Functions

```
#!/bin/bash

# A simple function that prints a greeting message
function say_hello {
   echo "Hello, $1!"
}

# Call the function with a name argument
say_hello "Alice"

# Call the function with a different name argument
say_hello "Bob"
```

In this script, we define a function called say_hello that takes one argument. The function simply prints a greeting message that includes the argument. We then call the function twice with different name arguments, which causes it to print two different greeting messages.
Functions in shell are defined using the function keyword followed by the function name and the function body enclosed in curly braces. The function can take zero or more arguments, which are accessed using positional parameters like $1, $2, etc.

Functions in shell are useful for encapsulating reusable code and making scripts more modular and easier to maintain. They can also help to avoid code duplication and make scripts more readable by breaking down complex tasks into smaller, more manageable pieces.

```
#!/bin/bash

function add {
   local result=$(( $1 + $2 ))
   echo $result
}

sum=$(add 3 5)

echo "The sum is: $sum"
```

In this script, we define a function called add that takes two arguments and returns their sum. The function uses a local variable called result to store the sum of the two arguments, which is then

printed to the standard output using the echo command. We then call the function with the arguments 3 and 5, and save the result in a variable called sum. Finally, we print the result to the standard output using the echo command.

Functions in shell can also use local variables, which are only accessible within the function and do not interfere with variables defined outside of the function. This can be useful for avoiding naming conflicts and making functions more self-contained.

```
#!/bin/bash

function file_exists {
   if [ -f "$1" ]; then
      echo "The file $1 exists."
   else
      echo "The file $1 does not exist."
   fi
}

file_exists "example.txt"
```

In this script, we define a function called file_exists that takes one argument, which is the name of a file. The function uses the if statement and the -f test operator to check if the file exists. If the file exists, the function prints a message saying so. If the file does not exist, the function prints a different message saying so.

We then call the function with the argument example.txt, which causes it to check if the file exists and print the appropriate message.