

# **Shell Scripts**

## For Beginners

KODEKLOUD





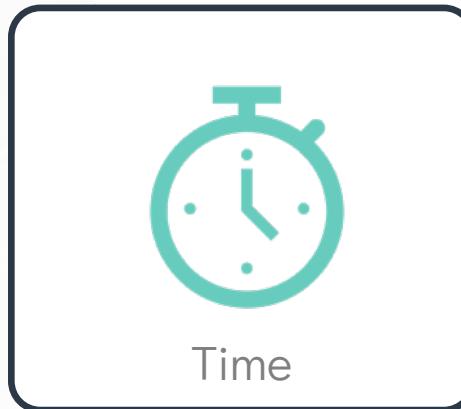
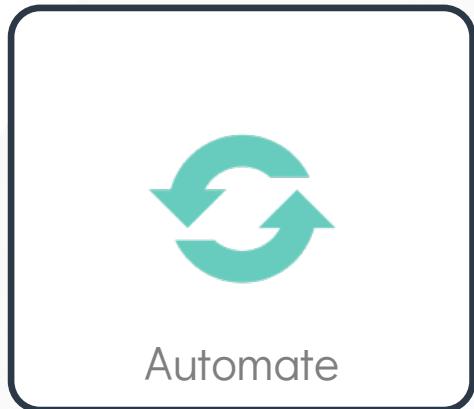
**Vijin  
Palazhi**



**Mumshad  
Mannambeth**

...

# Why Shell Scripts?



# Why Shell Scripts?

- Automate Daily Backups
- Automate Installation and Patching of software on multiple servers
- Monitor system periodically
- Raise alarms and send notifications
- Troubleshooting and Audits
- Many More



# Who is this for?

- Systems Administrators
- Developers
- IT Engineers
- Absolute Beginners
- No Programming Experience

...

# Objectives

What are Shell Scripts?

Variables

Control Logic

Loops

Executing a shell script

Shebang

Arithmetic Operations

Best Practices/IDEs

...



# Pre-Requisites

- Linux Basics
- Command line basics
- No programming knowledge required

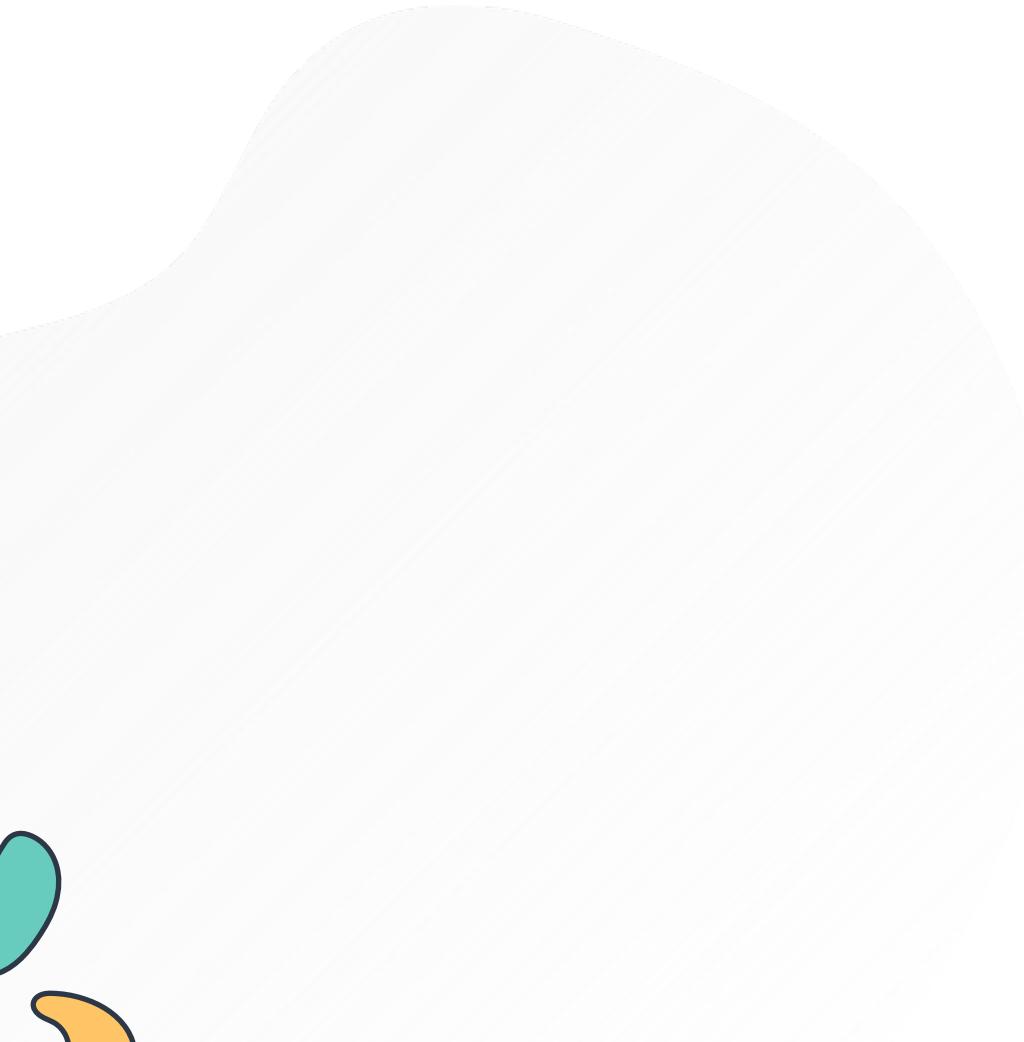


# Introduction



Shell Scripting

...



...



# Launch Sequence

1. Start Auxiliary Power
2. Switch to Internal Power
3. Auto Sequence Start
4. Main Engine Start
5. Lift Off



...

# Launch Sequence

1. Start Auxiliary Power

rocket-start-power

ls

rocket-ls

2. Switch to Internal Power

rocket-internal-power

useradd

rocket-add

3. Auto Sequence Start

rocket-start-sequence

mkdir

rocket-del

4. Main Engine Start

rocket-start-engine

which

rocket-status

5. Lift Off

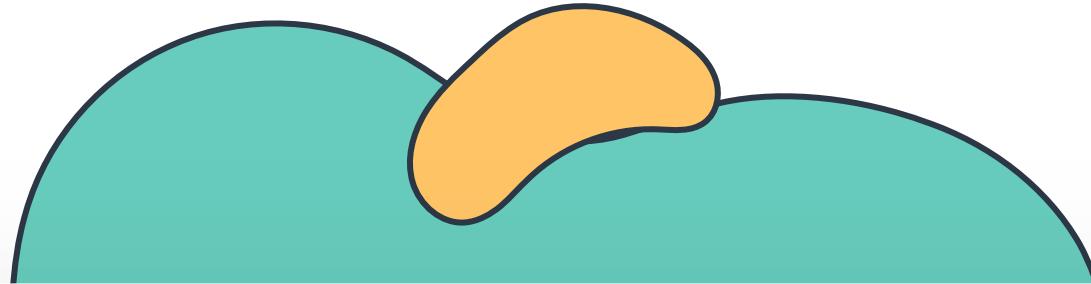
rocket-lift-off

dir

rocket-debug

# HANDS-ON LABS





# Creating your first **Script**



Shell Scripting



# Creating your First Script

```
$ mkdir lunar-mission
```

```
$ rocket-add lunar-mission
```

```
$ rocket-start-power lunar-mission  
$ rocket-internal-power lunar-mission  
$ rocket-start-sequence lunar-mission  
$ rocket-start-engine lunar-mission  
$ rocket-lift-off lunar-mission
```

```
$ rocket-status lunar-mission
```

```
$ bash create-and-launch-rocket.sh
```

```
create-and-launch-rocket.sh
```

```
✓ mkdir lunar-mission
```

```
✓ rocket-add lunar-mission
```

```
✓ rocket-start-power lunar-mission
```

```
✓ rocket-internal-power lunar-mission
```

```
✓ rocket-start-sequence lunar-mission
```

```
✓ rocket-start-engine lunar-mission
```

```
✓ rocket-lift-off lunar-mission
```

```
✓ rocket-status lunar-mission
```



# Run script as Command

```
$ bash create-and-launch-rocket.sh
```

```
$ create-and-launch-rocket      ✗  
create-and-launch-rocket      : command not found
```

```
$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
```

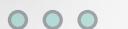
```
$ export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/home/michael
```

or

```
$ export PATH=$PATH:/home/michael
```

```
$ create-and-launch-rocket    ✓
```

```
$ which create-and-launch-rocket  
/home/michael/create-and-launch-rocket
```



# Executing a Script

```
$ /home/michael/ create-and-launch-rocket ✘  
-bash: ./create-and-launch-rocket: Permission denied
```

```
$ ls -l /home/michael/create-and-launch-rocket  
-rw-rw-r-- 1 michael michael 19 Mar 16 09:47 create-and-launch-rocket
```

```
$ chmod +x /home/michael/create-and-launch-rocket
```

```
$ ls -l /home/michael/create-and-launch-rocket  
-rwx-rwx-r-x 1 michael michael 19 Mar 16 09:47 create-and-launch-rocket
```

```
$ /home/michael/create-and-launch-rocket ✓
```



# Best Practice

*“Give your script a name  
that makes sense”*

*good:*

*create-and-launch-rocket*

*bad:*

*script1.sh*

*myscript.sh*

*test.sh*

*“Leave out .sh extension for  
executables”*

*good:*

*create-and-launch-rocket*

*bad:*

*create-and-launch-rocket.sh*

...

# HANDS-ON LABS



# Variables



Shell Scripting





# VARIABLES

create-and-launch-rocket

mkdir lunar-mission

rocket-add lunar-mission

rocket-start-power lunar-mission  
rocket-internal-power lunar-mission  
rocket-start-sequence lunar-mission  
rocket-start-engine lunar-mission  
rocket-lift-off lunar-mission

rocket-status lunar-mission

...

# VARIABLES

ALPHANUMERIC OR underscores

mission\_name mission-name

CASE SENSITIVE

MISSION\_NAME ≠ mission\_name

create-and-launch-rocket

mission\_name=mars-mission

mkdir \$mission\_name

rocket-add \$mission\_name

rocket-start-power \$mission\_name

rocket-internal-power \$mission name

rocket-crew-ready \$mission\_name

rocket-start-sequence \$mission name

rocket-start-engine \$mission name

rocket-lift-off \$mission\_name

rocket-status \$mission\_name





# VARIABLES

```
$ rocket-status lunar-mission  
launching      success      failed
```

```
create-and-launch-rocket  
mission_name=mars-mission  
mkdir $mission_name  
  
rocket-add $mission_name  
  
rocket-start-power $mission_name  
rocket-internal-power $mission name  
rocket-crew-ready $mission_name  
rocket-start-sequence $mission name  
rocket-start-engine $mission name  
rocket-lift-off $mission_name  
  
rocket_status=$(rocket-status $mission_name)  
echo "Status of launch: $rocket_status"
```



```
echo "Status of launch: $rocket_status_state"  
      "Status of launch: success_state"
```





```
echo "Status of launch: ${rocket_status}_state"  
      "Status of launch: success_state"
```



# Best Practice

*“Variable names must be in lower-case with underscores to separate words”*

*good:*

*mission\_name*

*bad:*

*Mission\_Name*

*Mission Name*

*Mission-name*

...

# HANDS-ON LABS



# Command Line Arguments



Shell Scripting

...

# Command Line Arguments

```
$ create-and-launch-rocket
```

```
$ #modify the create-and-Launch-rocket
```

```
$ create-and-launch-rocket
```

```
$ #modify the create-and-Launch-rocket
```

```
$ create-and-launch-rocket
```

```
$ create-and-launch-rocket saturn-mission
```

\$0

\$1

```
$ create-and-launch-rocket jupiter-mission
```

```
$ create-and-launch-rocket uranus-mission
```

```
create-and-launch-rocket
```

```
mission_name=$1
```

```
mkdir $mission_name
```

```
rocket-add $mission_name
```

```
rocket-start-power $mission_name
```

```
rocket-internal-power $mission_name
```

```
rocket-start-sequence $mission_name
```

```
rocket-start-engine $mission_name
```

```
rocket-lift-off $mission_name
```

```
rocket-status $mission_name
```

```
rocket_status=$(rocket-status $mission_name)
```

```
echo "Status of launch: $rocket_status"
```



# Command Line Arguments



```
create-and-launch-rocket  
mission_name= $1
```

```
mkdir $mission_name
```

```
rocket-add $mission_name
```

```
rocket-start-power $mission_name  
rocket-internal-power $mission_name  
rocket-start-sequence $mission_name  
rocket-start-engine $mission_name  
rocket-lift-off $mission_name
```

```
rocket-status $mission_name
```

```
rocket_status=$(rocket-status $mission_name)  
echo "Status of launch: $rocket_status"
```



```
create-and-launch-rocket
```

```
mkdir $1
```

```
rocket-add $1
```

```
rocket-start-power $1  
rocket-internal-power $1  
rocket-start-sequence $1  
rocket-start-engine $1  
rocket-lift-off $1
```

```
rocket-status $1
```

```
rocket_status=$(rocket-status $1)  
echo "Status of launch: $rocket_status"
```

# Best Practice

*“Design your script to be re-usable.”*

*“Script should not require to be edited before running.”*

*“Use command line arguments to pass inputs.”*

...

# Input

Shell Scripting





```
$ create-and-launch-rocket saturn-mission
```

```
$ create-and-launch-rocket  
Enter the mission name: saturn-mission
```

```
$ create-and-launch-rocket  
saturn-mission
```

```
create-and-launch-rocket
```

```
read mission_name
```

```
mkdir $mission_name
```

```
rocket-add $mission_name
```

```
rocket-start-power $mission_name
```

```
rocket-internal-power $mission_name
```

```
rocket-start-sequence $mission_name
```

```
rocket-start-engine $mission_name
```

```
rocket-lift-off $mission_name
```

```
rocket-status $mission_name
```

```
rocket_status=$(rocket-status $mission_name)  
echo "Status of launch: $rocket_status"
```





```
$ create-and-launch-rocket saturn-mission
```

```
$ create-and-launch-rocket  
Enter the mission name: saturn-mission
```

```
$ create-and-launch-rocket  
saturn-mission
```

```
$ create-and-launch-rocket  
Enter the mission name: saturn-mission
```

```
create-and-launch-rocket
```

```
read -p "Enter mission name:" mission_name
```

```
mkdir $mission_name
```

```
rocket-add $mission_name
```

```
rocket-start-power $mission_name
```

```
rocket-internal-power $mission_name
```

```
rocket-start-sequence $mission_name
```

```
rocket-start-engine $mission_name
```

```
rocket-lift-off $mission_name
```

```
rocket-status $mission_name
```

```
rocket_status=$(rocket-status $mission_name)
```

```
echo "Status of launch: $rocket_status"
```





```
create-and-launch-rocket
```

```
read -p "Enter mission name:" mission_name
```

```
mkdir $mission_name
```

```
rocket-add $mission_name
```

```
rocket-start-power $mission_name
```

```
rocket-internal-power $mission_name
```

```
rocket-start-sequence $mission_name
```

```
rocket-start-engine $mission_name
```

```
rocket-lift-off $mission_name
```

```
rocket-status $mission_name
```

```
rocket_status=$(rocket-status $mission_name)
```

```
echo "Status of launch: $rocket_status"
```

```
create-and-launch-rocket
```

```
mission_name= $1
```

```
mkdir $mission_name
```

```
rocket-add $mission_name
```

```
rocket-start-power $mission_name
```

```
rocket-internal-power $mission_name
```

```
rocket-start-sequence $mission_name
```

```
rocket-start-engine $mission_name
```

```
rocket-lift-off $mission_name
```

```
rocket-status $mission_name
```

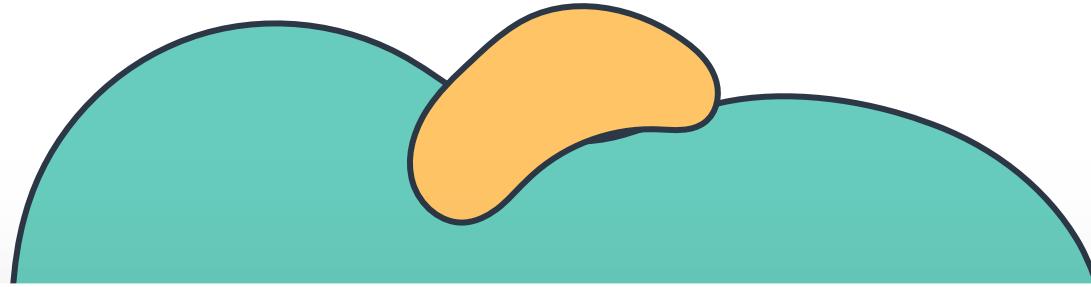
```
rocket_status=$(rocket-status $mission_name)
```

```
echo "Status of launch: $rocket_status"
```



# HANDS-ON LABS





# Arithmetic Operations



Shell Scripting



# expr

```
$ A=6
```

```
$ B=3
```

```
$ expr 6 + 3
```

```
9
```

```
$ expr $A + $B
```

```
9
```

```
$ expr 6 - 3
```

```
3
```

```
$ expr $A - $B
```

```
3
```

```
$ expr 6 / 3
```

```
2
```

```
$ expr $A / $B
```

```
2
```

```
$ expr 6 \* 3
```

```
18
```

```
$ expr $A \* $B
```

```
18
```

...



```
$ A=6  
$ B=3
```

```
$ expr $A + $B  
9
```

```
$ expr $A - $B  
3
```

```
$ expr $A / $B  
2
```

```
$ expr $A \* $B  
18
```

## double parentheses

```
$ echo $(( A + B ))  
9
```

```
$ echo $(( A-B ))  
3
```

```
$ echo $((A/B))  
2
```

```
$ echo $(( A * B ))  
18
```

...



# double parentheses

```
$ echo $(( A + B ))
```

```
9
```

```
$ echo $(( A-B ))
```

```
3
```

```
$ echo $((A/B))
```

```
2
```

```
$ echo $(( A * B ))
```

```
18
```

```
$ echo $(( ++A ))
```

```
7
```

```
$ echo $(( --A ))
```

```
6
```

```
$ echo $(( A++ ))
```

```
6
```

```
$ echo $(( A-- ))
```

```
7
```

...

**bc**

```
$ A=10  
$ B=3
```

```
$ expr $A / $B  
3
```

```
$ echo $((A/B))  
3
```

```
$ echo $A / $B | bc -l  
3.33333
```

...

# HANDS-ON LABS



# Conditional Logic

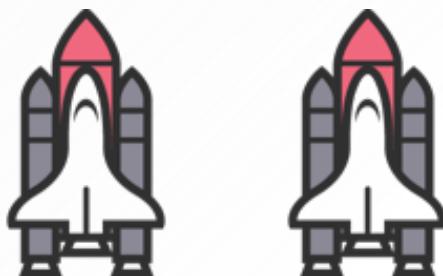


Shell Scripting

...

```
$ rocket-status lunar-mission  
launching [ ] success [ ] failed
```

```
$ rocket-debug lunar-mission  
overheating
```



```
create-and-launch-rocket
```

```
mission_name=$1
```

```
mkdir $mission_name
```

```
rocket-add $mission_name
```

```
rocket-start-power $mission_name
```

```
rocket-internal-power $mission_name
```

```
rocket-start-sequence $mission_name
```

```
rocket-start-engine $mission_name
```

```
rocket-lift-off $mission_name
```

```
rocket_status=$(rocket-status $mission_name)
```

```
if rocket-status is failed, then run this  
rocket-debug $mission_name
```



# Conditional Logic

```
$ rocket-status lunar-mission  
launching [ success ] failed
```

```
$ rocket-debug lunar-mission  
overheating
```

```
create-and-launch-rocket
```

```
mission_name=$1
```

```
mkdir $mission_name
```

```
rocket-add $mission_name
```

```
rocket-start-power $mission_name
```

```
rocket-internal-power $mission_name
```

```
rocket-start-sequence $mission_name
```

```
rocket-start-engine $mission_name
```

```
rocket-lift-off $mission_name
```

```
rocket_status=$(rocket-status $mission_name)
```

```
if rocket-status is failed, then run this  
if [ $rocket_status = "failed" ]  
then  
    rocket-debug $mission_name  
fi
```

# Conditional Logic

```
$ rocket-status lunar-mission  
launching [ success ] failed
```

```
$ rocket-debug lunar-mission  
overheating
```

```
create-and-launch-rocket  
● mission_name=$1  
  
● mkdir $mission_name  
  
● rocket-add $mission_name  
  
● rocket-start-power $mission_name  
● rocket-internal-power $mission_name  
● rocket-start-sequence $mission_name  
● rocket-start-engine $mission_name  
● rocket-lift-off $mission_name  
  
● rocket_status=$(rocket-status $mission_name)  
  
● if [ $rocket_status = "failed" ]  
then  
    ● rocket-debug $mission_name  
fi
```



# Else If

```
$ rocket-status lunar-mission  
launching [ ] success [ ] failed
```

```
$ rocket-debug lunar-mission  
overheating
```

create-and-launch-rocket

```
mission_name=$1  
  
mkdir $mission_name  
  
rocket-add $mission_name  
  
rocket-start-power $mission_name  
rocket-internal-power $mission_name  
rocket-start-sequence $mission_name  
rocket-start-engine $mission_name  
rocket-lift-off $mission_name  
  
rocket_status=$(rocket-status $mission_name)  
  
if [ $rocket_status = "failed" ]  
then  
    rocket-debug $mission_name  
elif [ $rocket_status = "success" ]  
then  
    echo "This is successful"  
fi
```

# Else

```
$ rocket-status lunar-mission  
launching [ success ] failed
```

```
$ rocket-debug lunar-mission  
overheating
```

```
mkair $mission_name  
rocket-add $mission_name  
rocket-start-power $mission_name  
rocket-internal-power $mission_name  
rocket-start-sequence $mission_name  
rocket-start-engine $mission_name  
rocket-lift-off $mission_name  
  
rocket_status=$(rocket-status $mission_name)  
  
if [ $rocket_status = "failed" ]  
then  
    rocket-debug $mission_name  
  
elif [ $rocket_status = "success" ]  
then  
    echo "This is successful"  
  
else  
  
    echo "The state is not failed or success"  
fi
```

# Conditional Operators

[ STRING1 ] = [ STRING2 ]

Example	Description
[ "abc" == "abc" ]	If string1 is exactly equal to string2 (true)
[ "abc" != "abc" ]	If string1 is not equal to string 2 (false)
[ 5 -eq 5 ]	If number1 is equal to number2 (true)
[ 5 -ne 5 ]	If number1 is not equal to number2 (false)
[ 6 -gt 5 ]	If number1 is greater than number2 (true)
[ 5 -lt 6 ]	If number1 is less than number2 (true)

# Conditional Operators

```
[[ STRING1 = STRING2 ]]
```

Example	Description
<code>[[ "abcd" = *bc* ]]</code>	If abcd contains bc (true)
<code>[[ "abc" = ab[cd] ]]</code> or <code>[[ "abd" = ab[cd] ]]</code>	If 3 <sup>rd</sup> character of abc is c or d (true)
<code>[[ "abe" = "ab[cd]" ]]</code>	If 3 <sup>rd</sup> character of abc is c or d (false)
<code>[[ "abc" &gt; "bcd" ]]</code>	If “abc” comes after “bcd” when sorted in alphabetical (lexographical) order (false)
<code>[[ "abc" &lt; "bcd" ]]</code>	If “abc” comes before “bcd” when sorted in alphabetical (lexographical) order (true)

Only in BASH

# Conditional Operators

[ COND1 ] && [ COND2 ]

[[ COND1 && COND2 ]]

[ COND1 ] || [ COND2 ]

[[ COND1 || COND2 ]]

Example	Description
[[ A -gt 4 && A -lt 10 ]]	If A is greater than 4 and less than 10
[[ A -gt 4    A -lt 10 ]]	If A is greater than 4 or less than 10



# Conditional Operators

Example	Description
[ -e FILE ]	if file exists
[ -d FILE ]	if file exists and is a directory
[ -s FILE ]	If file exists and has size greater than 0
[ -x FILE ]	If the file is executable
[ -w FILE ]	If the file is writeable

# HANDS-ON LABS

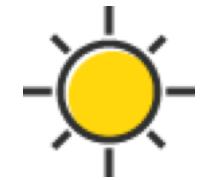
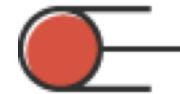
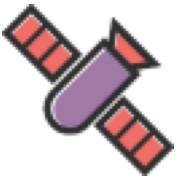


# Loops - For



Shell Scripting

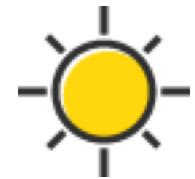
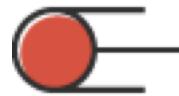
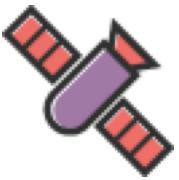




...



```
$ create-and-launch-rocket lunar-mission
```



```
$ create-and-launch-rocket lunar-mission
```

```
$ create-and-launch-rocket jupiter-mission
```

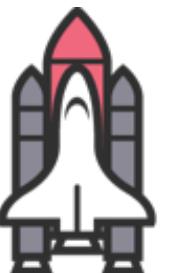
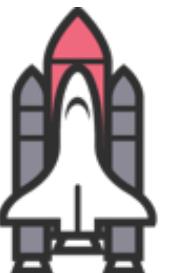
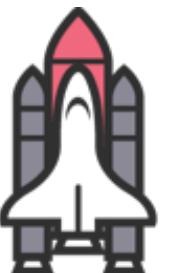
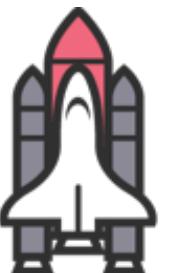
```
$ create-and-launch-rocket saturn-mission
```

```
$ create-and-launch-rocket satellite-mission
```

```
$ create-and-launch-rocket lunar-mission-2
```

```
$ create-and-launch-rocket mars-mission
```

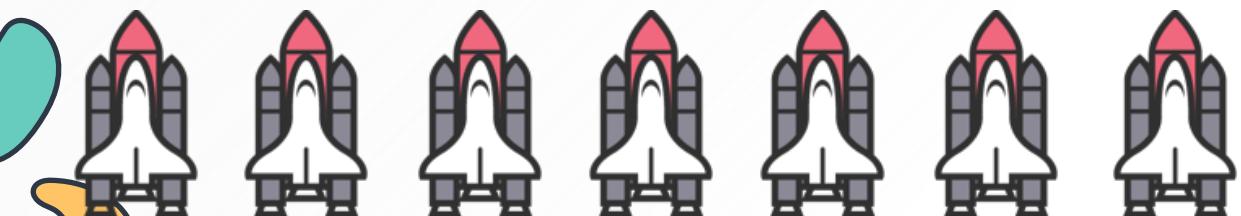
```
$ create-and-launch-rocket earth-mission
```



...



```
$ create-and-launch-rocket lunar-mission  
$ create-and-launch-rocket jupiter-mission  
$ create-and-launch-rocket saturn-mission  
$ create-and-launch-rocket satellite-mission  
$ create-and-launch-rocket lunar-mission-2  
$ create-and-launch-rocket mars-mission  
$ create-and-launch-rocket earth-mission
```



## launch-rocket.sh

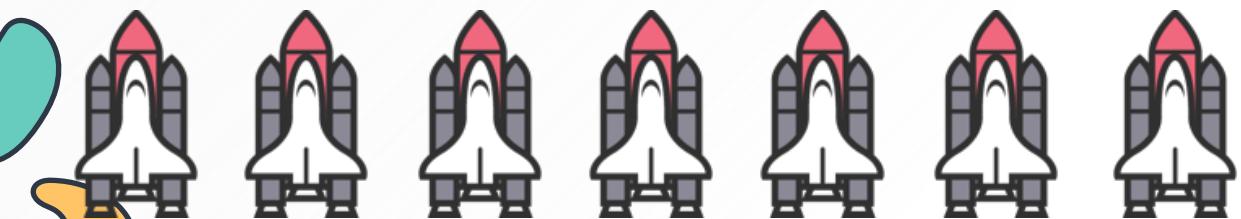
```
create-and-launch-rocket lunar-mission  
create-and-launch-rocket jupiter-mission  
create-and-launch-rocket saturn-mission  
create-and-launch-rocket satellite-mission  
create-and-launch-rocket lunar-mission-2  
create-and-launch-rocket mars-mission  
create-and-launch-rocket earth-mission
```

...



## launch-rocket.sh

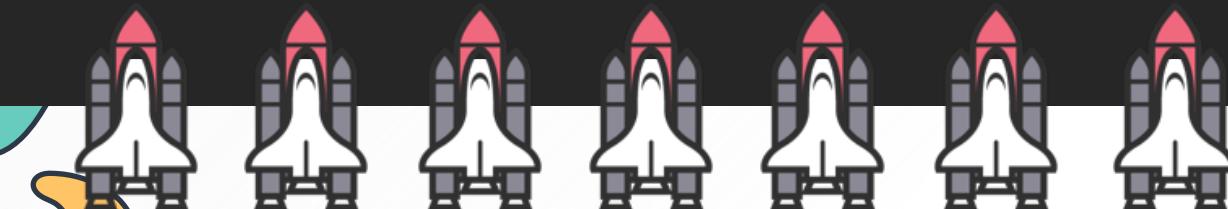
```
5 For each mission create and launch rocket  
for mission in lunar-mission jupiter-mission  
do  
    create-and-launch-rocket $mission  
done
```





launch-rockets.sh

```
❷ for mission in [lunar-mission][jupiter-mission][saturn-mission][satellite-mission][lunar-mission-2]
do
❸ create-and-launch-rocket $mission { earth-mission
done
```





## mission-names.txt

```
lunar-mission
jupiter-mission
saturn-mission
satellite-mission
lunar-mission-2
mars-mission
apollo-mission
spitzer-mission
viking-mission
pheonix-mission
chandrayan-mission
gaganyaan-mission
aditya-mission
nisar-mission
mangalyaan-mission
columbia-mission
challenger-mission
atlantis-mission
endeavour-mission
mercury-mission
gemini-mission
space-mission
```

## launch-rockets.sh

```
for mission in $(cat mission-names.txt)
do
    create-and-launch-rocket $mission
done
```

```
for mission in $(cat mission-names.txt)
do
    create-and-launch-rocket $mission
done
```

```
for mission in 1 2 3 4 5 6
do
    create-and-launch-rocket mission-$mission
done
```

```
for mission in {0..100}
do
    create-and-launch-rocket mission-$mission
done
```

mission-1  
mission-2  
mission-3  
mission-4  
mission-5  
mission-6

mission-1  
mission-2  
mission-3  
mission-4

mission-100

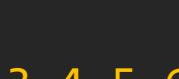
...



```
for mission in $(cat mission-names.txt)
do
  create-and-launch-rocket $mission
done
```



```
for (( mission = 0 ; mission <= 100; mission++ ))
do
  create-and-launch-rocket mission-$mission
done
```



```
for mission in 1 2 3 4 5 6
do
  create-and-launch-rocket mission-$mission
done
```



```
for mission in {0..100}
do
  create-and-launch-rocket mission-$mission
done
```

...



# Use a **For Loop** when you have to:

- Execute a command or a set of commands many times
- Iterate through files
- Iterate through lines within a file
- Iterate through the output of a command

...

# Real life use cases:

```
for file in $(ls)
do
    echo Line count of $file is $(cat $file | wc -l)
done
```

```
for server in $(cat servers.txt)
do
    ssh $server "uptime"
done
```

```
for package in $(cat install-packages.txt)
do
    sudo apt-get -y install $package
done
```

# HANDS-ON LABS



# Loops - While



Shell Scripting

...

```
$ rocket-status lunar-mission  
success
```



```
create-and-launch-rocket  
mission_name=$1  
  
mkdir $mission_name  
  
rocket-add $mission_name  
  
rocket-start-power $mission_name  
rocket-internal-power $mission_name  
rocket-start-sequence $mission_name  
rocket-start-engine $mission_name  
rocket-lift-off $mission_name  
rocket_status=rocket-status $mission_name  
  
if [ $rocket_status = "failed" ]  
then  
    rocket-debug $mission_name  
fi
```

...

```
$ rocket-status lunar-mission  
launching
```

```
create-and-launch-rocket
```

```
mission_name=$1
```

```
mkdir $mission_name
```

```
rocket-add $mission_name
```

```
rocket-start-power $mission_name
```

```
rocket-internal-power $mission_name
```

```
rocket-start-sequence $mission_name
```

```
rocket-start-engine $mission_name
```

```
rocket-lift-off $mission_name
```

```
rocket_status=rocket-status $mission_name
```

```
if [$rocket_status = "launching"]
```

```
then
```

```
sleep 2
```

```
rocket_status=rocket-status $mission_name
```

```
fi
```

```
if [ $rocket_status = "failed" ]
```

```
then
```

```
rocket-debug $mission_name
```

```
fi
```

```
$ rocket-status lunar-mission  
launching
```

```
create-and-launch-rocket  
mission_name=$1  
  
mkdir $mission_name  
  
rocket-add $mission_name  
  
rocket-start-power $mission_name  
rocket-internal-power $mission_name  
rocket-start-sequence $mission_name  
rocket-start-engine $mission_name  
rocket-lift-off $mission_name  
  
rocket_status=rocket-status $mission_name  
if [$rocket_status = "launching"]  
then  
    sleep 2  
    rocket_status=rocket-status $mission_name  
    if [$rocket_status = "launching"]  
    then  
        sleep 2  
    fi  
fi
```

```
mkdir $mission_name
rocket-add $mission_name
rocket-start-power $mission_name
rocket-internal-power $mission_name
rocket-start-sequence $mission_name
rocket-start-engine $mission_name
rocket-lift-off $mission_name
rocket_status=rocket-status $mission_name
if [$rocket_status = "launching"]
then
    sleep 2
    rocket_status=rocket-status $mission_name
    if [$rocket_status = "launching"]
    then
        sleep 2
        rocket_status=rocket-status $mission_name
        if [$rocket_status = "launching"]
        then
            sleep 2
        fi
    fi
fi
```

```
rocket-start-engine $mission_name  
rocket-lift-off $mission_name  
rocket_status=rocket-status $mission_name
```

```
while [ $rocket_status = "launching" ]  
do
```

```
sleep 2  
rocket_status=rocket-status $mission_name
```

```
done
```



rocket\_status=failed

```
create-and-launch-rocket
● mission_name=$1
● mkdir $mission_name
● rocket-add $mission_name
● rocket-start-power $mission_name
● rocket-internal-power $mission_name
● rocket-start-sequence $mission_name
● rocket-start-engine $mission_name
● rocket-lift-off $mission_name
● rocket_status=rocket-status $mission_name
? while [ $rocket_status = "launching" ]
do
  ● sleep 2
  ● rocket_status=rocket-status $mission_name
done
? if [ $rocket_status = "failed" ]
then
  ● rocket-debug $mission_name
fi
```



# Use a **While Loop** when you have to:

- Execute a command or a set of commands multiple times but you are not sure how many times.
- Execute a command or a set of commands until a specific condition occurs
- Create infinite loops
- Menu driven programs





# Real life use cases:

```
while true
Do
    echo "1. Shutdown"
    echo "2. Restart"
    echo "3. Exit Menu"
    read -p "Enter your choice: " choice

    if [ $choice -eq 1 ]
    then
        shutdown now
    elif [ $choice -eq 2 ]
    then
        shutdown -r now
    elif [ $choice -eq 3 ]
    then
        break
    else
        continue
    fi
```

done



# HANDS-ON LABS



# Case Statements

Shell Scripting

...



```
while true
do
    echo "1. Shutdown"
    echo "2. Restart"
    echo "3. Exit Menu"
    read -p "Enter your choice: " choice

    if [ $choice -eq 1 ]
    then
        shutdown now
    elif [ $choice -eq 2 ]
    then
        shutdown -r now
    elif [ $choice -eq 3 ]
    then
        break
    else
        continue
    fi
done
```





```
echo "1. Shutdown"
echo "2. Restart"
echo "3. Exit Menu"
read -p "Enter your choice: " choice

if [ $choice -eq 1 ]
then
    shutdown now
elif [ $choice -eq 2 ]
then
    shutdown -r now
elif [ $choice -eq 3 ]
then
    break
else
    continue
fi
```





# Case Statement

```
echo "1. Shutdown"
echo "2. Restart"
echo "3. Exit Menu"
read -p "Enter your choice: " choice

if [ $choice -eq 1 ]
then
    shutdown now
elif [ $choice -eq 2 ]
then
    shutdown -r now
elif [ $choice -eq 3 ]
then
    break
else
    continue
fi
```

```
echo "1. Shutdown"
echo "2. Restart"
echo "3. Exit Menu"
read -p "Enter your choice: " choice

case $choice in
    1) shutdown now ;;
    2) shutdown -r now ;;
    3) break ;;
    *) continue ;;
esac
```



# Case Statement

```
echo "1. Shutdown"
echo "2. Restart"
echo "3. Exit Menu"
read -p "Enter your choice: " choice

if [ $choice -eq 1 ]
then
    shutdown now
elif [ $choice -eq 2 ]
then
    shutdown -r now
elif [ $choice -eq 3 ]
then
    break
else
    continue
fi
```

```
echo "1. Shutdown"
echo "2. Restart"
echo "3. Exit Menu"
read -p "Enter your choice: " choice

case $choice in
    1) shutdown now
       ;;
    2) shutdown -r now
       ;;
    3) break
       ;;
    *) continue
       ;;

esac
```



# Case Statement

```
echo "1. Shutdown"
echo "2. Restart"
echo "3. Exit Menu"
read -p "Enter your choice: " choice

case $choice in
    1) shutdown now
       ;;
    2) shutdown -r now
       ;;
    3) break
       ;;
    *) continue
       ;;

esac
```



# Case Statement

```
while true
do
    echo "1. Shutdown"
    echo "2. Restart"
    echo "3. Exit Menu"
    read -p "Enter your choice: " choice

    case $choice in
        1) shutdown now
            ;;
        2) shutdown -r now
            ;;
        3) break
            ;;
        *) continue
            ;;

    esac
done
```



# HANDS-ON LABS



# SHEBANG

Shell Scripting



# SHEBANG

```
sh$ ls -l /bin/sh  
/bin/sh -> /bin/bash
```

launch-rockets.sh

```
#!/bin/bash  
for mission in {0..10}  
do  
    create-and-launch-rocket $mission  
done
```

Bourne Shell (sh)  
Debian Almquist Shell (dash)

```
sh$ launch-rockets.sh  
Launching mission {0..10}
```

```
sh$ bash launch-rockets.sh  
Launching mission 0  
Launching mission 1  
Launching mission 2  
Launching mission 3  
. .  
Launching mission 9  
Launching mission 10
```

Bourne again Shell (bash)

```
bash$ launch-rockets.sh  
Launching mission 0  
Launching mission 1  
Launching mission 2  
Launching mission 3  
Launching mission 4  
Launching mission 5  
Launching mission 6  
Launching mission 7  
Launching mission 8  
Launching mission 9  
Launching mission 10
```

# Best Practice

*“Always start with a Shebang in your scripts”*



# Exit Codes



Shell Scripting



# Exit Codes

```
$ ls  
/home /root /tmp
```

EXIT STATUS = 0

SUCCESS

```
$ echo $?  
0
```

```
$ lss  
Failed: command not found
```

EXIT STATUS > 0

FAILURE

```
$ echo $?  
127
```

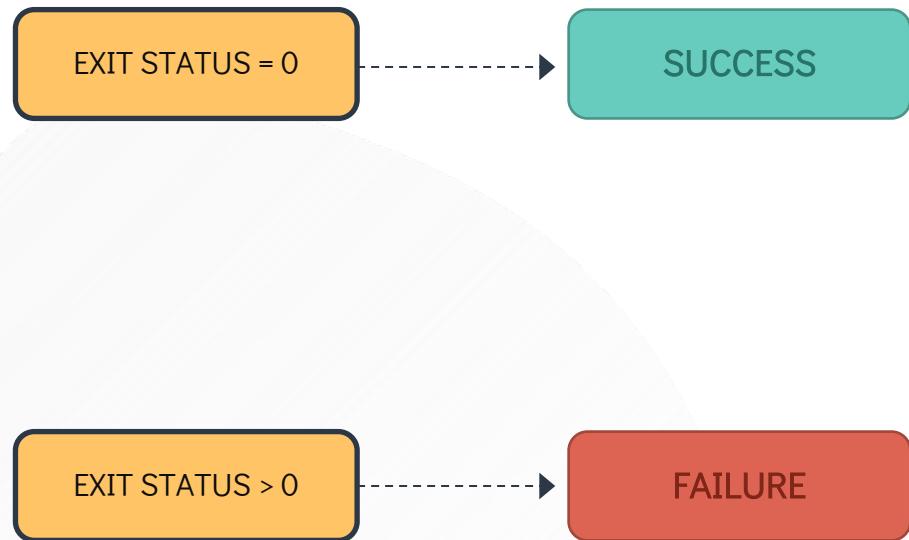
```
$ rocket-status  
success
```

```
$ echo $?  
0
```

```
$ rocket-status  
failed
```

```
$ echo $?  
1
```





```
$ create-and-launch-rocket  
failed
```

```
$ echo $?  
0
```

```
create-and-launch-rocket
```

```
mission_name=$1
```

```
mkdir $mission_name
```

```
rocket-add $mission_name
```

```
rocket-start-power $mission_name
```

```
rocket-internal-power $mission_name
```

```
rocket-start-sequence $mission_name
```

```
rocket-start-engine $mission_name
```

```
rocket-lift-off $mission_name
```

```
rocket_status=rocket-status $mission_name
```

```
while [ $rocket_status == "launching" ]  
do
```

```
    sleep 2
```

```
    rocket_status=rocket-status $mission_name
```

```
done
```

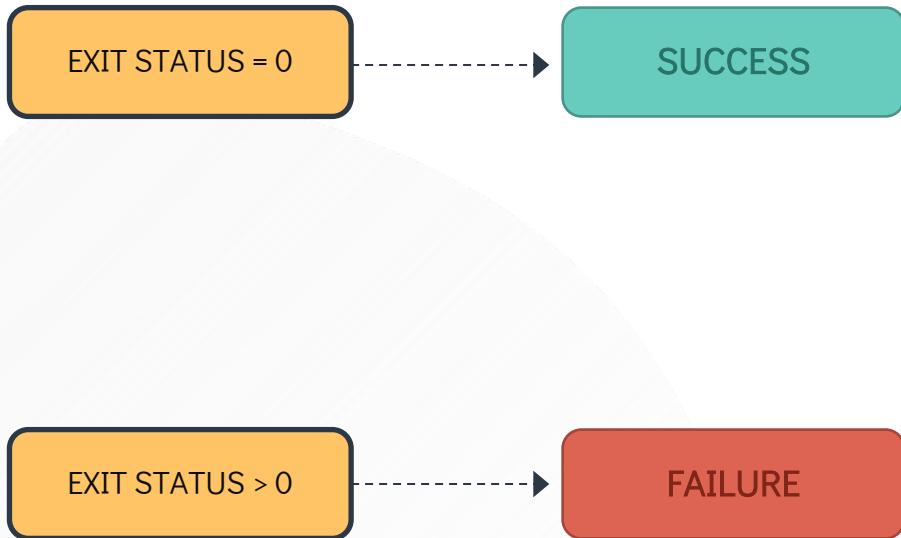
```
if [ $rocket_status = "failed" ]
```

```
then
```

```
    rocket-debug $mission_name
```

```
fi
```

## create-and-launch-rocket



```
$ create-and-launch-rocket  
failed
```

```
$ echo $?  
1
```

```
mission_name=$1  
  
mkdir $mission_name  
  
rocket-add $mission_name  
  
rocket-start-power $mission_name  
rocket-internal-power $mission_name  
rocket-start-sequence $mission_name  
rocket-start-engine $mission_name  
rocket-lift-off $mission_name  
  
rocket_status=rocket-status $mission_name  
while [ $rocket_status == "launching" ]  
do  
    sleep 2  
    rocket_status=rocket-status $mission_name  
done  
if [ $rocket_status = "failed" ]  
then  
    rocket-debug $mission_name  
    exit 1  
fi
```

# Best Practice

*“Always return appropriate exit codes in your script”*



# Functions



Shell Scripting



```
mission_name=$1

mkdir $mission_name

rocket-add $mission_name

rocket-start-power $mission_name
rocket-internal-power $mission_name
rocket-start-sequence $mission_name
rocket-start-engine $mission_name
rocket-lift-off $mission_name

rocket_status=$(rocket-status $mission_name)

while [ $rocket_status == "launching" ]
do
    sleep 2
    rocket_status=$(rocket-status $mission_name)
done
if [ $rocket_status = "failed" ]
then
    rocket-debug $mission_name
    exit 1
fi
```

## create-and-launch-rocket

```
mission_name=$1

mkdir $mission_name

rocket-add $mission_name

rocket-start-power $mission_name
rocket-internal-power $mission_name
rocket-start-sequence $mission_name
rocket-start-engine $mission_name
rocket-lift-off $mission_name

rocket_status=$(rocket-status $mission_name)

while [ $rocket_status == "launching" ]
do
    sleep 2
    rocket_status=$(rocket-status $mission_name)
done
if [$rocket_status = "failed" ]
then
    rocket-debug $mission_name
    exit 1
fi

mission_name=mars-mission

mkdir $mission_name

rocket-add $mission_name

rocket-start-power $mission_name
rocket-internal-power $mission_name
rocket-start-sequence $mission_name
rocket-start-engine $mission_name
rocket-lift-off $mission_name

rocket_status=$(rocket-status $mission_name)

while [ $rocket_status == "launching" ]
do
```

## create-and-launch-rocket

```
mission_name=$1
mkdir $mission_name
rocket-add $mission_name
rocket-start-power $mission_name
rocket-internal-power $mission_name
rocket-start-sequence $mission_name
rocket-start-engine $mission_name
rocket-lift-off $mission_name
rocket_status=$(rocket-status $mission_name)
while [ $rocket_status == "launching" ]
do
  sleep 2
  rocket_status=$(rocket-status $mission_name)
done
if [ $rocket_status = "failed" ]
then
  rocket-debug $mission_name
  exit 1
fi
```

```
mission_name=mars-mission
mkdir $mission_name
rocket-add $mission_name
rocket-start-power $mission_name
rocket-internal-power $mission_name
rocket-start-sequence $mission_name
rocket-start-engine $mission_name
rocket-lift-off $mission_name
rocket_status=$(rocket-status $mission_name)
while [ $rocket_status == "launching" ]
do
  sleep 2
  rocket_status=$(rocket-status $mission_name)
done
if [ $rocket_status = "failed" ]
then
  rocket-debug $mission_name
  exit 1
fi
```

```
mission_name=mars-mission
mkdir $mission_name
rocket-add $mission_name
rocket-start-power $mission_name
rocket-internal-power $mission_name
rocket-start-sequence $mission_name
```

```
mission_name=$1

mkdir $mission_name

rocket-add $mission_name

rocket-start-power $mission_name
rocket-internal-power $mission_name
rocket-start-sequence $mission_name
rocket-start-engine $mission_name
rocket-lift-off $mission_name

rocket_status=$(rocket-status $mission_name)

while [ $rocket_status == "launching" ]
do
  sleep 2
  rocket_status=$(rocket-status $mission_name)
done
if [$rocket_status = "failed" ]
then
  rocket-debug $mission_name
  exit 1
fi
```

```
function launch-rocket() {
    mission_name=$1

    mkdir $mission_name

    rocket-add $mission_name

    rocket-start-power $mission_name
    rocket-internal-power $mission_name
    rocket-start-sequence $mission_name
    rocket-start-engine $mission_name
    rocket-lift-off $mission_name

    rocket_status=$(rocket-status $mission_name)

    while [ $rocket_status == "launching" ]
    do
        sleep 2
        rocket_status=$(rocket-status $mission_name)
    done
    if [ $rocket_status = "failed" ]
    then
        rocket-debug $mission_name
        exit 1
    fi
}
```

```
function launch-rocket() {
    mission_name=$1
    mkdir $mission_name
    rocket-add $mission_name
    rocket-start-power $mission_name
    rocket-internal-power $mission_name
    rocket-start-sequence $mission_name
    rocket-start-engine $mission_name
    rocket-lift-off $mission_name
    rocket_status=$(rocket-status $mission_name)
    while [ $rocket_status == "launching" ]
    do
        sleep 2
        rocket_status=$(rocket-status $mission_name)
    done
    if [ $rocket_status = "failed" ]
    then
        rocket-debug $mission_name
        exit 1
    fi
}
launch-rocket lunar-mission
```

```
function launch-rocket() {
    mission_name=$1

    mkdir $mission_name

    rocket-add $mission_name

    rocket-start-power $mission_name
    rocket-internal-power $mission_name
    rocket-start-sequence $mission_name
    rocket-start-engine $mission_name
    rocket-lift-off $mission_name

    rocket_status=$(rocket-status $mission_name)

    while [ $rocket_status == "launching" ]
    do
        sleep 2
        rocket_status=$(rocket-status $mission_name)
    done
    if [ $rocket_status = "failed" ]
    then
        rocket-debug $mission_name
        exit 1
    fi
}
```

## Function Definition

```
launch-rocket lunar-mission
launch-rocket mars-mission
launch-rocket saturn-mission
launch-rocket mercury-mission
```

## Main

```
launch-rocket: command not found
```

## create-and-launch-rocket

```
launch-rocket lunar-mission  
launch-rocket mars-mission  
launch-rocket saturn-mission  
launch-rocket mercury-mission  
  
function launch-rocket()  
mission_name=$1  
  
mkdir $mission_name  
  
rocket-add $mission_name  
  
rocket-start-power $mission_name  
rocket-internal-power $mission_name  
rocket-start-sequence $mission_name  
rocket-start-engine $mission_name  
rocket-lift-off $mission_name  
  
rocket_status=$(rocket-status $mission_name)  
  
while [ $rocket_status == "launching" ]  
do  
    sleep 2  
    rocket_status=$(rocket-status $mission_name)  
done  
if [ $rocket_status = "failed" ]  
then  
    rocket-debug $mission_name  
    exit 1  
fi  
}
```

Main

Function Definition

```
function launch-rocket() {
    mission_name=$1

    mkdir $mission_name

    rocket-add $mission_name

    rocket-start-power $mission_name
    rocket-internal-power $mission_name
    rocket-start-sequence $mission_name
    rocket-start-engine $mission_name
    rocket-lift-off $mission_name

    rocket_status=$(rocket-status $mission_name)

    while [ $rocket_status == "launching" ]
    do
        sleep 2
        rocket_status=$(rocket-status $mission_name)
    done
    if [ $rocket_status = "failed" ]
    then
        rocket-debug $mission_name
        exit 1
    fi
}
```

## Function Definition

```
launch-rocket lunar-mission
launch-rocket mars-mission
launch-rocket saturn-mission
launch-rocket mercury-mission
```

## Main

## create-and-launch-rocket

```
function launch-rocket() {
    mission_name=$1

    mkdir $mission_name

    rocket-add $mission_name

    rocket-start-power $mission_name
    rocket-internal-power $mission_name
    rocket-start-sequence $mission_name
    rocket-start-engine $mission_name
    rocket-lift-off $mission_name

    rocket_status=$(rocket-status $mission_name)

    while [ $rocket_status == "launching" ]
    do
        sleep 2
        rocket_status=$(rocket-status $mission_name)
    done
    if [ $rocket_status = "failed" ]
    then
        rocket-debug $mission_name
        return 1
    fi
}

launch-rocket lunar-mission
launch-rocket mars-mission
launch-rocket saturn-mission
launch-rocket mercury-mission
```

## create-and-launch-rocket

```
function launch-rocket() {
    mission_name=$1

    mkdir $mission_name

    rocket-add $mission_name

    rocket-start-power $mission_name
    rocket-internal-power $mission_name
    rocket-start-sequence $mission_name
    rocket-start-engine $mission_name
    rocket-lift-off $mission_name

    rocket_status=$(rocket-status $mission_name)

    while [ $rocket_status == "launching" ]
    do
        sleep 2
        rocket_status=$(rocket-status $mission_name)
    done
    if [ $rocket_status = "failed" ]
    then
        rocket-debug $mission_name
        return 1
    fi
}

launch-rocket lunar-mission
LUNAR_STATUS_CODE=?

launch-rocket mars-mission
MARS_STATUS_CODE=?

launch-rocket saturn-mission
SATURN_STATUS_CODE=?

launch-rocket mercury-mission
MERCURY_STATUS_CODE=?
```



## When to use Functions?

- Break up large script that performs many different tasks:
  - Installing packages
  - Adding users
  - Configuring firewalls
  - Perform Mathematical calculations

...



```
function add(){  
    echo $(( $1 + $2 ))  
}
```

```
add 3 5
```





```
function add(){
    echo $(( $1 + $2 ))
}
```

```
sum=$( add 3 5 )
```



```
function add(){  
    echo $(( $1 + $2 ))  
}
```

sum=\$( add 3 5 )

```
function add(){  
    return $(( $1 + $2 ))  
}
```

add 3 5

sum=\$?



# Best Practice

*“Always develop scripts in a modular re-usable way using functions”*

*“Avoid duplicate code”*

*“Use arguments/parameters to pass in variables”*

...

# HANDS-ON LABS



# Project



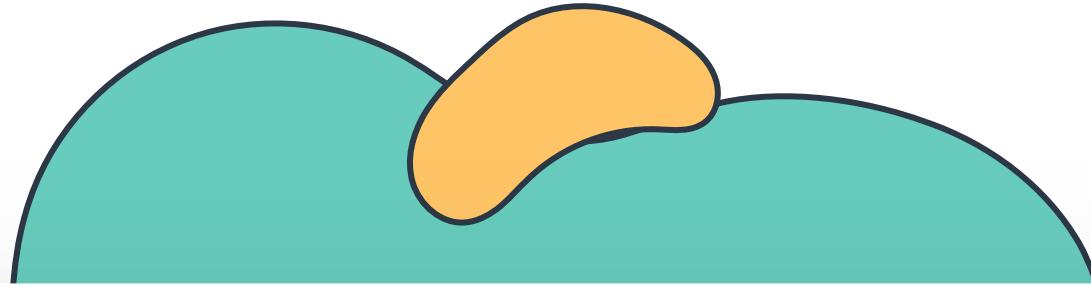
Shell Scripting

...



<https://github.com/kodekloudhub/learning-app-ecommerce>





# ShellCheck/IDEs



Shell Scripting



# VIM Editor

```
#!/bin/bash

while true
do
    echo "1. Shutdown"
    echo "2. Restart"
    echo "3. Exit Menu"

    read -p "Enter your choice: " choice

    if [ $choice -eq 1 ]
    then
        echo "shutdown now"
    elif [ $choice -eq 2 ]
    then
        echo "shudown -r now"
    elif [ $choice -eq 3 ]
    then
        break
    fi
done
```

"menu.sh" [noeol] 21L, 294C





# SHELLCHECK

```
$ apt-get install shellcheck
```

Or

```
$ yum install shellcheck
```

```
$ shellcheck menu.sh
```

In **menu.sh** line 9:

```
read -p "Enter your choice: " choice  
^--^ SC2162: read without -r will mangle backslashes.
```

In **menu.sh** line 11:

```
if [ $choice -eq 1 ]  
^-----^ SC2086: Double quote to prevent globbing and word splitting.
```

Did you mean:

```
if [ "$choice" -eq 1 ]
```

In **menu.sh** line 14:

```
elif [ $choice -eq 2 ]  
^-----^ SC2086: Double quote to prevent globbing and word splitting
```

Did you mean:

```
elif [ "$choice" -eq 2 ]
```



Version: 2020.1.2

Build: 201.7846.77

3 June 2020

[System requirements](#)[Installation Instructions](#)[Other versions](#)

## Download PyCharm

[Windows](#) [Mac](#) [Linux](#)

### Professional

For both Scientific and Web Python development. With HTML, JS, and SQL support.

[Download](#)[Free trial](#)

### Community

For pure Python development

[Download](#)[Free, open-source](#)

<https://www.jetbrains.com/pycharm/download>

```
#!/bin/bash
while true
do
    echo "1. Shutdown"
    echo "2. Restart"
    echo "3. Exit Menu"
    read -p "Enter your choice: " choice
    if [ $choice -eq 1 ]
    then
        echo "shutdown"
    elif [ $choice -eq 2 ]
    then
        echo "shutdown -r now"
    elif [ $choice -eq 3 ]
    then
        break
    fi
done
```

read without -r will mangle backslashes.  
See SC2162.

read -p "Enter your choice: " choice

Suppress 'read without -r will mangle backslashes' [YOrN] More actions... [Cntrl+C]



The screenshot shows the Visual Studio Code website interface. At the top, there are icons for the VS Code logo, Windows, macOS, and Linux. Below this, the title "Visual Studio Code" is displayed. A sidebar on the left contains icons for search, extensions, snippets, and other tools. The main area shows the "EXTENSIONS" tab of the marketplace. Several extensions are listed, including "C# (1.2.1)", "Python (3.1.1)", "Debugger for Chrome (0.14.0)", and "G/C++ (0.2.1)". Each entry includes a star rating and an "Install" button. To the right of the extensions, a code editor window displays some Python code. Below the extension list, the text "Editing and debugging on any OS" is followed by a note: "(By using Visual Studio Code you agree to its license and privacy statement)". A large purple "Download Visual Studio Code" button is at the bottom, along with a "Learn more >" link.

The screenshot shows the Atom website. At the top, there is a large circular graphic featuring the Atom logo in the center, surrounded by concentric rings of color. Below this, a card displays the "ATOM" logo, the version "1.48.0", and a link to "Release notes". Further down, the word "macOS" is shown with the note "For macOS 10.9 or later". At the bottom of the card is a yellow "Download" button with a downward arrow icon.





# styleguide

## Shell Style Guide

Revision 2.02

Authored, revised and maintained by many Googlers.

### Table of Contents

Section	Contents
Background	Which Shell to Use - When to use Shell
Shell Files and Interpreter Invocation	File Extensions - SUID/SGID
Environment	STDOUT vs STDERR
Comments	File Header - Function Comments - Implementation Comments - TODO Comments
Formatting	Indentation - Line Length and Long Strings - Pipelines - Loops - Case statement - Variable expansion - Quoting
Features and Bugs	ShellCheck - Command Substitution - Test, [...] , and {{...}} - Testing Strings - Wildcard Expansion of Filenames - Eval - Arrays - Pipes to While - Arithmetic
Naming Conventions	Function Names - Variable Names - Constants and Environment Variable Names - Source Filenames - Read-only Variables - Use Local Variables - Function Location - main
Calling Commands	Checking Return Values - Builtin Commands vs. External Commands
Conclusion	

<https://google.github.io/styleguide/shellguide.html>





CLOUD



# Thank You!



@vijinpalazhi

@mmumshad



KODE{KLOUD