Assignment 1: Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found"

```
FILE="myfile.txt"

if [ -e "$FILE" ]; then
   echo "File exists"
else
   echo "File not found"
fi
```

Assignment 2: Write a script that reads numbers from the user until they enter '0'. The script should also print whether each number is odd or even.

```
while true; do
   read num
   if [ "$num" -eq 0 ]; then
      break
   fi

   if [ $((num % 2)) -eq 0 ]; then
      echo "Even"
   else
      echo "Odd"
   fi
done
```

Assignment 3: Create a function that takes a filename as an argument and prints the number of lines in the file. Call this function from your script with different filenames

```
count_lines() {
   file=$1
   if [ -f "$file" ]; then
      lines=$(wc -l < "$file")
      echo "$file has $lines lines"
   else
      echo "File not found: $file"
   fi
}
count_lines file1.txt
count_lines file2.txt
```

Assignment 4: Write a script that creates a directory named TestDir and inside it, creates ten files named File1.txt, File2.txt, ... File10.txt. Each file should contain its filename as its content (e.g., File1.txt contains "File1.txt").

```
for i in {1..10}; do
   filename="File$i.txt"
   echo "$filename" > "TestDir/$filename"
done
```

Assignment 5: Modify the script to handle errors, such as the directory already existing or lacking permissions to create files. Add a debugging mode that prints additional information when enabled.

```
DIR="my_project"
DEBUG=true

[ "$DEBUG" = true ] && echo "Creating directory: $DIR"

if mkdir "$DIR" 2>/dev/null; then
  [ "$DEBUG" = true ] && echo "Directory created."
  echo "echo 'Hello, world!'" > "$DIR/main.sh"
  [ "$DEBUG" = true ] && echo "File main.sh created."
else
  if [ -d "$DIR" ]; then
    echo "Error: Directory '$DIR' already exists."
  else
    echo "Error: Cannot create directory '$DIR'. Check permissions."
  fi
fi
```

Assignment 6: Given a sample log file, write a script using grep to extract all lines containing "ERROR". Use awk to print the date, time, and error message of each extracted line. Data Processing with sed

```
grep "ERROR" sample.log | awk '{print $1, $2, substr($0, index($0,$3))}' | sed 's/ERROR //'
```

Assignment 7: Create a script that takes a text file and replaces all occurrences of "old_text" with "new_text". Use sed to perform this operation and output the result to a new file.

```
input_file="input.txt"
output_file="output.txt"

sed 's/old_text/new_text/g' "$input_file" > "$output_file"
```