Challenge-5

Janelle 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
# Define a global variable x with a value of 5
x <- 5

# Create a function that declares a local variable x with a value of 10
function_1 <- function() {
  x <- 10
  cat("Inside function_1, x =", x, "\n")
}

# Print the global x before calling the function
cat("Before calling function_1, x =", x, "\n")
```

```
## Before calling function_1, x = 5
```

```r
# Call the function
function_1()
```

```
## Inside function_1, x = 10
```

```r
# Print the global x after calling the function
cat("After calling function_1, x =", x, "\n")
```

```
## After calling function_1, 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 = 5

function_2 <- function(x) {
  total <<- total + x
}

function_2(5)
function_2(6)
cat("The updated total is:", total, "\n")
```

```
## The updated total is: 16
```

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

function_3 <- function(x) {
  total <<- total + x
  return(total)
}

function_3(5)
```

```
## [1] 105
```

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() {
    print(x)
  }
  # Call the inner function from within the outer function
  inner_function()
}

outer_function(5)
```

outer_function doesn't actually call inner_function or execute any code that would print the value of x

-> call inner_function within outer_function

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

```
#Explanation:

#You define the outer_function with a default value of 5 for the x parameter. Within outer_function, yo

#When you call outer_function(5), it calls inner_function() from within its scope.

#Since inner_function is inside outer_function, it has access to the x variable from the 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:

```
# Enter code here
# Install and load the magick package
# Specify a CRAN mirror
options(repos = c(CRAN = "https://cloud.r-project.org"))
install.packages("magick")
```

```
## Installing package into 'C:/Users/janel/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
## package 'magick' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\janel\AppData\Local\Temp\RtmpMfbruY\downloaded_packages
```

```
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(text, output_filename) {
  # Load the base image (replace "sample.jpg" with your image file path)
  base_image <- image_read("C:/Users/janel/OneDrive/Documents/Y2S1 NM2207/Week 5/meme_base_image.jpg")

  # Define font properties
  font_size <- 25
  font_color <- "black"
  font_path <- "C:/Windows/Fonts/BASKVILL.ttf"  # Replace with the path to your font file

  # Create a drawing object
  meme_draw <- image_draw(base_image)

  # Add text to the image using the 'annotate' function
  meme_image <- meme_draw %>%
    image_annotate(
      text = text,
```

```r
      color = font_color,
      size = font_size,
      location = "+20+20",  # X and Y coordinates for text position
      font = font_path
    )

  # Specify the output path to your "Downloads" directory
  output_path <- file.path(Sys.getenv("USERPROFILE"), "Downloads", output_filename)

  # Save the meme to the "Downloads" directory
  image_write(meme_image, path = output_path)

  cat("Meme generated and saved to", output_path, "\n")
}

# Usage of the function:
generate_meme("When there's a week 6 quiz...", "output_meme.jpg")
```

```
## Meme generated and saved to C:\Users\janel/Downloads/output_meme.jpg
```

```r
#output file name is called output_meme.jpg
```

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
# Function to analyze user input and calculate communication skill level

# Function to analyze user input and calculate communication skill level
analyze_text <- function(user_input) {
  # Count the number of characters and words in the input
  num_characters <- nchar(user_input)
  words <- strsplit(user_input, "\\s+")
  num_words <- length(words[[1]])

  # Calculate the average word length
  avg_word_length <- num_characters / num_words

  # Determine the communication skill level
  skill_level <- if (num_words > 0) {
  avg_word_length <- num_characters / num_words
  if (avg_word_length >= 4 && avg_word_length <= 6) {
    skill_level <- "Intermediate"
  } else if (avg_word_length < 4) {
    skill_level <- "Beginner"
  } else {
    skill_level <- "Advanced"
  }
} else {
  skill_level <- "Invalid Input"  # Handle the case when there are no words
}
  #don't use switch bc they use logical variable type, use if else statements

  # Prepare the result message
  result_message <- paste(
    "Number of characters:", num_characters,
    "\nNumber of words:", num_words,
    "\nAverage word length:", round(avg_word_length, 2),
    "\nCommunication skill level:", skill_level  # This should work correctly now
  )

  return(result_message)
}

cat("Welcome to the Text Analysis Game!\n")
```

```
## Welcome to the Text Analysis Game!
```

```
cat("Enter a sentence to analyze: ")
```

```
## Enter a sentence to analyze:
```

```
user_input <- readline(prompt = "")
```

```
# Check if the user wants to quit
if (tolower(user_input) == "quit") {
  cat("Goodbye! Thanks for playing.\n")
} else {
  # Analyze the user's input
  result <- analyze_text(user_input)
  cat("\nAnalysis Result:\n")
  cat(result, "\n")
}
```

```
##
## Analysis Result:
## Number of characters: 0
## Number of words: 0
## Average word length: NaN
## Communication skill level: Invalid Input
```

```
# Testing the function with a specific input
analyze_text("hello how are you")
```

```
## [1] "Number of characters: 17 \nNumber of words: 4 \nAverage word length: 4.25 \nCommunication skill
```

```
#check for infinite loops if code doesn't stop rendering
```