

Exercise 1:

Write a shell script that prints "Shell Scripting is Fun!" to the screen.

Hint 1: Remember to make the shell script executable with the `chmod` command.

Hint 2: Remember to start your script with a shebang!

Answer :

```
#!/bin/bash
# This script prints a simple message

echo "Shell Scripting is Fun!"
```

`chmod +x fun.sh` now run the script `./fun.sh`

Exercise 2:

Modify the shell script from exercise 1 to include a variable. The variable will hold the contents of the message "Shell Scripting is Fun!".

```
#!/bin/bash
# This script uses a variable to store a message

MESSAGE="Shell Scripting is Fun!"
echo "$MESSAGE"
```

Exercise 3:

Store the output of the command "hostname" in a variable. Display "This script is running on _____. " where "_____" is the output of the "hostname" command.

Hint: It's a best practice to use the `${VARIABLE}` syntax if there is text or characters that directly precede or follow the variable.

```
#!/bin/bash
# This script displays the system's hostname using a variable

HOSTNAME=$(hostname)
echo "This script is running on ${HOSTNAME}."
```

Exercise 4: Write a shell script to check to see if the file `/etc/shadow` exists. If it does exist, display `"Shadow passwords are enabled."` Next, check to see if you can write to the file. If you can, display `"You have permissions to edit /etc/shadow."` If you cannot, display `"You do NOT have permissions to edit /etc/shadow."`

```
#!/bin/bash
# This script checks if /etc/shadow exists and whether the user has write permission.

FILE="/etc/shadow"

# Check if the file exists
if [ -e "$FILE" ]; then
    echo "Shadow passwords are enabled."

    # Check if the file is writable
    if [ -w "$FILE" ]; then
        echo "You have permissions to edit ${FILE}."
    else
        echo "You do NOT have permissions to edit ${FILE}."
    fi
else
    echo "${FILE} does not exist on this system."
fi
```

Exercise 5:

Write a shell script that displays `"man"`, `"bear"`, `"pig"`, `"dog"`, `"cat"`, and `"sheep"` to the screen with each appearing on a separate line. Try to do this in as few lines as possible.

Hint: Loops can be used to perform repetitive tasks.

```
#!/bin/bash
# This script prints a list of animals, each on a separate line.

for ANIMAL in man bear pig dog cat sheep
do
    echo "$ANIMAL"
done
```

Exercise 6:

Write a shell script that prompts the user for a name of a file or directory and reports if it is a regular file, a directory, or other type of file. Also perform an `ls` command against the file or directory with the long listing option.

```
#!/bin/bash
# This script asks for a file or directory name and reports its type, then lists details.

read -p "Enter the name of a file or directory: " FILE

if [ -e "$FILE" ]; then
    if [ -f "$FILE" ]; then
        echo "${FILE} is a regular file."
    elif [ -d "$FILE" ]; then
        echo "${FILE} is a directory."
    else
        echo "${FILE} is another type of file."
    fi
    echo
    echo "Details for ${FILE}:"
    ls -l "$FILE"
else
    echo "The file or directory '${FILE}' does not exist."
fi
```

Exercise 7:

Modify the previous script so that it accepts the file or directory name as an argument instead of prompting the user to enter it.

```
#!/bin/bash
# This script checks a file or directory passed as an argument and reports its type.

# Check if an argument was provided
if [ $# -eq 0 ]; then
    echo "Usage: $0 <file_or_directory_name>"
    exit 1
fi

FILE=$1

if [ -e "$FILE" ]; then
    if [ -f "$FILE" ]; then
```

```

    echo "${FILE} is a regular file."
elif [ -d "$FILE" ]; then
    echo "${FILE} is a directory."
else
    echo "${FILE} is another type of file."
fi
echo
echo "Details for ${FILE}:"
ls -l "$FILE"
else
    echo "The file or directory '${FILE}' does not exist."
fi

```

Exercise 8:

Modify the previous script to accept an unlimited number of files and directories as arguments. Hint: You'll want to use a special variable.

```

#!/bin/bash
# This script checks an unlimited number of files or directories passed as arguments.

# Check if at least one argument is provided
if [ $# -eq 0 ]; then
    echo "Usage: $0 <file_or_directory_name> [additional_files_or_directories...]"
    exit 1
fi

# Loop through all arguments
for FILE in "$@"
do
    if [ -e "$FILE" ]; then
        if [ -f "$FILE" ]; then
            echo "${FILE} is a regular file."
        elif [ -d "$FILE" ]; then
            echo "${FILE} is a directory."
        else
            echo "${FILE} is another type of file."
        fi
        echo "Details for ${FILE}:"
        ls -l "$FILE"
        echo "-----"
    else
        echo "The file or directory '${FILE}' does not exist."
    fi
done

```

```
    echo "-----"  
fi  
done
```