

# Comaparision\_Shell\_Scripting\_Python\_Scripting

## Comments

### Shell Scripting

```
: ' This is a multi-line comment '  
(or)  
<<COMMENT  
This is a multi-line comment.  
It spans multiple lines.  
COMMENT
```

### Python Scripting

```
''' This is a multi-line comment '''  
(or)  
"""  
This is a multi-line comment.  
It spans multiple lines.  
"""
```

## Variables

### String

### Shell Scripting

```
name="abc"  
filename="$name xyz"  
echo ${#name}  
echo ${name:0:1}  
echo ${name:0:3}  
echo ${name/ab/ef}  
echo "${name^}"  
echo "${name^^}"  
echo "${name,,}"  
[[ $name == *"abc"* ]]
```

### Python Scripting

```
name="abc"  
full_name = name + " xyz"  
len(name)  
name[0]  
name[:3]  
name.replace("abc", "XYZ")  
name.capitalize()  
name.upper()  
name.lower()  
'abc' in name
```

## Integer & Float

### Shell Scripting

```
num=10  
num=10.5
```

[Handled as a string in Shell]

### Python Scripting

```
num = 10  
num = 10.5
```

var="" (Empty string)

var = None

## Arrays/List

### Shell Scripting

```
arr=(1 2 3 4 5)
echo ${arr[1]}
arr+=(6)
unset arr[2]
echo ${#arr[@]}
$(printf "%s\n" "${arr[@]}")
arr3=("${arr1[@]}" "${arr2[@]}")
[[ " ${arr[@]} " =~ "3" ]]

for i in "${arr[@]"; do
    echo "$i"
done
```

### Python Scripting

```
arr = [1, 2, 3]
list = [1, apple, banana]
arr[1]
arr.append(6)
arr.remove(2)
len(arr)
arr.sort
arr3 = arr1 + arr2
if 3 in arr:

for i in arr:
    print(i)
```

## Tuple

### Shell Scripting

N/A

### Python Scripting

```
tpl = (1, 2, 3, 4, 5)
tpl[0]
len(tpl)
if 3 in tpl:
for i in tpl:
    print(i)
tpl[1] = 10 # Error!
```

## Dictionary

### Shell Scripting

```
declare -A dict=( [name]="Padmaja" [age]=25 )
echo ${dict[name]}
dict[city]="Pune"
unset dict[age]
[ -v dict[name] ]

for key in "${!dict[@]"; do
    echo "$key -> ${dict[$key]}"
done
```

### Python Scripting

```
dict = {"name": "Padmaja", "age": 25}
dict["name"]
dict["city"] = "Pune"
del dict["age"]
if "name" in dict:

for key, value in dict.items():
    print(f"{key} -> {value}")
```

## Print value

### Shell Scripting

```
name="abc"  
echo "My name is $name"
```

### Python Scripting

```
name = "abc"  
print(f"My name is {name}")
```

## Arithmetic Operations

### Shell Scripting

```
a=10  
b=5  
sum=$((a + b))  
diff=$((a - b))  
prod=$((a * b))  
quot=$((a / b))  
rem=$((a % b))  
((a++))  
((b--))
```

### Python Scripting

```
a = 10  
b = 5  
sum = a + b  
diff = a - b  
prod = a * b  
quot = a / b  
rem = a % b  
a += 1  
b -= 1
```

# Logical Operators

## Shell Scripting

```
if [[ $a -gt 10 && $b -lt 10 ]]; then
    echo "Both conditions are true"
fi

if [[ $a -gt 10 || $b -gt 10 ]]; then
    echo "At least one condition is true"
fi

if [[ ! $a -eq 10 ]]; then
    echo "a is not 10"
fi
```

## Python Scripting

```
a = 15
b = 5
```

```
if a > 10 and b < 10:
    print("Both conditions are true")

if a > 10 or b > 10:
    print("At least one condition is true")

if not a == 10:
    print("a is not 10")
```

# Identity Operators

## Shell Scripting

N/A

## Python Scripting

```
a = [1, 2, 3]
b = a
print(a is b) # True

a = [1, 2, 3]
b = [1, 2, 3]
print(a is not b) # True
(different memory locations)
```



# User Input

## Shell Scripting

```
*read -p "Enter your name: " name
echo "Hello, $name"

*read -p "Enter two numbers: " num1 num2
sum=$((num1 + num2))
echo "Sum: $sum"

*# Array Input
read -p "Enter numbers: " -a arr
echo "First number: ${arr[0]}"

*# Reading a Whole Line
IFS= read -r line
echo "You entered: $line"

*# Timed Input (5 seconds)
read -t 5 -p "Enter your name (within 5s): " name
echo "You entered: $name"

*# Silent Input (Password)
read -s -p "Enter password: " password
echo -e "\nPassword entered"
```

## Python Scripting

```
*name = input("Enter your name: ")
print(f"Hello, {name}")

* num1, num2 = map(int, input("Enter two numbers: ").split())
print(f"Sum: {num1 + num2}")

* # List Input
arr = list(map(int, input("Enter numbers: ").split()))
print(f"First number: {arr[0]}")

* # Reading a Whole Line
line = input("Enter a line: ")
print(f"You entered: {line}")

*Its complicated

# Silent Input (Password)
password = getpass.getpass("Enter password: ")
print("Password entered")
```

# If Else Condition

## Shell Scripting

```
str1="hello"
str2="hello"
# String Matching
if [[ "$str1" == "$str2" ]]; then
    echo "Strings are equal"
else
    echo "Strings are not equal"
fi

a=15
b=10
if [[ $a -gt $b ]]; then
    echo "a is greater than b"
fi

if [[ $a -gt 10 && $b -lt 20 ]]; then
    echo "Both conditions met"
fi

# Check if file exists
if [[ -f "test.txt" ]]; then
    echo "File exists"
fi

# Check if directory exists
if [[ -d "/tmp" ]]; then
    echo "Directory exists"
fi

# Check if word exists in a file
if grep -q "hello" test.txt; then
    echo "Word found in file"
fi
```

Handwritten notes for Shell Scripting:

- Brackets `{ }` are drawn around the variable assignments `a=15` and `b=10`.
- A red circle is drawn around `-gt` in the condition `$a -gt $b`. A red arrow points to a list of operators: `-eq or ==`, `-ge`, `-le`, `-ne or !=`, and `-lt`.
- A red circle is drawn around `&&` in the condition `$a -gt 10 && $b -lt 20`. A red arrow points to a box containing `||`.

## Python Scripting

```
str1="hello"
str2="hello"
a=15
b=10

if str1 == str2:
    print("Strings are equal")
else:
    print("Strings are not equal")

if a > b:
    print("a is greater than b")

# AND & OR Condition
if a > 10 and b < 20:
    print("Both conditions met")

str_value = "hello world"
if "world" in str_value:
    print("String contains 'world'")

if os.path.isfile("test.txt"):
    print("File exists")
if os.path.isdir("/tmp"):
    print("Directory exists")
```

Handwritten notes for Python Scripting:

- Brackets `{ }` are drawn around the variable assignments `a=15` and `b=10`.
- A red box is drawn around the comparison operators `>`, `>=`, `<`, and `<=`. A red arrow points from the `&&` operator in the Shell Scripting section to this box.
- A red arrow points from the `or` keyword in the Python code to the word `or` in the text `# AND & OR Condition`.

Note: For python

```
a=45
if a is 45:
    print(a)
--> it will print 45

b=[45, 56, 5, 3]
print(45 in b)
--> it will print true
```

```
def get_coordinates():
    return (3, 4)

x, y = get_coordinates() # Unpack the returned tuple (x=3, y=4)
```

## For Loop

### Shell Scripting

```
((i=1; i<=50; i+=5)) fruits=("apple" "banana" "cherry")  
((i=1; i<=5; i++)) for fruit in "${fruits[@]}"  
i in {1..3} line in $(cat list.txt)  
i in 1 2 3
```

```
for item in list; do
```

```
# Commands to execute for each item
```

```
done
```

### Python Scripting

```
with open("file.txt") as f:  
    for line in f:  
        for i in range(1, 11, 2): by two  
        for i in range(1, 6): increment by one
```

```
arr = ["apple", "banana", "cherry"]  
for item in arr:  
    print(item)
```

## While Loop

### Shell Scripting

```
-d "/tmp"  
-f "test.txt"  
"$str" == *"hello"*  
$a -gt 5 && $b -lt 20
```

```
while [[ $a -lt $b ]]; do
```

```
    echo "$a"
```

```
    ((a++))
```

```
done
```

```
while grep -q "hello" test.txt; do
```

```
# Read comma-separated values from a file
```

```
while IFS=, read -r var1 var2; do
```

```
    echo "First: $var1, Second: $var2"
```

```
done < file.txt
```

### Python Scripting

```
os.path.isdir("/tmp")  
os.path.isfile("test.txt")  
"hello" in str_value  
a > 5 and b < 20
```

```
while a < b:
```

```
    print(a)
```

```
    a += 1
```

```
while "hello" in open("test.txt").read():
```

```
with open("file.txt") as f:
```

```
    for line in f:
```

```
        var1, var2 = line.strip().split(",")
```

```
        print(f"First: {var1}, Second: {var2}")
```



# Function

## Shell Scripting

```
my_function() {  
    echo "Hello, World!"  
}  
my_function
```

```
greet() {  
    echo "Hello, $1!"  
}  
greet_default() {  
    echo "Hello, ${1:-Guest}!"  
}
```

```
greet_default  
greet_default "abc"
```

```
print_list() {  
    for item in "$@"; do  
        echo "$item"  
    done  
}  
print_list Apple Banana Orange
```

N/A

```
# Function with local and global variables  
global_var="Global"  
my_function_scope() {  
    local local_var="Local"  
    echo "$global_var - $local_var"  
}  
my_function_scope
```

## Python Scripting

```
def my_function():  
    print("Hello, World!")  
my_function()
```

```
def greet(name):  
    print(f"Hello, {name}!")  
def greet_default(name="Guest"):  
    print(f"Hello, {name}!")
```

```
greet_default()  
greet_default("abc")
```

```
def print_list(items):  
    for item in items:  
        print(item)
```

```
print_list(["Apple", "Banana", "Orange"])
```

```
# Function with a dictionary
```

```
def print_dict(data):  
    for key, value in data.items():  
        print(f"{key}: {value}")
```

```
print_dict({"name": "abc", "role":  
            "DevOps Engineer"})
```

```
# Function with local and global variables  
global_var = "Global"
```

```
def my_function_scope():  
    local_var = "Local"  
    print(f"{global_var} - {local_var}")
```

```
my_function_scope()
```



# Module

## Shell Scripting

Shell Module (utils.sh)

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

(main.sh)

```
#!/bin/bash
```

```
source utils.sh
```

```
result=$(add 10 20)
```

```
echo "Addition result: $result"
```

## Python Scripting

Python Module (utils.py)

```
def add(a, b):  
    return a + b
```

(main.py)

```
import utils
```

```
result = utils.add(10, 20)
```

```
print(f"Addition result: {result}")
```

# Package

## Shell Scripting

N/A

## Python Scripting

mypackage/

```
|— __init__.py # Marks this directory as a package  
|— math_utils.py # Module inside the package
```

```
def add(a, b):
```

```
    return a + b
```

(main.py)

```
import mypackage.math_utils
```

```
result = mypackage.math_utils.add(3, 5)
```

```
print(f"Addition result: {result}")
```

# Regular Expressions

## Shell Scripting

## Python Scripting

### 1. Matching a Pattern

```
text="Hello World"

# Check if "Hello" exists in the text
if [[ "$text" =~ Hello ]]; then
    echo "Pattern Matched"
else
    echo "No Match"
fi
```

```
# Check if "Hello" exists in the text
if re.search(r"Hello", text):
    print("Pattern Matched")
else:
    print("No Match")
```

### 2. Replacing Text

```
new_text=$(echo "$text" | sed 's/World/Python/')
echo "$new_text"
```

```
new_text = re.sub(r"World", "Python", text)
print(new_text)
```

### 3. Extracting Text

```
text="My number is 9876543210"

# Extract digits from text
echo "$text" | grep -oE '[0-9]+'
```

```
text = "My number is 9876543210"

# Extract digits from text
numbers = re.findall(r'\d+', text)
print(numbers)
```

# File Operations

## Shell Scripting

## Python Scripting

### 1. Opening and Reading a File

```
while IFS= read -r line; do
    echo "Line: $line"
done < myfile.txt
```

```
with open("myfile.txt", "r") as file:
    for line in file:
        print("Line:", line.strip())
```

### 2. Writing to a File

```
echo "Hello, this is a test file" > myfile.txt
```

```
with open("myfile.txt", "w") as file:
    file.write("Hello, this is a test file\n")
```

```
echo "Adding another line" >> myfile.txt
```

```
with open("myfile.txt", "a") as file:
    file.write("Adding another line\n")
```

### 3. Reading a File Word by Word

```
while read -r word; do
    echo "Word: $word"
done < <(cat myfile.txt)
```

```
with open("myfile.txt", "r") as file:
    for line in file:
        for word in line.split():
            print("Word:", word)
```

### 5. Reading Specific Lines

```
head -n 3 myfile.txt
```

```
with open("myfile.txt", "r") as file:
    lines = file.readlines()
    print("First 3 lines:", lines[:3])
```

```
tail -n 3 myfile.txt
```

```
print("Last 3 lines:", lines[-3:])
```

### 6. Writing Multiple Lines to a File

```
cat <<EOF > myfile.txt
This is line 1
This is line 2
This is line 3
EOF
```

```
lines = ["This is line 1\n", "This is line 2\n", "This is line 3\n"]

with open("myfile.txt", "w") as file:
    file.writelines(lines)
```

# Break

Exit the Loop

## Shell Scripting

```
for i in {1..5}; do
    if [[ $i -eq 3 ]]; then
        echo "Breaking loop at $i"
        break
    fi
    echo "Iteration: $i"
done
```

## Python Scripting

```
for i in range(1, 6):
    if i == 3:
        print("Breaking loop at", i)
        break
    print("Iteration:", i)
```

# continue

Skip the Current Iteration

## Shell Scripting

```
for i in {1..5}; do
    if [[ $i -eq 3 ]]; then
        echo "Skipping iteration $i"
        continue
    fi
    echo "Iteration: $i"
done
```

## Python Scripting

```
for i in range(1, 6):
    if i == 3:
        print("Skipping iteration", i)
        continue
    print("Iteration:", i)
```



# sleep

Pause Execution

## Shell Scripting

```
echo "Sleeping for 2 seconds..."
sleep 2
echo "Resumed execution"
```

## Python Scripting

```
print("Sleeping for 2 seconds...")
time.sleep(2)
print("Resumed execution")
```

# exit

Terminate the Script

## Shell Scripting

```
echo "Script executed successfully"
exit 0 # 0 indicates success
```

```
echo "An error occurred"
exit 1 # 1 indicates failure
```

## Python Scripting

```
print("Script executed successfully")
sys.exit(0) # 0 indicates success
```

```
print("An error occurred")
sys.exit(1) # 1 indicates failure
```

```
./myscript.sh
if [[ $? -eq 0 ]]; then
    echo "Script ran successfully"
else
    echo "Script failed"
fi
```

```
try:
    print("Running script...")
    sys.exit(1) # Simulating failure
except SystemExit as e:
    if e.code == 0:
        print("Script ran successfully")
    else:
        print("Script failed")
```

# Environment Variables

## Shell Scripting

```
# Temporary environment variable (for current session)
MY_VAR="Hello, World"
echo "MY_VAR: $MY_VAR"

# Export variable (available to child processes)
export MY_VAR="Hello, Exported World"

unset MY_VAR # Removes MY_VAR from the environment
printenv # List all environment variables
```

## Python Scripting

```
# Temporary environment variable
os.environ["MY_VAR"] = "Hello, World"

print("MY_VAR:", os.environ["MY_VAR"])

del os.environ["MY_VAR"] # Removes MY_VAR from the environment
print(os.environ)
```

# Exception Handling

## Shell Scripting

exit codes, trap commands,  
and conditional statements.

## Python Scripting

try-except blocks

### 1. Basic Exception Handling

```
# Try to access a non-existent file
if ! cat myfile.txt; then
    echo "Error: File not found"
fi
```

```
try:
    with open("myfile.txt", "r") as file:
        content = file.read()
except FileNotFoundError:
    print("Error: File not found")
```

### 2. Handling Multiple Exceptions

```
if ! mkdir /root/test 2>/dev/null; then
    echo "Error: Permission denied or directory creation failed"
fi
```

```
try:
    with open("myfile.txt", "r") as file:
        content = file.read()
    num = 10 / 0 # This will cause an exception
except FileNotFoundError:
    print("Error: File not found")
except ZeroDivisionError:
    print("Error: Division by zero is not allowed")
```

```
trap 'echo "An error occurred. Exiting..."; exit 1' ERR

ls /root # This will cause an error if not run as root
echo "This line won't be executed if an error occurs above"
```

```
def error_handler():
    print("An error occurred. Exiting...")
    sys.exit(1)

try:
    with open("/root/myfile.txt", "r") as file:
        content = file.read()
except Exception:
    error_handler()
```

## Commands in Parallel

### Shell Scripting

```
echo "Task 1" &
echo "Task 2" &
wait # Wait for all background jobs
```

### Python Scripting

```
# Running multiple commands in parallel using subprocess
p1 = subprocess.Popen(["echo", "Task 1"])
p2 = subprocess.Popen(["echo", "Task 2"])

p1.wait() # Wait for process 1
p2.wait() # Wait for process 2
```