**Topics**

1. **Functions**:
   * **Definition**: A function is a reusable block of code that performs a specific task.
   * **Syntax**:

python

Copy code

def function\_name(parameters):

# Code block

return value

* + Functions can take arguments, process them, and return results.

1. **Modules**:
   * Modules are Python files containing reusable code (functions, classes, etc.).
   * **Built-in Modules**: os, sys, subprocess, etc.
   * **Importing Modules**:

python

Copy code

import module\_name

from module\_name import specific\_function

1. **Common DevOps Modules**:
   * os: For interacting with the operating system.
   * subprocess: For executing shell commands.

**Practical Exercises**

**1. Create a Simple Function**

* Script:

python

Copy code

def greet(name):

return f"Hello, {name}! Welcome to Python for DevOps."

# Call the function

print(greet("Shakti"))

* Output:

css

Copy code

Hello, Shakti! Welcome to Python for DevOps.

**2. Write a Function to Check Server Uptime**

* Objective: Simulate a server uptime check.
* Script:

python

Copy code

def check\_uptime(server\_name):

# Simulate uptime check

return f"The server {server\_name} has been running for 24 hours."

# Test the function

server = "web\_server1"

print(check\_uptime(server))

* Output:

arduino

Copy code

The server web\_server1 has been running for 24 hours.

**3. Use the os Module to Interact with the System**

* Objective: List all files in the current directory.
* Script:

python

Copy code

import os

def list\_files(directory):

return os.listdir(directory)

# Call the function

print("Files in the current directory:")

print(list\_files("."))

* Output:

less

Copy code

Files in the current directory:

['file1.py', 'file2.log', 'welcome.py']

**4. Execute Shell Commands Using subprocess**

* Objective: Run system commands and capture their output.
* Script:

python

Copy code

import subprocess

def run\_command(command):

result = subprocess.run(command, shell=True, capture\_output=True, text=True)

return result.stdout

# Call the function

print("Disk Usage:")

print(run\_command("df -h"))

* Output (example):

bash

Copy code

Disk Usage:

Filesystem Size Used Avail Use% Mounted on

/dev/sda1 50G 25G 25G 50% /

**5. Create a Module for Reuse**

* Create a file named utilities.py and add the following:

python

Copy code

def add\_numbers(a, b):

return a + b

def multiply\_numbers(a, b):

return a \* b

* Create a separate script to import and use the module:

python

Copy code

from utilities import add\_numbers, multiply\_numbers

print("Addition:", add\_numbers(5, 3))

print("Multiplication:", multiply\_numbers(5, 3))

* Output:

makefile

Copy code

Addition: 8

Multiplication: 15

**6. Challenge: Automate Server Checks**

* Objective: Create a script that checks multiple servers and executes a shell command on them.
* Script:

python

Copy code

import subprocess

def check\_server(server\_name):

# Simulate ping command (use 'ping' in Unix or 'ping -n 1' on Windows)

command = f"ping -c 1 {server\_name}"

result = subprocess.run(command, shell=True, capture\_output=True, text=True)

if result.returncode == 0:

return f"{server\_name} is reachable."

else:

return f"{server\_name} is not reachable."

servers = ["192.168.1.10", "192.168.1.20", "192.168.1.30"]

for server in servers:

print(check\_server(server))

* Test with your server IPs or dummy values.

**7. Bonus: Function to Combine Shell Command Outputs**

* Objective: Fetch disk usage and memory usage in one function.
* Script:

python

Copy code

import subprocess

def system\_status():

disk\_usage = subprocess.run("df -h", shell=True, capture\_output=True, text=True).stdout

memory\_usage = subprocess.run("free -h", shell=True, capture\_output=True, text=True).stdout

return f"Disk Usage:\n{disk\_usage}\nMemory Usage:\n{memory\_usage}"

print(system\_status())

* Output (example):

vbnet

Copy code

Disk Usage:

Filesystem Size Used Avail Use% Mounted on

/dev/sda1 50G 25G 25G 50% /

Memory Usage:

total used free shared buff/cache available

Mem: 8G 2G 4G 1G 2G 6G