



# Module 3 - System Administration Python

Session 4 - Modules and Introspection

**Presented by Tim Medin**

© SANS, Cyber Aces, Red Siege. All Rights Reserved. Redistribution Prohibited.

YOUR GATEWAY TO CYBERSECURITY SKILLS AND CAREERS

Welcome to the Module 3, System Administration. In this sub-section we'll be discussing Python. First, let's get you introduced to this scripting and programming language.

# SANS CYBER ACES ONLINE TUTORIALS

YOUR GATEWAY TO CYBERSECURITY SKILLS AND CAREERS

## 1. Introduction to Operating Systems

- 01. Linux
- 02. Windows

## 2. Networking

## 3. System Administration

- 01. Bash
- 02. PowerShell
- 03. Python

This training material was originally developed to help students, teachers, and mentors prepare for the Cyber Aces Online Competition. This module focuses on the basics of system administration and scripting. . This session is part of Module 3, System Administration. This module is split into three sections, Bash, PowerShell, and Python. In this session, we will continue our examination of Python.

The three modules of Cyber Aces Online are Operating Systems, Networking, and System Administration.

For more information about the Cyber Aces program, please visit the Cyber Aces website at <https://CyberAces.org/>.

## Module 3 - System Administration Python

- Syntax & Data Types
- Flow Control
- Building a Script
- Modules
- Practical Uses

In this session, you learn how to use modules in Python.



## Common Modules



There are many modules that come standard with python

- **sys** - System specific modules
- **os** - File system and OS functions

Install other modules "pip3", it is like "yum" but for python3

There are a number of useful modules that come preinstalled with python3. We've already used the "sys" module to access arguments in our scripts, but the module does a lot more. We'll look at it deeper shortly.

We can install other modules using "pip3". This functions much the same way as "yum", but this is specific to python3.



## pip3

5

We can search for modules with:

```
$ pip3 search search-text
```

Install a package:

```
$ pip3 install package-name
```

Uninstall

```
$ pip3 uninstall package-name
```

We can use pip3 to search for packages using "search".

```
$ pip3 search search-text
```

Install a package:

```
$ pip3 install package-name
```

Uninstall

```
$ pip3 uninstall package-name
```



## Introspection

6

It is a fancy, built-in "help me"

Display a list of all the variables and methods

**dir(x)**

Access Python's built-in documentation

**help(x)**

Identify the type of a variable

**type(variable)**

Introspection is a way to use python to learn how to use python. It is a fancy, built-in "help me". We can display a list of all the variables and methods with "dir":

**dir(x)**

Or use "help" to access Python's built-in documentation:

**help(x)**

Or identify the type of a variable:

**type(variable)**



## dir (module)

7

```
>>> import sys
>>> dir(sys)
['__displayhook__', '__doc__', '__excepthook__', '__interactivehook__',
 '__loader__', '__name__', '__package__', '__spec__', '__stderr__', '__stdin__',
 '__stdout__', '__clear_type_cache__', '__current_frames__', '__debugmallocstats__',
 '__getframe__', '__git__', '__home__', '__xoptions__', 'abiflags', 'api_version', 'argv',
 'base_exec_prefix', 'base_prefix', 'builtin_module_names', 'byteorder',
 'call_tracing', 'callstats', 'copyright', 'displayhook', 'dont_write_bytecode',
 'exc_info', 'excepthook', 'exec_prefix', 'executable', 'exit', 'flags',
 'float_info', 'float_repr_style', 'get_asyncgen_hooks',
 'getCoroutineWrapper', 'getallocatedblocks', 'getcheckinterval',
 'getdefaultencoding', 'getdlopenflags', 'getfilesystemencodeerrors',
 'getfilesystemencoding', 'getprofile', 'getrecursionlimit', 'getrefcount',
 'getsizeof', 'getswitchinterval', 'gettrace', 'hash_info', 'hexversion',
 'implementation', 'int_info', 'intern', 'is_finalizing', 'maxsize',
 'maxunicode', 'meta_path', 'modules', 'path',
 'path_importer_cache', 'platform', 'prefix', 'ps1', 'ps2',
 'set_asyncgen_hooks', 'set_coroutine_wrapper', 'setcheckinterval',
 'setdlopenflags', 'setprofile', 'setrecursionlimit', 'setswitchinterval',
 'settrace', 'stderr', 'stdin', 'stdout', 'thread_info', 'version',
 'version_info', 'warnoptions']
```

We can use dir on a module or an object (such as a string) to see the properties and methods we have available.

```
>>> import sys
>>> dir(sys)
['__displayhook__', '__doc__', '__excepthook__',
 '__interactivehook__', '__loader__', '__name__', '__package__',
 '__spec__', '__stderr__', '__stdin__', '__stdout__',
 '__clear_type_cache__', '__current_frames__', '__debugmallocstats__',
 '__getframe__', '__git__', '__home__', '__xoptions__', 'abiflags', 'api_version',
 'argv', 'base_exec_prefix', 'base_prefix', 'builtin_module_names',
 'byteorder', 'call_tracing', 'callstats', 'copyright', 'displayhook',
 'dont_write_bytecode', 'exc_info', 'excepthook', 'exec_prefix',
 'executable', 'exit', 'flags', 'float_info', 'float_repr_style',
 'get_asyncgen_hooks', 'getCoroutineWrapper', 'getallocatedblocks',
 'getcheckinterval', 'getdefaultencoding', 'getdlopenflags',
 'getfilesystemencodeerrors', 'getfilesystemencoding', 'getprofile',
 'getrecursionlimit', 'getrefcount', 'getsizeof', 'getswitchinterval',
 'gettrace', 'hash_info', 'hexversion', 'implementation', 'int_info',
 'intern', 'is_finalizing', 'maxsize', 'maxunicode', 'meta_path',
 'modules', 'path', 'path_hooks', 'path_importer_cache', 'platform',
 'prefix', 'ps1', 'ps2', 'set_asyncgen_hooks',
 'set_coroutine_wrapper', 'setcheckinterval', 'setdlopenflags',
 'setprofile', 'setrecursionlimit', 'setswitchinterval', 'settrace',
 'stderr', 'stdin', 'stdout', 'thread_info', 'version',
 'version_info', 'warnoptions']
```

This output can be a little difficult to read, but each item is in quotes. We can see a number of interesting methods, including "argv" that we used before.



## dir (object)



```
>>> s = 'something'  
>>> dir(s)  
['__add__', '__class__', '__contains__', '__delattr__', '__dir__',  
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__',  
'__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__',  
'__init_subclass__', '__iter__', '__le__', '__len__', '__lt__',  
'__mod__', '__mul__', '__ne__', '__new__', '__reduce__',  
'__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',  
'__sizeof__', '__str__', '__subclasshook__', 'capitalize',  
'casifold', 'center', 'count', 'encode', 'endswith', 'expandtabs',  
'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha',  
'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric',  
'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust',  
'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind',  
'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',  
'splitlines', 'startswith', 'strip', 'swapcase', 'title',  
'translate', 'upper', 'zfill']
```

We can use "dir" on objects too so we can see what actions we can take on or with the object.

```
>>> s = 'something'  
>>> dir(s)  
['__add__', '__class__', '__contains__', '__delattr__',  
'__dir__', '__doc__', '__eq__', '__format__', '__ge__',  
'__getattribute__', '__getitem__', '__getnewargs__',  
'__gt__', '__hash__', '__init__', '__init_subclass__',  
'__iter__', '__le__', '__len__', '__lt__', '__mod__',  
'__mul__', '__ne__', '__new__', '__reduce__',  
'__reduce_ex__', '__repr__', '__rmod__', '__rmul__',  
'__setattr__', '__sizeof__', '__str__',  
'__subclasshook__', 'capitalize', 'casifold', 'center',  
'count', 'encode', 'endswith', 'expandtabs', 'find',  
'format', 'format_map', 'index', 'isalnum', 'isalpha',  
'isdecimal', 'isdigit', 'isidentifier', 'islower',  
'isnumeric', 'isprintable', 'isspace', 'istitle',  
'isupper', 'join', 'ljust', 'lower', 'lstrip',  
'maketrans', 'partition', 'replace', 'rfind', 'rindex',  
'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',  
'splitlines', 'startswith', 'strip', 'swapcase', 'title',  
'translate', 'upper', 'zfill']
```

For example, we can see the "upper" method to convert the string to upper case.



## help



We can use "help" with an object, but we have to get the type first: `help(type(s))`  
Or we can use help with the module name as shown below

```
>>> import os
>>> help(os)
Help on module os:

NAME
    os - OS routines for NT or Posix depending on what system we're on.

DESCRIPTION
    This exports:
        - all functions from posix or nt, e.g. unlink, stat, etc.
        - os.path is either posixpath or ntpath
        - os.name is either 'posix' or 'nt'
        - os.curdir is a string representing the current directory (always '.')
        - os.pardir is a string representing the parent directory (always '..')
...trimmed for brevity...
```

We can use help to get a better understanding of how to use an object or module. To get help on an object, we need to get the type first.

```
>>> s = 'something'
>>> help(type(s))
```

However, with a module we can simply provide the module name.

```
>>> import os
>>> help(os)
Help on module os:
```

NAME

```
    os - OS routines for NT or Posix depending on what
    system we're on.
```

DESCRIPTION

This exports:

- all functions from posix or nt, e.g. unlink, stat, etc.
- os.path is either posixpath or ntpath
- os.name is either 'posix' or 'nt'



## Review Setup

10

You'll need internet access in your Linux VM

```
$ sudo dhclient ens33
```

You will need to install tools to build (compile) new python modules

```
$ sudo yum install python3-devel gcc
```

You'll need internet access in your Linux VM. You will need to obtain an IP address for your VM with the command below.

```
$ sudo dhclient ens33
```

You will then need to install some additional tools with this command:

```
$ sudo yum install python3-devel gcc
```



## Review

11

### Tasks:

- Install the "psutil" module
- Use the module to print:
  - CPU utilization percentage
  - Disk usage
  - Network connection

Note: You will have to launch python3 and pip3 as root

```
$ sudo python3
```

```
$ sudo pip3 <other commands here>
```

Your tasks are to do the following:

- Install the "psutil" module
- Use the module to print the following:
  - CPU utilization percentage
  - Disk usage
  - Network connections

Note: You will have to launch python3 and pip3 as root

```
$ sudo python3
```

```
$ sudo pip3 <other commands here>
```



## Answers

12

```
$ sudo pip3 install psutil
$ sudo python3
>>> import psutil
>>> help(psutil)
>>> print(psutil.cpu_percent())
6.4
>>> help(psutil.disk_usage)
>>> print(psutil.disk_usage('/'))
sdiskusage(total=18238930944, used=4290990080, free=13947940864,
percent=23.5)
>>> print(psutil.net_connections())
[sconn(fd=12, family=<AddressFamily.AF_INET: 2>,
...truncated for brevity...
```

You will need to dig through the help on the module to find the methods used here. You'll also need to look at the help to see how to use the "disk\_usage" method.

```
>>> import psutil
>>> help(psutil)
>>> print(psutil.cpu_percent())
6.4
>>> help(psutil.disk_usage)
>>> print(psutil.disk_usage('/'))
sdiskusage(total=18238930944, used=4290990080,
free=13947940864, percent=23.5)
>>> print(psutil.net_connections())
[sconn(fd=12, family=<AddressFamily.AF_INET: 2>,
...truncated for brevity...
```

## Module 3 - System Administration

### Python

- Syntax & Data Types
- Flow Control
- Building a Script



- Practical Uses

You've just learned how to use modules in Python. In the next module, we will conclude this module with practical uses in Python.