



# BridgeLabz

Employability Delivered

Programming  
Constructs –  
Functions

## 4. Functions

- Functions in Bash Scripting are a great way to reuse code.
- Think of a function as a small script within a script. It's a small chunk of code which you may call multiple times within your script. They are particularly useful if you have certain tasks which need to be performed several times.

# Function Example

---

```
#!/bin/bash -x

function myfunc() {
    echo $1
}
result="$( myfunc $((RANDOM%2)) )"
if [ $result -eq 1 ]
then
    echo "success"
else
    echo "failure"
fi
functionTest.sh (END)
```

```
++ myfunc 1
++ echo 1
+ result=1
+ '[' 1 -eq 1 ']'
+ echo success
success
```



**UC 7**

Refactor the Code  
to write a function  
to get work hours

# Calculating Wages till Number of Working Days or Total Working Hours per month is Reached

```
#!/bin/bash -x
```

```
# CONSTANTS FOR THE PROGRAM
```

```
IS_PART_TIME=1;
```

```
IS_FULL_TIME=2;
```

```
MAX_HRS_IN_MONTH=4;
```

```
EMP_RATE_PER_HR=20;
```

```
NUM_WORKING_DAYS=20;
```

```
# VARIABLES
```

```
totalEmpHr=0;
```

```
totalWorkingDays=0;
```

```
function getWorkingHours() {
```

```
    case $1 in
```

```
        $IS_FULL_TIME)
```

```
            workHours=8
```

```
            ;;
```

```
        $IS_PART_TIME)
```

```
            workHours=4
```

```
            ;;
```

```
        *)
```

```
            workHours=0
```

```
            ;;
```

```
    esac
```

```
    echo $workHours
```

```
}
```

```
while [[ $totalWorkHours -lt $MAX_HRS_IN_MONTH &&  
        $totalWorkingDays -lt $NUM_WORKING_DAYS ]];
```

```
do
```

```
    ((totalWorkingDays++))
```

```
    workHours="$( getWorkingHours $((RANDOM%3)) )"*
```

```
    totalWorkHours=$((totalWorkHours+workHours))
```

```
done
```

```
totalSalary=$((totalWorkHours*$EMP_RATE_PER_HR));
```

```
|Narayans-MacBook-Pro:TerminalCommands narayan$ ./empWageFunction.sh
```

```
+ IS_PART_TIME=1
```

```
+ IS_FULL_TIME=2
```

```
+ MAX_HRS_IN_MONTH=4
```

```
+ EMP_RATE_PER_HR=20
```

```
+ NUM_WORKING_DAYS=20
```

```
+ totalEmpHr=0
```

```
+ totalWorkingDays=0
```

```
+ [[ '' -lt 4 ]]
```

```
+ [[ 0 -lt 20 ]]
```

```
+ (( totalWorkingDays++ ))
```

```
++ getWorkingHours 2
```

```
++ case $1 in
```

```
++ workHours=8
```

```
++ echo 8
```

```
+ workHours=8
```

```
+ totalWorkHours=8
```

```
+ [[ 0 -lt 4 ]]
```

```
+ totalSalary=160
```

# Functions Practice Problems

1. Help user find degF or degC based on their Conversion Selection. Use Case Statement and ensure that the inputs are within the Freezing Point ( 0 °C / 32 °F ) and the Boiling Point of Water ( 100 °C / 212 °F )
  - a.  $\text{degF} = (\text{degC} * 9/5) + 32$
  - b.  $\text{degC} = (\text{degF} - 32) * 5/9$
2. Write a function to check if the two numbers are Palindromes
3. Take a number from user and check if the number is a Prime then show that its palindrome is also prime
  - a. Write function check if number is Prime
  - b. Write function to get the Palindrome
  - c. Check if the Palindrome number is also prime



# BridgeLabz

Employability Delivered

Thank  
You