Bash Scripts

Basics

```
# create a file with .sh extension
> vim hello_world.sh

# add a echo statement
# echo "hello world"

# execute shell script using bash
> bash hello_world.sh

# execute using shebang header
# note: add shebang header on top of the file
# #! /usr/bin/bash

# change the permissions
> chmod +x hello_world.sh

# execute the file
> ./hello_world.sh
```

string quotes

```
# print hello world
echo "hello world"
echo 'hello world'
echo hello world
```

variables

```
# numeric variable
num=10

# decimal variable
salary=10.50

# string variable
first_name=amit
```

```
last_name="Kulkarni"
company='sunbeam'
address=pune

# print the values
echo $first_name
echo "last name is: $last_name"
echo "company is" $company
echo "address is ${address}"
```

special variables

```
echo "the exit status of last task: $?"
echo "PID of last background task: $!"
echo "PID of shell: $$"
echo "file name of the shell script: $0"
echo "last argument of previous command: $_"
```

command execution

```
# execute a command and print the output
echo "todays date is $(date)"

# declare a variable
command="pwd"
echo "current working directory is: $(pwd)"
```

expressions

```
# variables
num1=10
num2=20

# expressions
addition=$(expr $num1 + $num2)
subtraction=`echo "$num1 - $num2" | bc`
division=`echo "scale=2; $num1 / $num2" | bc`
multiplication1=$(expr $num1 \* $num2)
multiplication2=`echo "$num1 * $num2" | bc`
```

```
echo "addition = $addition"
echo "subtraction = $subtraction"
echo "division = $division"
echo "multiplication1 = $multiplication1"
echo "multiplication2 = $multiplication2"
```

array

```


# declare an empty array
array=()
echo $array

# declare array of numbers
numbers=(10 20 30 40 50)

# get all members
echo "all values = ${numbers[@]}"

# get size of array
echo "size of numbers = ${#numbers[@]}"

# get string length of nth ele,ent
echo "size of numbers = ${#numbers[n]}"


# append a value to the array
numbers+=(60)

# get the first element
echo "first element = ${numbers[0]}"

# get the last element
echo "last element = ${numbers[-1]}"



# declare array of string values
names=("steve" "bill" "jeff")
echo $names
```

string operations

```


name="amit"
echo "$name"
```

```
# substitute a with A
echo "${name/a/A}"

# slicing from 0 to 2
echo "${name:0:2}"

# slicing from 0 (default)
echo "${name::2}"

# slicing from right
echo "${name:(-2)}"

# slicing from right only 1 character
echo "${name:(-2):1}"

# convert the first character to lowercase
echo "${name,}"
```

bracket expansion

```
echo {1..10}
echo {a..z}
echo 1.{0..9}
echo ---{A..E}---
echo {{1..3},{4..6},{7..9}}
```

conditions

```
# check conditions
# -eq: equal
# -ne: not equal
# -gt: greater than
# -lt: less than
# -z: empty string

# if..then condition
num=10
if [ $(expr $num % 2) -eq 0 ]
then
    echo "this is even number"
fi

# if..else condition
if [ $(expr $num % 2) -eq 0 ]
then
```

```
    echo "this is even number"
else
    echo "this is odd number"
fi

# switch
operation='A'
case $value in
    A)
        echo "addition"
        ;;
    S)
        echo "Subtraction"
        ;;
    *)
        echo "Usage: $0 {start|stop|ssh}"
        ;;
esac
```

file conditions

```
# -e: file exists
# -r: file readable
# -h: file is symlink
# -d: directory
# -w: file is writable
# -s: file size is > 0
# -f: is file
# -x: is executable
# -nt: if 1st is more recent than second
# -ot: if 2nd is more recent than first
# -ef: whether both the files are same

# file exists
if [ -e $file ]
then
    echo "$file exists"
else
    echo "file does not exist"
fi
```

loops

```
# basic for loop
numbers=(1 2 3 4 5)
```

```bash
for i in ${numbers[@]}
do
  echo "$i"
done

# traditional for loop
for ((i = 0 ; i < 100 ; i++))
do
  echo "$i"
done

# while loop
while true
do
  # code here
done
```

functions

```bash
#  function declaration
function test_function {
    echo "inside test_function"
}

# function call
test_function

# read parameters inside the function
function add_two_numbers {
    result=$(expr $1 + $2)
    echo "result = $result"
}

# function call with parameters
add_two_numbers 10 20

# function returning value
function multiply_two_numbers {
    result=$(expr $1 \* $2)
    echo $result
}

# capture returned value
multiplication=$(multiply_two_numbers 10 20)
```