Lab Assignment & Solution

Cybersecurity Professional Program

Introduction to Python for Security

Functions

PY-05-LS7 Directory Listing

Note: Solutions for the instructor are shown inside the green box.

Lab Objective

Understand how to use recursion to interact with the file system.

Lab Mission

Implement recursion to work with the file system while using various OS module functions.

Lab Duration

30–40 minutes

Requirements

- Basic knowledge of Python
- Working knowledge of functions and the OS module

Resources

- Environment & Tools
    - Windows or Linux VM
        - PyCharm
        - Python 3

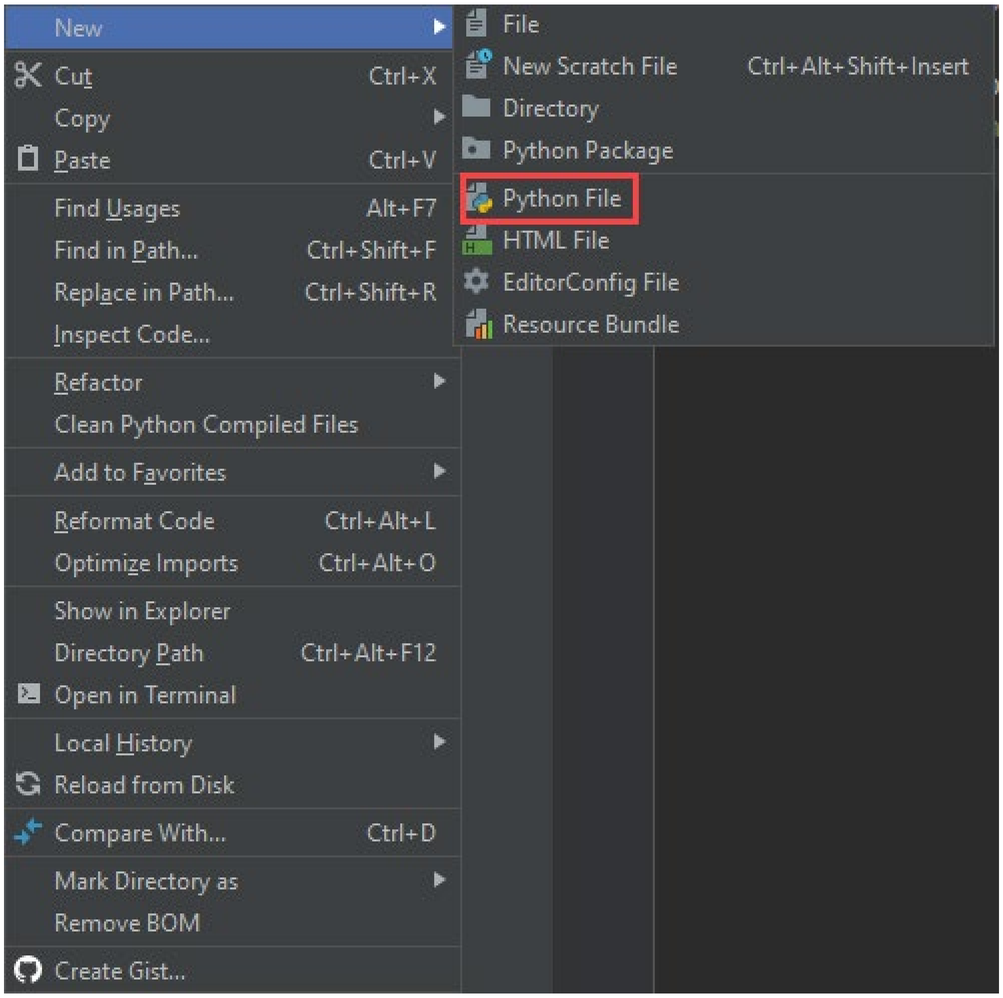
Textbook References

- Chapter 5: Functions
    - Section 3: Recursion

Lab Task: Directory Listing in Python

Write a program that will print all files in a directory, as well as the name and size of each file, and iterate through any directories found.

1 Create a new Python file in PyCharm by right clicking the project you created and selecting **New** > ***Python File***.



2 Import the ***os*** and ***sys*** modules.

```
import os
import sys
```

3 Create a variable to act as a separator between the output: **<----->**

```
line = "<------------------------------------------------------->"
```

4 Create a variable to store the system's type and print it.

```
system = sys.platform
print("You are using {}".format(system))
```

5 Create a variable to store user input of a directory path.

```
root_folder = input("Enter folder: ")
```

6 Create a function that accepts a parameter that will be a directory path. The function will be dedicated to handling the mapping operation.

```
def mapper(path):
```

7 In the function, create a loop to iterate over the directory's content.

```
def mapper(path):
      for item in os.listdir(path):
```

8 Create a variable to store the full path of an iterated item.

```
def mapper(path):
      for item in os.listdir(path):
            full_path = r"{}\{}".format(path, item)
```

9 Create a condition to check if the full path leads to a file.

```
def mapper(path):
      for item in os.listdir(path):
            full_path = r"{}\{}".format(path, item)
            if os.path.isfile(full_path):
```

10 If the path leads to a file, create a variable to save the file's size and print it.

```python
def mapper(path):
    for item in os.listdir(path):
        full_path = r"{}\{}".format(path, item)
        if os.path.isfile(full_path):
            size = os.stat(full_path).st_size
            print("Found {} -> weighs {}
bytes.".format(full_path, size))
```

11 Create a condition to check if the full path leads to a directory.

```python
def mapper(path):
    for item in os.listdir(path):
        full_path = r"{}\{}".format(path, item)
        if os.path.isfile(full_path):
            size = os.stat(full_path).st_size
            print("Found {} -> weighs {} bytes.".format(full_path,size))
        elif os.path.isdir(full_path):
```

12 If the path leads to a directory, print a message stating that the program is entering the directory, and invoke the function recursively.

```python
def mapper(path):
    for item in os.listdir(path):
        full_path = r"{}\{}".format(path, item)
        if os.path.isfile(full_path):
            size = os.stat(full_path).st_size
            print("Found {} -> weighs {} bytes.".format(full_path,size))
        elif os.path.isdir(full_path):
            print("{}\nEntering folder {}\n{}".format(line, item, line))
            mapper(full_path)
```

13 If the path leads to anything but a directory or a file, print a message that states that the object is unknown.

```
def mapper(path):
      for item in os.listdir(path):
           full_path = r"{}\{}".format(path, item)
           if os.path.isfile(full_path):
                 size = os.stat(full_path).st_size
                 print("Found {} -> weighs {} bytes.".format(full_path, size))
           elif os.path.isdir(full_path):
                 print("{}\nEntering folder {}\n{}".format(line, item, line))
                 mapper(full_path)
             else:
                    print("Unknown.")
```

14 Place the code in the function within a ***try*** block and catch a potential exception.

```
def mapper(path):
    try:
        for item in os.listdir(path):
            full_path = r"{}\{}".format(path, item)
            if os.path.isfile(full_path):
                size = os.stat(full_path).st_size
                print("Found {} -> weighs {} bytes.".format(full_path, size))
            elif os.path.isdir(full_path):
                print("{}\nEntering folder {}\n{}".format(line, item, line))
                mapper(full_path)
            else:
                print("Unknown.")

    except Exception as error:
        print(error)
```

15

16 Invoke the mapping function.

```
mapper(root_folder)
```