*#!/bin/bash*

*# This is a very simple Bash script demonstrating basic functionalities.*

*# It's designed to be easily understood by beginners.*

*# ----------------------------------------------------------------------*

*# SECTION 1: Variables and basic output*

*# ----------------------------------------------------------------------*

*# Line 1: Shebang - specifies the interpreter for the script.*

*# It tells the operating system to execute this script with bash.*

*# Line 2: A comment line - these start with a '#' and are ignored by the interpreter.*

*# They are used to explain the code and make it easier to understand.*

*# Line 3: Declare a variable called 'script\_name' and assign it a string value.*

script\_name="My First Elementary Bash Script"

*# Line 4: Use 'echo' to print a string to the terminal.*

echo "----------------------------------------------------"

*# Line 5: Print the value of the 'script\_name' variable.*

echo "$script\_name"

*# Line 6: Another separator line for better readability.*

echo "----------------------------------------------------"

*# Line 7: Declare another variable for a welcome message.*

greeting="Welcome to Bash Scripting!"

*# Line 8: Print the greeting message.*

echo "$greeting"

*# Line 9: Declare a numeric variable.*

current\_year=$(date +%Y) *# Get the current year dynamically.*

*# Line 10: Display the current year.*

echo "It's currently the year $current\_year."

*# ----------------------------------------------------------------------*

*# SECTION 2: User Input*

*# ----------------------------------------------------------------------*

*# Line 11: Prompt the user for their name.*

echo -n "Please enter your name: "

*# Line 12: Read the user's input and store it in a variable called 'user\_name'.*

read user\_name

*# Line 13: Display a personalized greeting.*

echo "Hello, $user\_name! Glad you're here."

*# Line 14: Prompt for their favorite color.*

echo -n "What's your favorite color? "

*# Line 15: Read the color input.*

read favorite\_color

*# Line 16: Confirm their favorite color.*

echo "$user\_name, your favorite color is $favorite\_color."

*# ----------------------------------------------------------------------*

*# SECTION 3: Conditional Statements (If-Else)*

*# ----------------------------------------------------------------------*

*# Line 17: Introduce a conditional statement based on the favorite color.*

if [ "$favorite\_color" == "blue" ]; then *# Check if the color is "blue".*

*# Line 18: If true, print a specific message.*

echo "Ah, blue! A popular choice."

elif [ "$favorite\_color" == "green" ]; then *# Check if the color is "green" if not blue.*

*# Line 19: If true, print a specific message.*

echo "Green is the color of nature."

else

*# Line 20: If neither, print a general message.*

echo "That's an interesting color choice!"

fi

*# Line 21: Check if a file exists (using an arbitrary filename for demonstration).*

file\_to\_check="test\_file.txt"

*# Line 22: Introduce an if-else statement to check for file existence.*

if [ -e "$file\_to\_check" ]; then *# The -e operator checks if a file exists.*

*# Line 23: If the file exists, print a message.*

echo "File '$file\_to\_check' already exists."

else

*# Line 24: If the file does not exist, print a message.*

echo "File '$file\_to\_check' does not exist."

*# Line 25: Create the file.*

touch "$file\_to\_check"

*# Line 26: Confirm file creation.*

echo "Created '$file\_to\_check'."

fi

*# ----------------------------------------------------------------------*

*# SECTION 4: Loops*

*# ----------------------------------------------------------------------*

*# Line 27: Introduce a simple for loop.*

echo "Counting to 5 with a for loop:"

for i in 1 2 3 4 5; do

*# Line 28: Print the current number in the loop.*

echo "Number: $i"

done

*# Line 29: Introduce a while loop.*

echo "Counting down from 3 with a while loop:"

count=3

while [ $count -gt 0 ]; do *# Loop as long as 'count' is greater than 0.*

*# Line 30: Print the current count.*

echo "Count: $count"

*# Line 31: Decrement the count.*

((count--))

done

echo "Blast off!"

*# ----------------------------------------------------------------------*

*# SECTION 5: More advanced variable usage and arithmetic*

*# ----------------------------------------------------------------------*

*# Line 32: Declare an array.*

fruits=("Apple" "Banana" "Cherry" "Date")

*# Line 33: Print all elements of the array.*

echo "My favorite fruits are: ${fruits[@]}"

*# Line 34: Access a specific element of the array.*

echo "The first fruit is: ${fruits[0]}"

*# Line 35: Arithmetic operations.*

num1=10

num2=5

*# Line 36: Perform addition.*

sum=$((num1 + num2))

echo "Sum of $num1 and $num2 is: $sum"

*# Line 37: Perform subtraction.*

difference=$((num1 - num2))

echo "Difference of $num1 and $num2 is: $difference"

*# Line 38: Perform multiplication.*

product=$((num1 \* num2))

echo "Product of $num1 and $num2 is: $product"

*# Line 39: Perform division.*

quotient=$((num1 / num2))

echo "Quotient of $num1 and $num2 is: $quotient"

*# Line 40: Perform modulo (remainder) operation.*

remainder=$((num1 % 3))

echo "Remainder of $num1 divided by 3 is: $remainder"

*# ----------------------------------------------------------------------*

*# SECTION 6: Functions*

*# ----------------------------------------------------------------------*

*# Line 41: Define a simple function.*

my\_function() {

*# Line 42: Print a message from within the function.*

echo "This is a function called 'my\_function'."

}

*# Line 43: Call the function.*

my\_function

*# Line 44: Function with arguments.*

greet\_user() {

*# Line 45: Print a personalized greeting using the argument passed to the function.*

echo "Hello there, $1!" *# $1 refers to the first argument.*

}

*# Line 46: Call the function with an argument.*

greet\_user "Bash Learner"

*# ----------------------------------------------------------------------*

*# SECTION 7: File Operations (Simple)*

*# ----------------------------------------------------------------------*

*# Line 47: Define a filename for file operations.*

data\_file="data.txt"

*# Line 48: Write some data to the file, overwriting existing content.*

echo "This is the first line." > "$data\_file"

*# Line 49: Append another line to the file.*

echo "This is the second line." >> "$data\_file"

*# Line 50: Append yet another line.*

echo "And a third line." >> "$data\_file"

*# Line 51: Read the content of the file and display it.*

echo "Content of '$data\_file':"

cat "$data\_file"

*# Line 52: Check if the file is empty.*

if [ -s "$data\_file" ]; then *# -s checks if the file has a size greater than 0.*

echo "'$data\_file' is not empty."

else

echo "'$data\_file' is empty."

fi

*# Line 53: Remove the file.*

rm "$data\_file"

*# Line 54: Confirm file removal.*

echo "Removed '$data\_file'."

*# ----------------------------------------------------------------------*

*# SECTION 8: Loops for processing data (e.g., file names)*

*# ----------------------------------------------------------------------*

*# Line 55: Create some dummy files for demonstrating loops with file processing.*

touch file1.txt file2.log file3.txt file4.bak

*# Line 56: Loop through all .txt files in the current directory.*

echo "Processing .txt files:"

for file in \*.txt; do

*# Line 57: Print the name of each .txt file.*

echo " Found: $file"

*# Line 58: Add a line to each file.*

echo "Added line to $file" >> "$file"

done

*# Line 59: Remove the dummy files.*

rm file1.txt file2.log file3.txt file4.bak

*# ----------------------------------------------------------------------*

*# SECTION 9: Interactive menu (using `select`)*

*# ----------------------------------------------------------------------*

*# Line 60: Set the prompt for the 'select' menu.*

PS3="Choose an option: "

*# Line 61: Define the options for the menu.*

options=("Option 1" "Option 2" "Exit")

*# Line 62: Create the select loop.*

select choice in "${options[@]}"; do

*# Line 63: Use a case statement to handle user choices.*

case "$choice" in

"Option 1")

*# Line 64: Action for Option 1.*

echo "You chose Option 1."

;;

"Option 2")

*# Line 65: Action for Option 2.*

echo "You chose Option 2."

;;

"Exit")

*# Line 66: Exit the script if the user chooses 'Exit'.*

echo "Exiting script."

break *# Break out of the select loop.*

;;

\*)

*# Line 67: Handle invalid input.*

echo "Invalid option: $REPLY" *# $REPLY stores the raw input.*

;;

esac

done

*# ----------------------------------------------------------------------*

*# SECTION 10: More example lines to reach ~100*

*# ----------------------------------------------------------------------*

*# Line 68: Another echo statement for spacing.*

echo ""

*# Line 69: Print the current date and time.*

echo "Current Date and Time: $(date)"

*# Line 70: Display the current working directory.*

echo "Current Directory: $(pwd)"

*# Line 71: List the files in the current directory.*

echo "Files in current directory:"

ls -l

*# Line 72: Print a message about the script's completion.*

echo "Script finished executing."

*# Line 73: Add more empty lines for spacing and to reach the line count.*