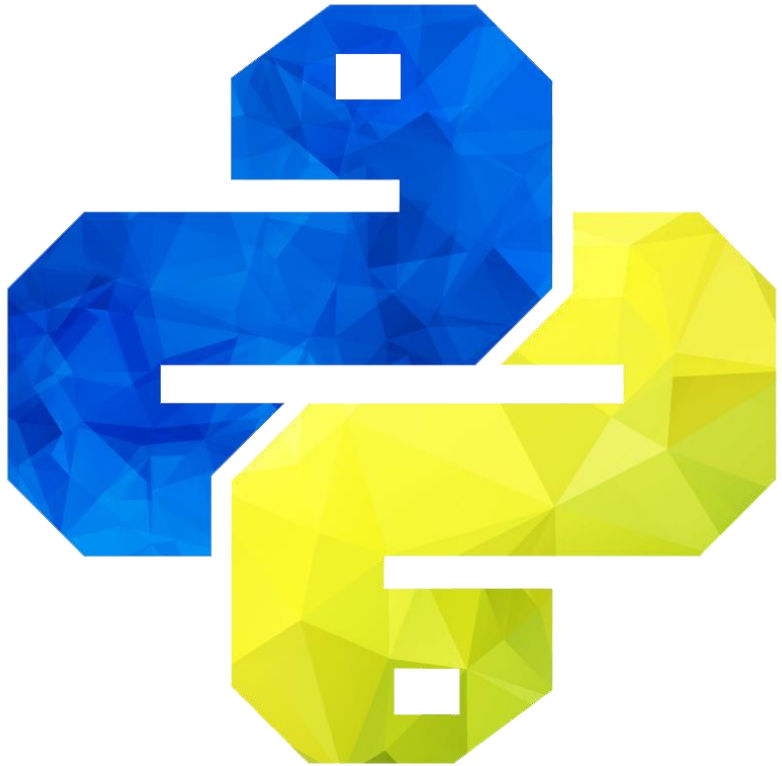
More with Modules

Cybersecurity Boot Camp

Advanced Python: Day 3

Python: Reading and Writing Files

Today we will cover Python's built-in modules.



Class Objectives

By the end of class today, you will be able to:

- Return a file's metadata using the `os.stat()` function
- Use functions to get the size of a file, when it was last accessed, and when it was last modified
- Convert dates and times from one format to another using the the datetime module
- Remove files from a system using the `os.remove()` function.
- Extract zip files using the zipfile module.

In previous classes, we covered:

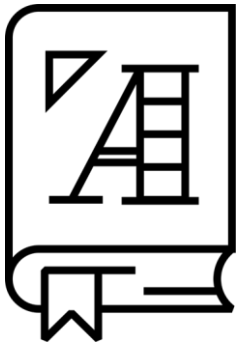
Data	Logic
1. Numbers	1. Operators
2. Strings	2. Conditionals
3. Booleans	3. Loops
4. Lists	4. Functions
5. Dictionaries	5. Modules

Today's Class:

Data	Logic
1. Numbers	1. Operators
2. Strings	2. Conditionals
3. Booleans	3. Loops
4. Lists	4. Functions
5. Dictionaries	5. Modules (More)

File Statistic (Metadata)

Metadata



Metadata contains information such as file type, file size, file source, and when the file was last accessed.

Metadata can include clues about suspicious data during a security investigation.

File Statistics

The `os.stat()` function takes in a file as a parameter and returns an object that contains metadata on that file. In the example, we're saving this object into a variable.

```
import os

file_path = os.path.join("Resources", "SomeCoolStuff.txt")
statInfo = os.stat(file_path)

print(statInfo.st_size)
timeAccessed = statInfo.st_atime

import datetime

timeAccessed = datetime.datetime.fromtimestamp(timeAccessed).strftime('%c')
print(timeAccessed)


timeChanged = statInfo.st_mtime
timeChanged = datetime.datetime.fromtimestamp(timeChanged).strftime('%c')
print(timeChanged)

os.remove("Resources/BigOlWallpaper.jpg")
```

Create a reference to the file path that you want to analyze.

File Statistics

The `os.stat()` function takes in a file as a parameter and returns an object that contains metadata on that file. In the example, we're saving this object into a variable.

```
import os

file_path = os.path.join("Resources", "SomeCoolStuff.txt")
statInfo = os.stat(file_path)

print(statInfo.st_size)
timeAccessed = statInfo.st_atime

import datetime

timeAccessed = datetime.datetime.fromtimestamp(timeAccessed).strftime('%c')
print(timeAccessed)


timeChanged = statInfo.st_mtime
timeChanged = datetime.datetime.fromtimestamp(timeChanged).strftime('%c')
print(timeChanged)

os.remove("Resources/BigOlWallpaper.jpg")
```

Collect the stats and save it to a variable

File Statistics

The `os.stat()` function takes in a file as a parameter and returns an object that contains metadata on that file. In the example, we're saving this object into a variable.

```
import os

file_path = os.path.join("Resources", "SomeCoolStuff.txt")
statInfo = os.stat(file_path)

print(statInfo.st_size)
timeAccessed = statInfo.st_atime

import datetime

timeAccessed = datetime.datetime.fromtimestamp(timeAccessed).strftime('%c')
print(timeAccessed)


timeChanged = statInfo.st_mtime
timeChanged = datetime.datetime.fromtimestamp(timeChanged).strftime('%c')
print(timeChanged)

os.remove("Resources/BigOlWallpaper.jpg")
```

To collect the file size in bytes…

File Statistics

The `os.stat()` function takes in a file as a parameter and returns an object that contains metadata on that file. In the example, we're saving this object into a variable.

```
import os

file_path = os.path.join("Resources", "SomeCoolStuff.txt")
statInfo = os.stat(file_path)

print(statInfo.st_size)
timeAccessed = statInfo.st_atime

import datetime

timeAccessed = datetime.datetime.fromtimestamp(timeAccessed).strftime('%c')
print(timeAccessed)


timeChanged = statInfo.st_mtime
timeChanged = datetime.datetime.fromtimestamp(timeChanged).strftime('%c')
print(timeChanged)

os.remove("Resources/BigOlWallpaper.jpg")
```

To collect the last time file was accessed (opened) in unix epoch time

File Statistics

The `os.stat()` function takes in a file as a parameter and returns an object that contains metadata on that file. In the example, we're saving this object into a variable.

```
import os

file_path = os.path.join("Resources", "SomeCoolStuff.txt")
statInfo = os.stat(file_path)

print(statInfo.st_size)
timeAccessed = statInfo.st_atime

import datetime

timeAccessed = datetime.datetime.fromtimestamp(timeAccessed).strftime('%c')
print(timeAccessed)


timeChanged = statInfo.st_mtime
timeChanged = datetime.datetime.fromtimestamp(timeChanged).strftime('%c')
print(timeChanged)

os.remove("Resources/BigOlWallpaper.jpg")
```

Epoch time is annoyingly hard to read, so let's convert that into an actual datetime format

File Statistics

The `os.stat()` function takes in a file as a parameter and returns an object that contains metadata on that file. In the example, we're saving this object into a variable.

```
import os

file_path = os.path.join("Resources", "SomeCoolStuff.txt")
statInfo = os.stat(file_path)

print(statInfo.st_size)
timeAccessed = statInfo.st_atime

import datetime

timeAccessed = datetime.datetime.fromtimestamp(timeAccessed).strftime('%c')
print(timeAccessed)

timeChanged = statInfo.st_mtime
timeChanged = datetime.datetime.fromtimestamp(timeChanged).strftime('%c')
print(timeChanged)

os.remove("Resources/BigOlWallpaper.jpg")
```

This takes the timeAccessed timestamp and converts it into a much more comprehensive form

File Statistics

The `os.stat()` function takes in a file as a parameter and returns an object that contains metadata on that file. In the example, we're saving this object into a variable.

```
import os

file_path = os.path.join("Resources", "SomeCoolStuff.txt")
statInfo = os.stat(file_path)

print(statInfo.st_size)
timeAccessed = statInfo.st_atime

import datetime

timeAccessed = datetime.datetime.fromtimestamp(timeAccessed).strftime('%c')
print(timeAccessed)


timeChanged = statInfo.st_mtime
timeChanged = datetime.datetime.fromtimestamp(timeChanged).strftime('%c')
print(timeChanged)

os.remove("Resources/BigOlWallpaper.jpg")
```

If a potentially malicious file is uncovered sometimes it is best to just remove it immediately.

This can be accomplished by `os.remove()`

Activity: Searching the Red Flag Sea

In this activity, you will write a script that searches a folder for any files that seem out of place.

Activities/ 03-Stu_SearchRedFlagSea/RedFlagSea.py
Activities/ 03-Stu_SearchRedFlagSea/Text.zip

Suggested Time:
25 minutes

Your Turn: Searching the Red Flag Sea

Instructions:

In this activity, a script that searches a folder for any files that seem out of place.

All of the files should be the same size and have been modified around the same time, but some may have had some additional changes made to them and they should stand out.

Write a Python script that reads through all of the files in the Text folder. The script should print out a line that looks like the following for each file:

- File Path (folder\name), File Size in Bytes, Last Modified Date..
- Example: Text\zrJ1cdXhuZOB, 100, Sun Jul 29 10:55:57 2018
- We've provided a lot of the solution code for you in the solution file, and the comments should guide you the rest of the way.

25 minutes

Activity: Red Flag CSV

In this activity, you will work with the code you wrote in the previous exercise and modify it so that the application creates a new CSV with the file metadata inside of it

Suggested Time:
10 minutes

Your Turn: Red Flag CSV

Instructions:

Using the previous activity as a jumping off point, modify the application so that instead of printing out all of the file metadata to the screen, it will instead create a CSV file that holds all of the information instead.

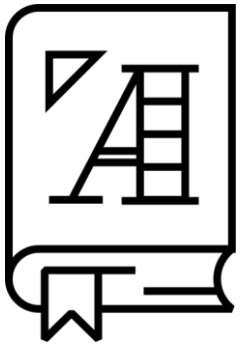
You only need to add and modify two lines of your original code. We provided the script file for you with where this needs to happen.

Open: Activities/04-Stu_RedFlagCSV/Unsolved/RedFlagSea.csv

Take a Break!

The Zip File Module

Built-In Modules



Zip file format is a common archive and compression standard that works by taking large files or collections of files and compresses them into a form that is more easily sent and received.

`zipfile` Module (Activities/ 05-Ins_ZipfileModule/ ZipFiles.py)

Providing users with the ability to read, write, append and list a zip file

```
# Import the zipfile module into the application
import zipfile

# Create a reference to the ZIP file you would like to extract
zipfileReference = zipfile.ZipFile("Text.zip")

# Run the `.extractall()` function on the variable containing the
connection to the external ZIP file
zipfileReference.extractall()

# creating another reference, this time to a locked ZIP file
lockedZip = zipfile.ZipFile("Books.zip")

# Variable used to store the password string
password = "password"

# Running the `.extractall()` function with a password
lockedZip.extractall(pwd=password.encode('cp850','replace'))
```

`zipfile.ZipFile` function returns a zip file object. In the example, we've stored the returned object into a variable.

We save that object into a variable, and then we run the `.extractall()` function on the variable.

Activity: Zip-a-Dee-Doo-Dah

In this activity, you are given a password-protected zip file filled with music sheets and then will have to unpack the file using Python's zipfile module.

Activities/
06-Stu_ZipADeeDooDah/Unsolved/ZipADeeDooDah.py

Suggested Time:
10 minutes

Your Turn: Zip-a-Dee-Doo-Dah

Instructions:

- Create a Python application that takes the MusicSheets file and unzips it.
- The zip file IS password protected! ComicSans is the password.
- The script file will walk you step-by-step through the solution code you will need to write.

Times Up! Let’s Review.

Activity: XKCD Passwords

In this activity, you will create a robust random-password generator.

Activities/07-Stu_XKCDPassword/Unsolved/XKCDPassword.py
Activities/07-Stu_XKCDPassword/Unsolved/WordList.zip

Suggested Time:
40 minutes

Your Turn: XKCD Passwords

Instructions:

This application will require the `secrets`, `os`, and `zipfile` modules.

The following are three definitive parts to this application:

- The first part connects to the zip file provided and extracts all of its contents
- The second part builds a complete word list by looping through the text files that were originally stored within the zip file
- The third part randomly selects four words from the complete word list, puts them together into a single string, and then prints the result to the terminal.

Hints:

This is a challenging activity, and will likely take quite some time to finish. Feel free to work in groups!

Remember to test your application often to ensure it is working properly. Do not try and write the entire program in one go!

Parts of this code are similar to activities you have done previously. Feel free to look back at your previous code for ideas on how you could potentially solve this problem.

40 Minutes

Times Up! Let’s Review.

Class Objectives

Today we covered:

- ✓ Return a file's metadata using the `os.stat()` function
- ✓ Use functions to get the size of a file, when it was last accessed, and when it was last modified
- ✓ Convert dates and times from one format to another using the the datetime module
- ✓ Remove files from a system using the `os.remove()` function.
- ✓ Extract zip files using the zipfile module.

Kali Linux for Next Class

1) Kali.org/downloads/
2) Select Kali Linux 64-Bit or 32-Bit
   -Select from either first two images
   -Should download an ISO file
3) Setup in Virtual Box
4) Default login: ID: root Password: toor
5) Create snapshot
6) Change default password (look in your commands notebook)
7) From CLI, run: apt-get update after it is done, run apt-get upgrade. After all done, reboot Kali