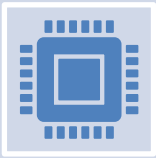
Python OS Module Overview

An introduction to the OS module with practical exercises and detailed solutions

Introduction to OS Module

The os module in Python provides a way of using operating system dependent functionality.



It allows you to interface with the underlying operating system that Python is running on – be it Windows, Mac, or Linux.

You can perform several operations like navigating the file system, obtaining file information, and managing processes.

Exercises Overview

Ping	Ping a Host
Trace	Trace Route
Create	Create a Folder
Change	Change Working Directory
Delete	Delete a Folder
List	List Files in a Directory
Check	Check File Existence
Rename	Rename a File
Remove	Remove a File
Get	Get Environment Variables

Ping a Host

```python
import os

try:
    with open("output.txt", "w") as file:
        # Ensure the file is accessible
        pass
    os.system("ping 8.8.8.8 > output.txt")
except IOError as e:
    print(f"Error: {e.strerror}")
```

Use os.system() to simulate pinging a website or IP.

Checking ping

1. Open a File for Reading
2. Iterate Through Each Line
3. Remove Newline Characters
4. Check for Specific Substring ("ms")
5. Print Success Message and Break
6. Print End Message

```python
import os

file=open('output.txt','r')
for line in file:
    newline=line.replace('\n','')
    if "ms" in newline:
        print("The connecting successful!!")
        break
print("End")
```

```
The connecting successful!!
End

Process finished with exit code 0
```

Trace Route

- Trace the route packets take to a network host.

```python
import os
a=os.system('tracert www.google.com')
print(a)
```

```
Tracing route to www.google.com [2a00:1450:4028:808::2004]
over a maximum of 30 hops:

  1     1 ms     1 ms     1 ms  2a0d:6fc0:e23:1800::1
  2     4 ms     3 ms     3 ms  2a0d:6fc0:fff:ffff:ffff:ffff:ffff:ff17
  3     *        *        *     Request timed out.
  4     6 ms     5 ms     4 ms  2001:40a8:4431:1::1
  5     5 ms     5 ms     4 ms  2001:4860:1:1::2734
  6     5 ms     5 ms     5 ms  2a00:1450:805c::1
  7     5 ms     5 ms     6 ms  2001:4860:0:1::90
  8     5 ms     5 ms     6 ms  2001:4860:0:1::566
  9     8 ms     5 ms     5 ms  2001:4860:0:1::861b
 10     5 ms     4 ms     5 ms  2001:4860:0:1::6925
 11     5 ms     4 ms     4 ms  tlv03s01-in-x04.1e100.net [2a00:1450:4028:808::2004]

Trace complete.
0

Process finished with exit code 0
```

Create a Folder and change working directory

- Create a new folder at a specified path using os.mkdir() and chdir().

```python
import os
if not os.path.exists('newfolder'):
    os.mkdir('newfolder')
    os.chdir('newfolder')
print(os.getcwd())
```

```
C:\limudeyhutz\6-20240401 שיעורT130507Z-001\6 שיעור\ExLesson5

Process finished with exit code 0
```

Delete a Folder

```python
import os
os.rmdir('newfolder')
```

- Remove an empty directory using os.rmdir().

- Solution:
- os.rmdir('/path/new_folder')

List Files

List all files and folders in a specified directory.

```python
import os

# Step 1: Create a new folder in the current directory
folder_name = "folder1"
os.makedirs(folder_name, exist_ok=True)  # Use exist_ok=True to avoid raising an error if the folder already exists

# Step 2: Change the working directory to the newly created folder
os.chdir(folder_name)

# Step 3: List the contents of the folder
folder_contents = os.listdir()

# Step 4: Print the contents of the folder
print("Contents of", folder_name + ":")
for item in folder_contents:
    print(item)
```

Ipconfig and found the ip

01

Execute the ipconfig command using the subprocess module.

02

Capture the command's output.

03

Parse the output to locate the line containing "IPv4 Address".

04

Extract the IP address from that line.

05

Print the IP address.

```python
import os

# Initialize a flag to indicate whether the IPv4 address has been found
ipv4_found = False

# Run the 'ipconfig' command and capture its output
with os.popen('ipconfig') as cmd:
    # Read each line from the command output
    for line in cmd:
        newline=line.replace('\n','')
        # Check if 'IPv4' is in the current line
        if 'IPv4 Address' in newline:
            # Extract the IPv4 address (assuming it follows 'IPv4 Address' and some spaces)
            ipv4_address = newline.split(':')[-1].strip()
            print("IPv4 Address found:", ipv4_address)
            ipv4_found = True
            break  # Break out of the loop after finding the IPv4 address

# If no IPv4 address was found in the output, print a different message
if not ipv4_found:
    print("No IPv4 Address found.")
```

solution

Exercise 1

Develop a Python script leveraging the os module to execute ping, ipconfig, and tracert commands, saving the outputs to separate files, and then printing the contents of those files.

- **Steps:**

- **Ping Command:** Execute a ping command targeting a website and save the output to a file.
- **Ipconfig Command:** Run the ipconfig command and save its output to a file.
- **Tracert Command:** Execute a tracert command to trace the route to a specified website and save the output to a file.
- Print Files: Open and print the contents of each file created in the previous steps.

- **Expected Outcome:**
- The script successfully executes each command and saves the output to separate files.
- The contents of each file are correctly printed, displaying the results of the ping, ipconfig, and tracert commands.

Exercise 2

Create Folder: Generate a new folder in the current directory.

Navigate to Folder: Change the script's working directory to the new folder.

Execute Ping Command: Run the ping command targeting a specified website and redirect the output to a file within the newly created folder.

Analyze Results: Open the file, iterate through its contents to detect if the connection was successful, and print a corresponding message.

Expected Behavior:

The script successfully creates a new folder and sets it as the current working directory.

The ping command's output is saved to a file within this folder.

The script reads the file, identifies a successful connection, and prints a confirmation message. If unsuccessful, it prints an error message.

Exercise 3

Execute Command with Redirection: Use os.system() to run the ipconfig command, redirecting its output to a predetermined file.

Read and Parse Output: Open the designated file, read through its contents line by line, and look for lines containing "IPv4 Address" or "IPv6 Address".

Extract and Print Addresses: For each relevant line, extract the address, and print it clearly labeled as either IPv4 or IPv6.

Clean Up: After processing the file, delete it to clean up any temporary data.

Expected Outcome:

The script will print each found IPv4 and IPv6 address.

If no addresses are found, appropriate messages indicating the absence of addresses will be displayed.

The output file will be removed after the extraction process.

```python
import os

# Execute ping command and save output to a file
os.system('ping 8.8.8.8 > ping_output.txt')

# Execute ipconfig command and save output to a file
os.system('ipconfig > ipconfig_output.txt')

# Execute tracert command and save output to a file
os.system('tracert www.example.com > tracert_output.txt')

# Print the contents of each file
for filename in ['ping_output.txt', 'ipconfig_output.txt', 'tracert_output.txt']:
    print(f"Contents of {filename}:")
    with open(filename, 'r') as file:
        print(file.read())
    print()  # Add an empty line for separation
```

Solution exercise 1

```python
import os
# Step 1: Create a new folder in the current directory
folder_name = "ping_results"
os.mkdir(folder_name)

# Step 2: Change the working directory to the newly created folder
os.chdir(folder_name)

# Step 3: Execute the ping command targeting a specified website and redirect the output to a file
website = "www.example.com"
ping_command = f'ping {website} > ping_output.txt'
os.system(ping_command)

# Step 4: Analyze the results
ping_successful = False
with open('ping_output.txt', 'r') as file:
    for line in file:
        newline=line.replace("\n", "")
        if "Reply from" in newline:  # Check if the line contains "Reply from", indicating a successful ping
            ping_successful = True
            break

# Step 5: Print a corresponding message
if ping_successful:
    print("Ping was successful. Connection is active.")
else:
    print("Ping failed. No connection.")
```

Solution exercise 2

Solution exercise 3

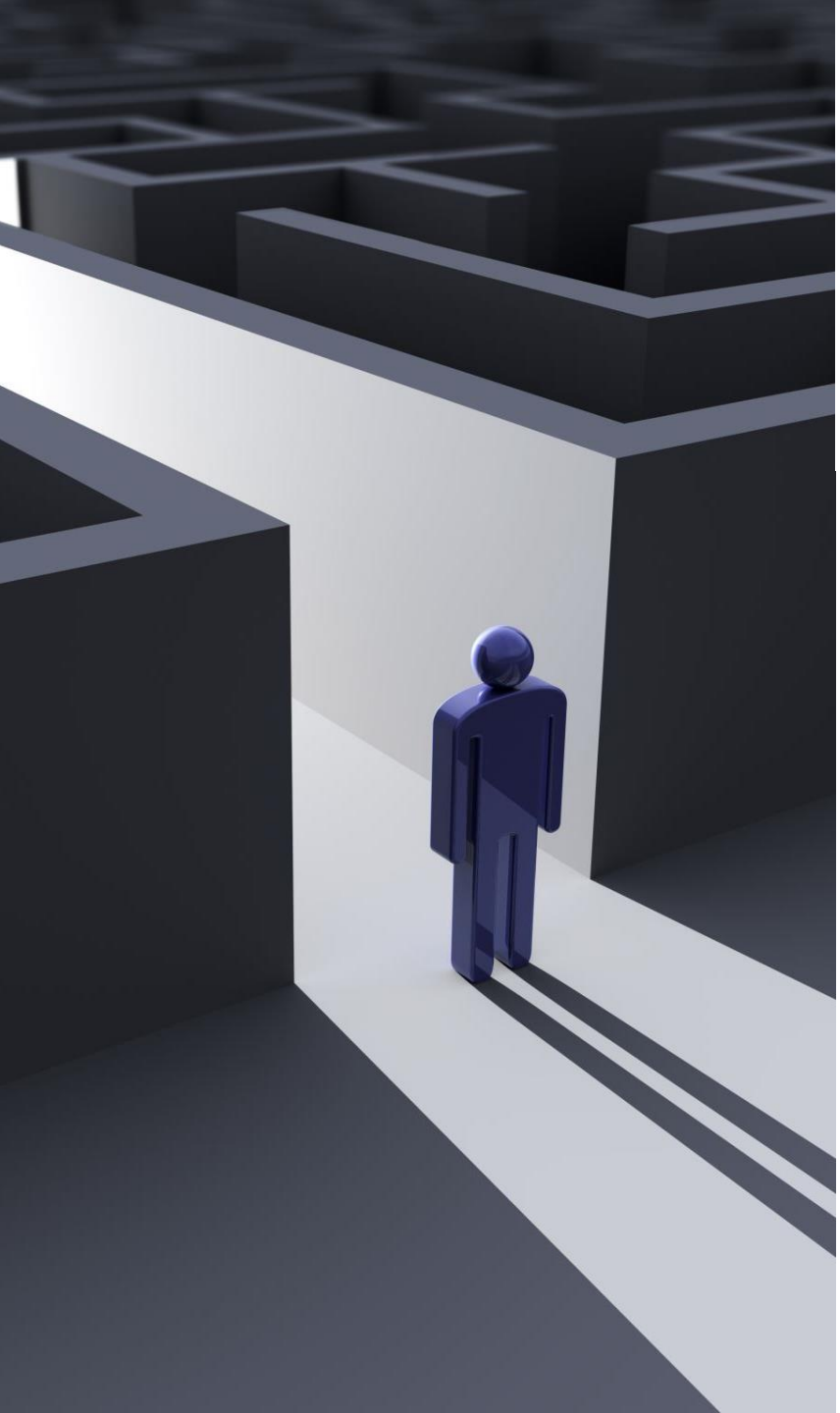
```python
import os

# Define a fixed file name for the output
output_file = "ipconfig_output.txt"

# Run 'ipconfig' and redirect its output to the file
os.system(f'ipconfig > {output_file}')

# Now, open the output file to read the contents
try:
    with open(output_file, 'r') as file:
        ipv4_found = False
        ipv6_found = False
        # Read each line from the file
        for line in file:
            # Check for IPv4 and IPv6 addresses
            if 'IPv4 Address' in line:
                ipv4_address = line.split(':')[-1].strip()
                print("IPv4 Address found:", ipv4_address)
                ipv4_found = True
            elif 'IPv6 Address' in line and not 'Temporary' in line:
                ipv6_address = line.split(':')[-1].strip()
                print("IPv6 Address found:", ipv6_address)
                ipv6_found = True

        # Messages if no addresses were found
        if not ipv4_found:
            print("No IPv4 Address found.")
        if not ipv6_found:
            print("No IPv6 Address found.")
finally:
    # Remove the output file after reading to clean up
    os.remove(output_file)
```



Check File Existence

```
print(os.path.exists('folder1'))
```

```
True
```

- Check if a specified file exists using os.path.exists().

- Solution:
- os.path.exists('/path/file.txt')

Rename a File

Rename a file from old name to new name using os.rename().

Solution:

os.rename('old_name.txt', 'new_name.txt')

Remove a File

Delete a specified file from the filesystem.

Solution:

```
os.remove('file_path.txt')
```

Get Environment Variables

Print all environment variables accessible to OS.

Solution:

```
import os
print(os.environ)
```

Conclusion

The os module is essential for managing operating system tasks, directories, and files directly from Python.

These exercises provide a practical approach to mastering its functions.