_path, output_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:

```r
# Enter code here
text_analysis_game <- function(user_input) {
  cat("Welcome to the Text Analysis Game!\n")
  cat("You entered the following sentence:\n", user_input, "\n")

  if (nchar(user_input) == 0) {
    cat("You didn't enter a sentence. Please try again.\n")
    return()
  }

  num_words <- length(unlist(strsplit(user_input, "\\s+")))
  num_chars <- nchar(user_input)
  avg_word_length <- num_chars / num_words

  cat("\nText Statistics:\n")
  cat("Number of words:", num_words, "\n")
  cat("Number of characters:", num_chars, "\n")
  cat("Average word length:", avg_word_length, "\n")

  if (avg_word_length <= 4) {
    skill_level <- "Novice Communicator"
  } else if (avg_word_length <= 6) {
    skill_level <- "Intermediate Communicator"
  } else {
    skill_level <- "Advanced Communicator"
  }

  cat("\nCommunication Skill Level:", skill_level, "\n")

  cat("\nThanks for playing!\n")
}

user_input <- "Man, I sure am hungry, maybe I should go to the GrubHub and Git some Grub from the GrubHu
text_analysis_game(user_input)
```

```
## Welcome to the Text Analysis Game!
## You entered the following sentence:
##  Man, I sure am hungry, maybe I should go to the GrubHub and Git some Grub from the GrubHub
##
## Text Statistics:
## Number of words: 19
## Number of characters: 90
## Average word length: 4.736842
##
## Communication Skill Level: Intermediate Communicator
##
## Thanks for playing!
```