

# CompTIA PenTest+ Guide to Penetration Testing, 1e

## Module 13: Writing and Understanding Code

# Module Objectives (1 of 2)

By the end of this module, you should be able to:

1. Explain basic programming concepts
2. Describe common data structures
3. Write a simple C program
4. Create Bash scripts
5. Create PowerShell scripts
6. Explain how webpages are created with HTML

# Module Objectives (2 of 2)

By the end of this module, you should be able to:

7. Create basic Perl Programs
8. Explain basic object-oriented programming concepts
9. Create basic Python programs
10. Create basic Ruby programs
11. Create basic JavaScript programs
12. Describe some of the uses of programming in penetration testing

# Introduction to Computer Programming (1 of 8)

- Penetration testers should know how both threat actors and security professionals use computer programming
- Understanding exploit code will provide insight into threat actor intent
- Programming and scripting can help automate many tasks, saving time
- To be successful, programmers must understand programming language syntax, down to placement of semicolons and parentheses
- Many colleges still don't teach programming with security in mind

# Introduction to Computer Programming (2 of 8)

## Branching, Looping, and Testing

- Programming languages have methods of branching, looping, and testing (BLT)
- Functions are mini programs inside main program to carry out a task
- Branching takes operation from one area of program to another
- Looping - performing a task over and over
- Testing conducted on variables to evaluate whether a condition is true or false
- Looping commonly continues until testing condition indicates to stop

# Introduction to Computer Programming (3 of 8)

## Branching, Looping, and Testing

- Algorithms are listings of tasks necessary to solve a problem
- Programmers may use pseudocode, an English-like language to help develop program structure and algorithm
- Example: pseudocode algorithm to purchase BLT sandwich ingredients

```
PurchaseIngredients Function
```

```
    Call GetCar Function
```

```
    Call DriveToStore Function
```

```
    Purchase Bacon, Bread, Tomatoes, Lettuce, and Mayonnaise at store
```

```
End PurchaseIngredients Function
```

# Introduction to Computer Programming (4 of 8)

## Variables

- Places to store information of varying types within running program
- Complex variable types are sometimes referred to as data structures

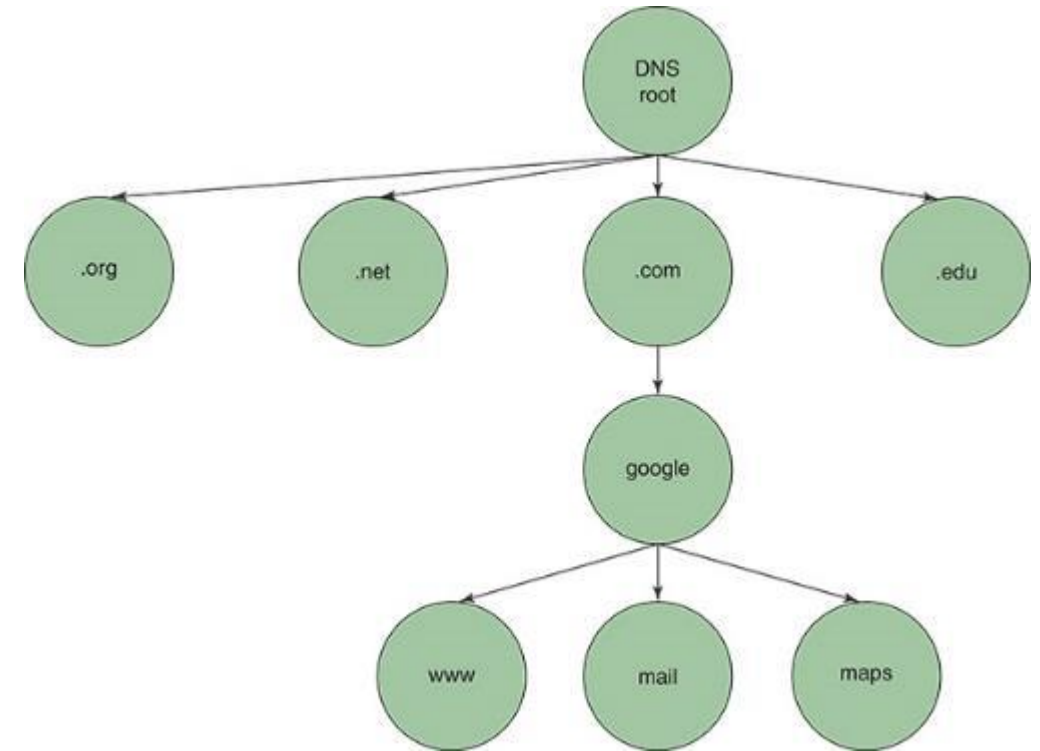
## Arrays

- Specific types of variable that can contain multiple values, such as numbers or strings
- Each variable within an array is stored in a unique location referenced with an index number that starts at 0

# Introduction to Computer Programming (5 of 8)

## Trees

- Trees are data structures that store data in a hierarchical fashion
- Root node starts each tree and has one or more child nodes



Google DNS information stored in a tree data structure



# Introduction to Computer Programming (6 of 8)

## Compiled versus Interpreted Programming Languages

- Compiled programming languages must have code processed into binary form before execution
- A compiler is a software tool that performs this task
- Compiling hides the source code which is a security advantage
- Many scripts are written in interpreted languages that are executed one line at a time

# Introduction to Computer Programming (7 of 8)

## Compiled versus Interpreted Programming Languages

### Compiled Languages

- C, C++, C#
- COBOL
- Java
- Pascal
- Visual Basic

### Interpreted Languages

- Bash
- JavaScript
- Perl
- Python
- PowerShell

# Introduction to Computer Programming (8 of 8)

## Documentation

- Documenting work during coding is a highly recommended best practice
- Makes code easier to understand or modify when used by others
- Access to source code documentation could be argued as providing a threat actor with useful info
  - Most agree that benefits outweigh risks of properly documented code

# Learning the C Language (1 of 5)

- The C programming language was developed by Dennis Ritchie at Bell Laboratories in 1972
- Compiled languages can be written in text editors or in compilers
- The C language is still widely used by security pros and hackers alike due to its power and cross-platform usability
- C source code must be compiled before it can be executed
- Microsoft Visual C++ Compiler, GNU C and C++ compilers, and Intel's C++ compilers are among most popular languages used today

# Learning the C Language (2 of 5)

## Anatomy of a C Program

- The “Hello, world!” program is one of first tasks C programmers learn

```
/* The famous "Hello, world!" C program */  
#include <stdio.h> /* Load the standard IO library.  
The library contains  
functions your C program might need to call to  
perform various tasks. */  
main()  
{  
printf("Hello, world!\n\n");  
}
```

# Learning the C Language (3 of 5)

## Declaring Variables

- C and most languages allow for assigning a type to variables
- Common C variables include Int, Float, Double, Char, String, and Const
  - Types represent format or size of potential variable
  - An Int can represent positive and negative integers
  - Float variables are assigned real numbers that include decimal point
  - Character and string variables can hold text values
  - Constant or Const are variables that do not change

# Learning the C Language (4 of 5)

## Branching, Looping, and Testing in C

- C implements branching, looping, and testing using a few techniques
- A `while` loop will repeat an action or actions until a condition is met

```
while (counter <= 10) //Do what's inside the braces until  
false
```

- The `do` loop performs action then tests to see if should continue
- A `for` loop starts by assigning a variable a value, lists condition to test until, and then an action to take while condition is still true

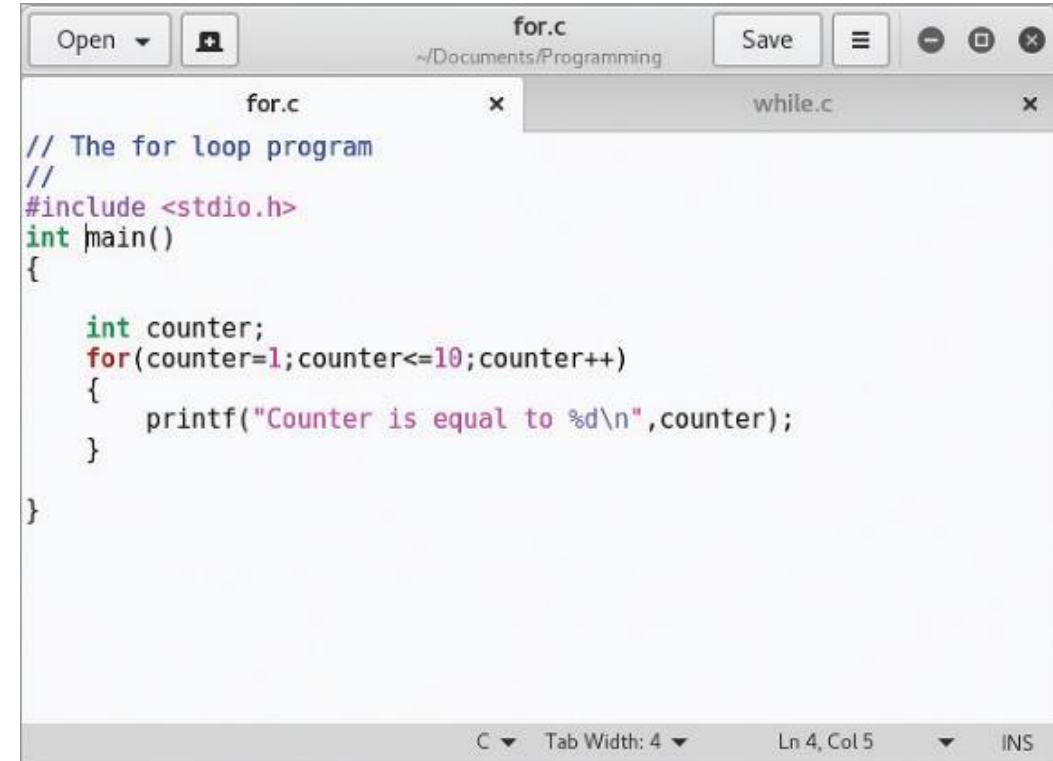
```
for(int counter=1;counter<=10;counter++);
```

# Learning the C Language (5 of 5)

## Branching, Looping, and Testing in C

This example does the following:

- Creates integer variable “counter” and assigns value of 1
- Prints the text specified
- Tests to see if counter <= 10
- If true, increments variable by 1 and prints text again
- If false, exits loop

A screenshot of a code editor window titled 'for.c' with a file path of '~\Documents\Programming'. The editor contains the following C code:

```
// The for loop program
//
#include <stdio.h>
int main()
{
    int counter;
    for(counter=1; counter<=10; counter++)
    {
        printf("Counter is equal to %d\n", counter);
    }
}
```

The code is syntax-highlighted. The editor interface includes 'Open', 'Save', and menu icons at the top, and a status bar at the bottom showing 'C', 'Tab Width: 4', 'Ln 4, Col 5', and 'INS'. A second tab titled 'while.c' is visible in the background.

for loop



# Understanding HTML Basics (1 of 2)

- Hypertext Markup Language (HTML) is primarily used for formatting and layout of web pages
- Understanding basic HTML syntax is important, as it is used in commonly seen attacks like hiding malicious links in phishing emails
- Ability to examine webpages to recognize suspicious HTML is beneficial
- HTML5 is current version and offers improvements over previous
  - Natively supports rich media elements such as audio and video
  - Eliminates need for some third-party media players like Flash

# Understanding HTML Basics (2 of 2)

## Creating a Webpage with HTML

- HTML can quickly and easily be written to create a basic text page

```
<!-- This is how you add a comment to an HTML webpage -->
```

```
<html>
```

```
<head>
```

```
<title>Hello, world--again</title>
```

```
</head>
```

```
<body>
```

This is where you put page text, such as marketing copy for an e-commerce business.

```
</body>
```

```
</html>
```

# Shells and Shell Scripts (1 of 8)

- Shells are command line environments for manual execution of commands
- Windows command line (cmd.exe) and Linux terminal are shells
- Shell scripting is the consolidation of commands into a script for a purpose such as command automation
- Shell scripting not as advanced as using a programming language like Python or Perl
- Supports command execution with ability to add some programming constructs and logical operations for input and output handling

# Shells and Shell Scripts (2 of 8)

## Bash and Bash Scripts

- Bourne-again shell (Bash) is default terminal for many Linux distributions
- Bash scripts can be written in text editor
  - Must begin with `#!/bin/bash` for first line
  - Filename should end in `."sh"`
  - Script files should be modified so are executable using `chmod u+x`

# Shells and Shell Scripts (3 of 8)

## Bash and Bash Scripts

### Bash Variables

- Bash does not have variable types as do other programming languages

### Bash Operators

- Bash allows many operators for testing conditions or performing math

### Testing Conditions in Bash

- For testing conditions, Bash uses an `if...then...else` structure

# Shells and Shell Scripts (4 of 8)

## Bash and Bash Scripts

### Looping in Bash

- Bash provides for looping using `for` and `while` loops, as C does

### Redirecting Standard Input and Output

- Command shell function allows command output to be saved to file rather than displayed on the terminal screen
- A command or script results or output can be directed to a specified output file easily

```
./helloWorld.sh > output.txt
```

# Shells and Shell Scripts (5 of 8)

## PowerShell and PowerShell Scripts

- PowerShell is a command shell developed by Microsoft for use in Windows
- Commands in PowerShell often called “cmdlets” on target systems
- PowerShell is used to execute commands locally or on remote systems
- PowerShell scripts can be written in text editor and executed in PowerShell terminal environment
- Originally designed for Windows, can now work on Mac and Linux

# Shells and Shell Scripts (6 of 8)

## PowerShell and PowerShell Scripts

- The hello, world program in PowerShell is: `Write-Host "Hello, world!"`
- To execute a script, the following command is used: `./helloWorld.ps1`
- The PowerShell command environment comes with security restrictions on the execution of scripts
  - A command may be necessary to allow scripts to execute and appropriate security and permission levels allowed



# Shells and Shell Scripts (7 of 8)

## PowerShell and PowerShell Scripts

### PowerShell Variables

- Variables work like other languages, but type is set automatically

### PowerShell Operators

- Many operators for testing conditions or performing math are available

### Testing Conditions in PowerShell

- Just like Bash, PowerShell uses an `if...then...else` structure

# Shells and Shell Scripts (8 of 8)

## PowerShell and PowerShell Scripts

### Looping in PowerShell

- PowerShell provides for looping using `for` and `while` loops, as C does

### Redirecting Standard Input and Output

- PowerShell allows command output to be saved to file rather than displayed on the terminal screen
- A command or script results or output can be directed to a specified output file easily

```
./helloWorld.ps1 > output.txt
```

# Discussion Activity 13-1

Shell scripting languages are powerful extensions of the capabilities of the shell or command environment being used. Bash and PowerShell scripting are presented in this module as a useful means of working within their respective command shells.

Think back to examples of specific tools used in pen testing for both Linux and Windows systems in the previous modules. What tools might be well-suited for executing from within shell scripts? Discuss possible use cases or scenarios for these tools being employed using shell scripts.

# Understanding Perl (1 of 3)

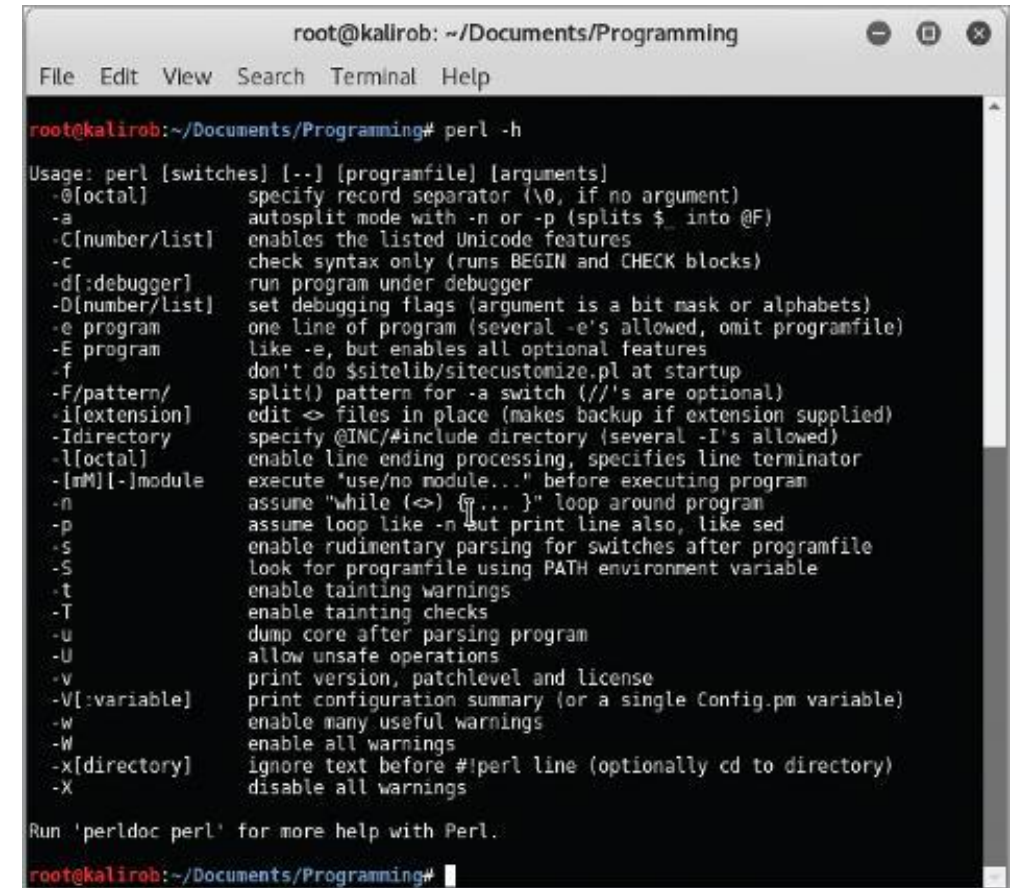
## Background on Perl

- The Practical Extraction and Report Language (Perl) is a powerful scripting language
- Perl was developed for \*nix platforms by Larry Wall in 1987
- Perl syntax is similar to C
- Perl 5.34 is the current stable version as of writing of this course
- Scripts in Perl can be written in a text editor and end with “.pl”
- Hackers have used Perl for automated exploits and bots
- System admin and security pros often use Perl for task automation

# Understanding Perl (2 of 3)

## Understanding the Basics of Perl

- Several excellent resources exist to assist with getting started and learning the Perl language



```
root@kalirob: ~/Documents/Programming
File Edit View Search Terminal Help

root@kalirob:~/Documents/Programming# perl -h

Usage: perl [switches] [--] [programfile] [arguments]
  -@[octal]      specify record separator (\0, if no argument)
  -a            autosplit mode with -n or -p (splits $_ into @F)
  -C[number/list] enables the listed Unicode features
  -c            check syntax only (runs BEGIN and CHECK blocks)
  -d[:debugger] run program under debugger
  -D[number/list] set debugging flags (argument is a bit mask or alphabets)
  -e program     one line of program (several -e's allowed, omit programfile)
  -E program     like -e, but enables all optional features
  -f            don't do $sitelib/sitecustomize.pl at startup
  -F/pattern/    split() pattern for -a switch (///'s are optional)
  -i[extension] edit <=> files in place (makes backup if extension supplied)
  -Idirectory    specify @INC/#include directory (several -I's allowed)
  -l[octal]      enable line ending processing, specifies line terminator
  -[mM][...]module execute 'use/no module...' before executing program
  -n            assume "while (<=) { ... }" loop around program
  -p            assume loop like -n but print line also, like sed
  -s            enable rudimentary parsing for switches after programfile
  -S            look for programfile using PATH environment variable
  -t            enable tainting warnings
  -T            enable tainting checks
  -u            dump core after parsing program
  -U            allow unsafe operations
  -v            print version, patchlevel and license
  -V[:variable] print configuration summary (or a single Config.pm variable)
  -w            enable many useful warnings
  -W            enable all warnings
  -x[directory] ignore text before #!perl line (optionally cd to directory)
  -X            disable all warnings

Run 'perldoc perl' for more help with Perl.

root@kalirob:~/Documents/Programming#
```

Perl help information

# Understanding Perl (3 of 3)

## Understanding the BLT of Perl

### Branching in Perl

- Perl has a syntax for branching that is very similar to C language

### Looping in Perl

- A Perl `for` loop is identical to the `for` loop in C

### Testing Conditions in Perl

- Standard operators for testing conditions in Perl are very similar or identical to operators used in the C language

# Understanding Object-Oriented Programming Concepts

- Object-oriented programming is a way of developing programs that revolutionizes the way programs can be created and used
- Some traditional scripting languages can now use object-oriented programming concepts as well
- A “class” is a structure that holds pieces of data and functions
- Classes are written in many object-oriented languages

# Understanding Python (1 of 9)

## Background on Python

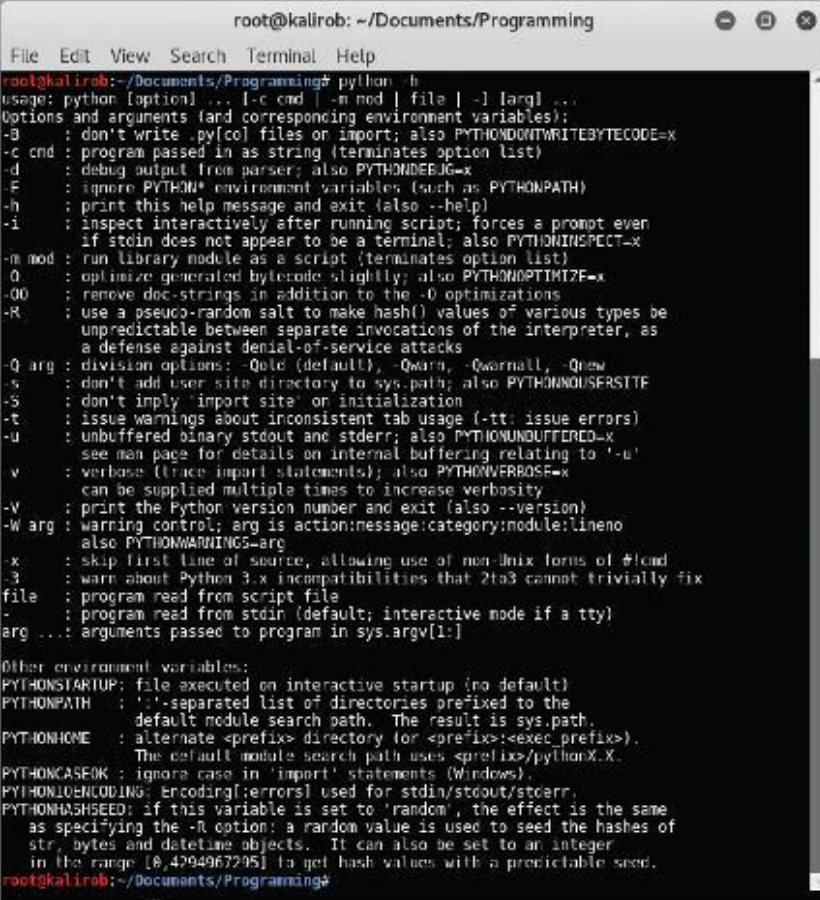
- Python was named after Monty Python's Flying Circus TV comedy show
- Python is a scripting language with object-oriented features
- Popular for creating small- to medium-sized programs quickly
- Designed for easy readability, requiring indentation to define code blocks
  - Syntax errors can be created by having improper spacing
- Implementation of Python began in December 1989
- Cross platform; capable of development and use on most any OS



# Understanding Python (2 of 9)

## Understanding the Basics of Python

- The `python -h` command will display parameters used with the `python` command
- Programs written in python will end in a `.py`



```
root@kalirob: ~/Documents/Programming
File Edit View Search Terminal Help
root@kalirob:~/Documents/Programming# python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-B      : don't write .py[co] files on import; also PYTHONDOTWRITEBYTECODE=x
-c cmd  : program passed in as string (terminates option list)
-d      : debug output from parser; also PYTHONDEBUG=x
-E      : ignore PYTHON* environment variables (such as PYTHONPATH)
-h      : print this help message and exit (also --help)
-i      : inspect interactively after running script; forces a prompt even
        : if stdin does not appear to be a terminal; also PYTHONINSPECT=x
-m mod  : run library module as a script (terminates option list)
-O      : optimize generated bytecode slightly; also PYTHONOPTIMIZE=x
-OO     : remove doc-strings in addition to the -O optimizations
-R      : use a pseudo-random salt to make hash() values of various types be
        : unpredictable between separate invocations of the interpreter, as
        : a defense against denial-of-service attacks
-q arg  : division options: -Qold (default), -Qwarn, -Qwarnall, -Qnew
-s      : don't add user site directory to sys.path; also PYTHONNOUSERSITE
-S      : don't imply 'import site' on initialization
-t      : issue warnings about inconsistent tab usage (-tt: issue errors)
-u      : unbuffered binary stdout and stderr; also PYTHONUNBUFFERED=x
        : see man page for details on internal buffering relating to '-u'
-v      : verbose (trace import statements); also PYTHONVERBOSE=x
        : can be supplied multiple times to increase verbosity
-V      : print the Python version number and exit (also --version)
-W arg  : warning control; arg is action:message:category:module:lineno
        : also PYTHONWARNINGS=arg
-x      : skip first line of source, allowing use of non-Unix forms of #cmd
-3      : warn about Python 3.x incompatibilities that 2to3 cannot trivially fix
file    : program read from script file
-       : program read from stdin (default; interactive mode if a tty)
arg ... : arguments passed to program in sys.argv[1:]

Other environment variables:
PYTHONSTARTUP: file executed on interactive startup (no default)
PYTHONPATH   : ':'-separated list of directories prefixed to the
        : default module search path. The result is sys.path.
PYTHONHOME   : alternate <prefix> directory (or <prefix>:<exec prefix>).
        : The default module search path uses <prefix>/pythonX.X.
PYTHONCASEOK : ignore case in 'import' statements (Windows).
PYTHONIOENCODING: encoding[:errors] used for stdin/stdout/stderr.
PYTHONHASHSEED: if this variable is set to 'random', the effect is the same
        : as specifying the -R option: a random value is used to seed the hashes of
        : str, bytes and datetime objects. It can also be set to an integer
        : in the range [0,4294967295] to get hash values with a predictable seed.
root@kalirob:~/Documents/Programming#
```

Python help information

# Understanding Python (3 of 9)

## Understanding the BLT of Python

- Several important syntax rules are important to remember for Python:
  - Spacing is important; improper spacing will cause syntax errors
  - Variables do not begin with a special symbol
  - No special characters are required at the end of a line of code
  - Comment lines begin with the # symbol

# Understanding Python (4 of 9)

## Understanding the BLT of Python

### Branching in Python

- Functions are mini programs inside main program to carry out a task
- To jump from one function to another, call the function by entering its name, followed by function brackets
- A function must be defined before it can be called
- Functions can take parameters

# Understanding Python (5 of 9)

## Understanding the BLT of Python

### Looping in Python

- The Python `for` and `while` loops use different syntax than Perl or C
- A `for` loop will repeat until it has gone through a list of specified items

```
names = ["Bob", "Jamal", "Sasha"]  
for x in names:  
    print(x)
```

# Understanding Python (6 of 9)

## Understanding the BLT of Python

### Testing Conditions in Python

- The standard Python operators are identical to the operators in Perl

### If Statements and Logical Operators

- Python uses comparison or logical operators: `if`, `else`, `elif` and others
- Code will perform action only when the `if` condition is true; otherwise, the `else` case will be executed
- An `elif` will allow for nesting conditional operators

# Understanding Python (7 of 9)

## Understanding the BLT of Python

### If Statements and Logical Operators

- Remember indentation and spacing is critical in Python programming

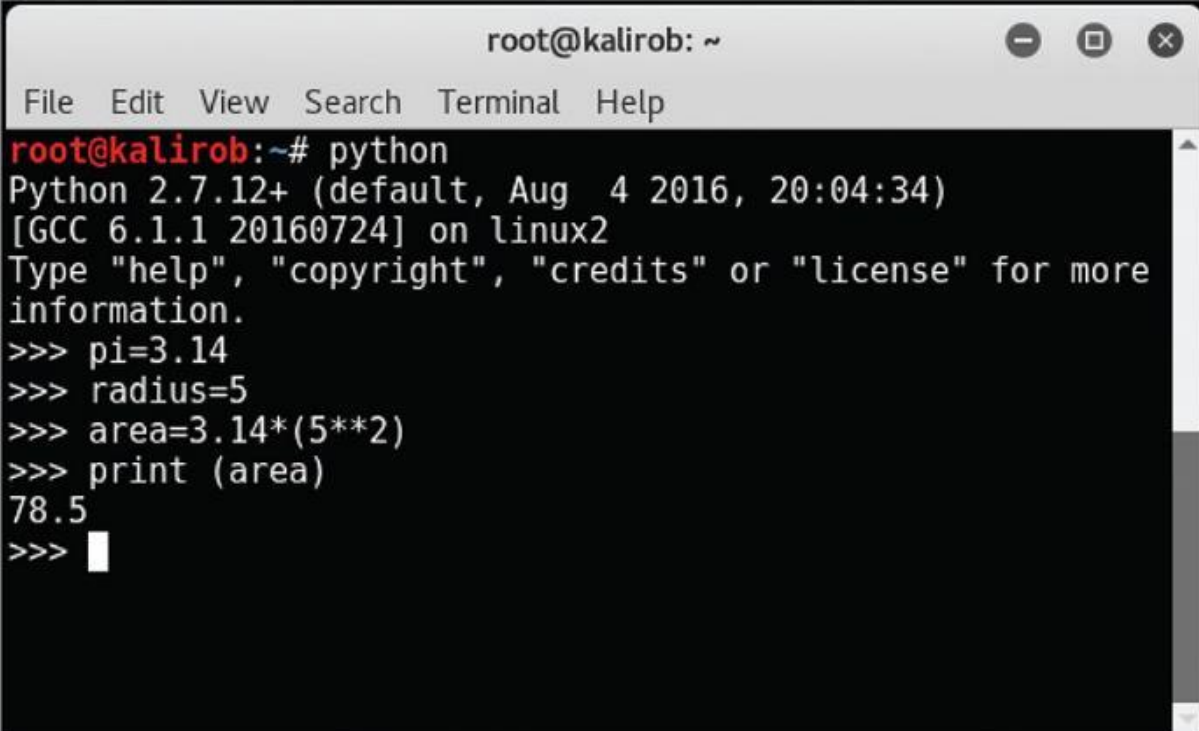
```
if (age) > 12
    print ("You must be a know-it-all!")

else
    print ("Sorry, but I don't know why the sky is blue.")
```

# Understanding Python (8 of 9)

## The Python Shell

- Python environment has an interactive shell which allows for entering commands and their immediate execution
- Python shell is known as REPL
  - Read, Evaluate, Print, Loop



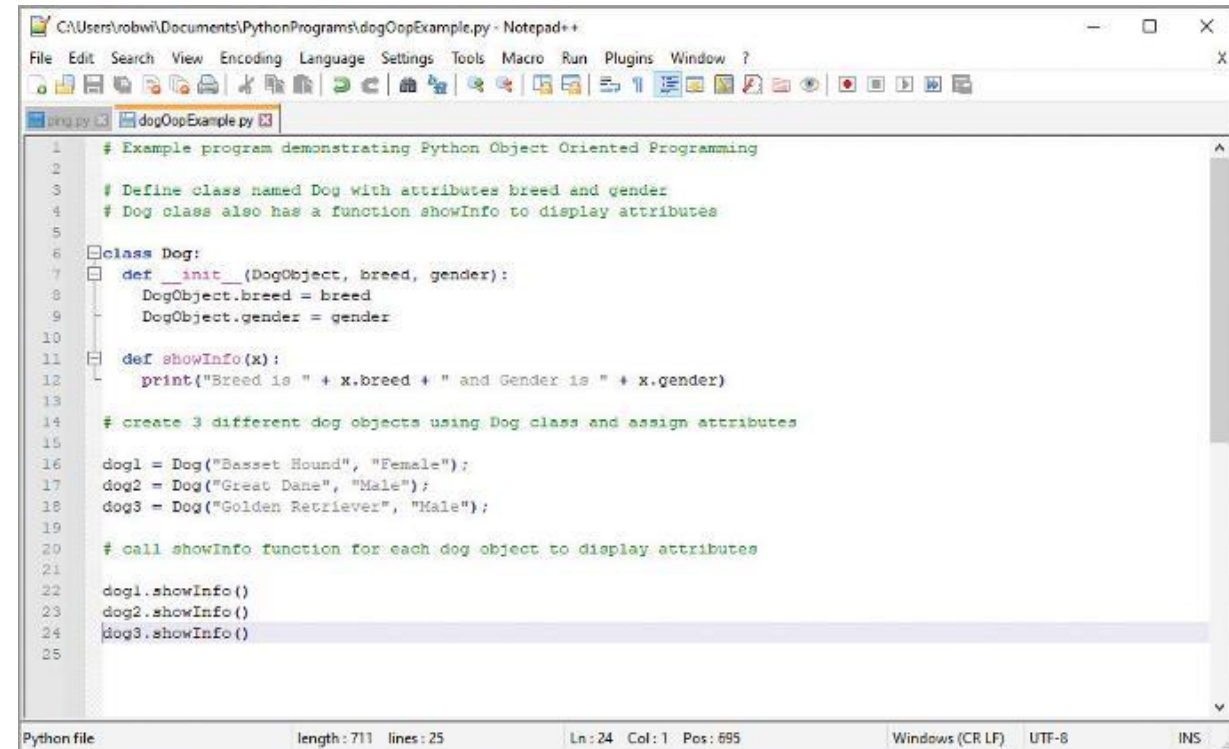
```
root@kalirob: ~
File Edit View Search Terminal Help
root@kalirob:~# python
Python 2.7.12+ (default, Aug  4 2016, 20:04:34)
[GCC 6.1.1 20160724] on linux2
Type "help", "copyright", "credits" or "license" for more
information.
>>> pi=3.14
>>> radius=5
>>> area=3.14*(5**2)
>>> print (area)
78.5
>>> 
```

Python shell (REPL)

# Understanding Python (9 of 9)

## Object-Oriented Programming in Python

- Python uses object-oriented programming (OOP) which is outside of scope of this course
- Example program shows a class called “Dog” which has three attributes associated with it
- This construct is foundation of object in OOP



```
1 # Example program demonstrating Python Object Oriented Programming
2
3 # Define class named Dog with attributes breed and gender
4 # Dog class also has a function showInfo to display attributes
5
6 class Dog:
7     def __init__(DogObject, breed, gender):
8         DogObject.breed = breed
9         DogObject.gender = gender
10
11     def showInfo(x):
12         print("Breed is " + x.breed + " and Gender is " + x.gender)
13
14 # create 3 different dog objects using Dog class and assign attributes
15
16 dog1 = Dog("Basset Hound", "Female");
17 dog2 = Dog("Great Dane", "Male");
18 dog3 = Dog("Golden Retriever", "Male");
19
20 # call showInfo function for each dog object to display attributes
21
22 dog1.showInfo()
23 dog2.showInfo()
24 dog3.showInfo()
25
```

dogOopExample.py program

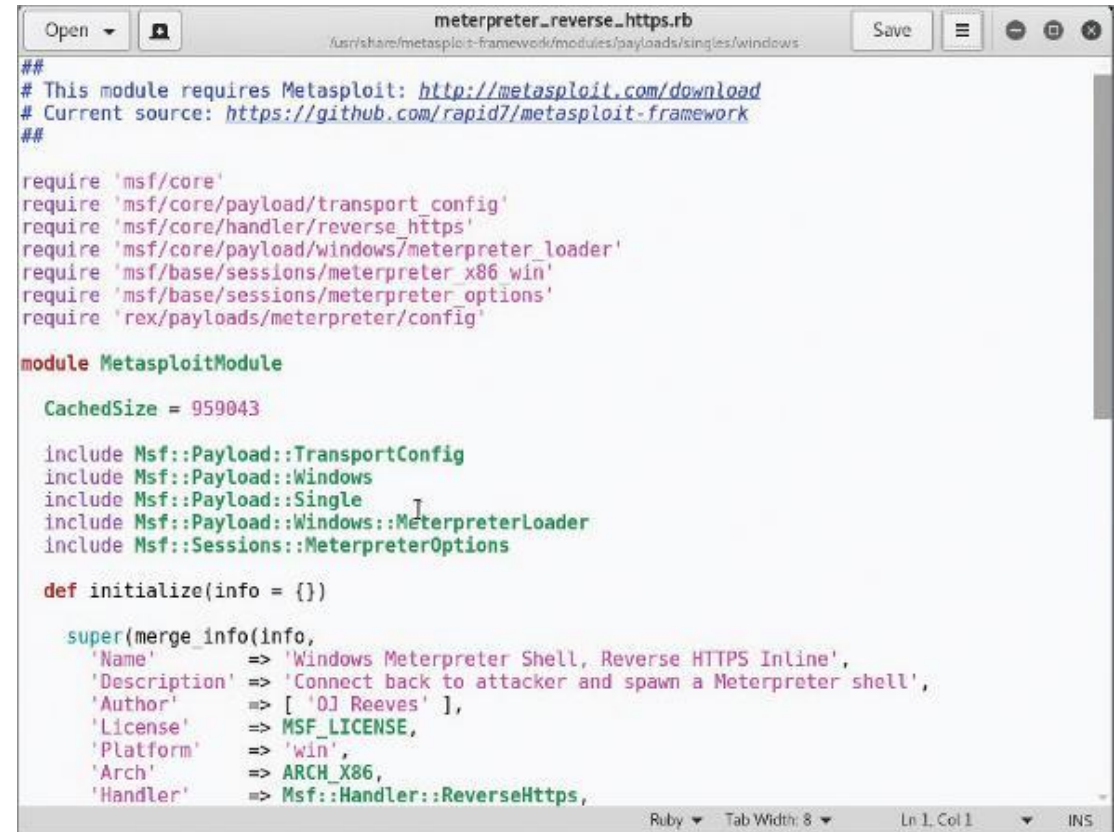


# Understanding Ruby (1 of 8)

- Many pen testers use the interpreted language Ruby with object-oriented programming features
- Ruby is similar to Perl
- Simple scripts can be written in Ruby without use of OOP
- The Metasploit application is a Ruby program
- Ruby can be used to develop exploits using Metasploit Framework

# Understanding Ruby (2 of 8)

- Due to the prevalence of Metasploit in penetration testing, Ruby is a useful language to understand
- Ruby programs will use the .rb file extension



```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'
require 'msf/core/payload/transport_config'
require 'msf/core/handler/reverse_https'
require 'msf/core/payload/windows/meterpreter_loader'
require 'msf/base/sessions/meterpreter_x86_win'
require 'msf/base/sessions/meterpreter_options'
require 'rex/payloads/meterpreter/config'

module MetasploitModule

  CachedSize = 959043

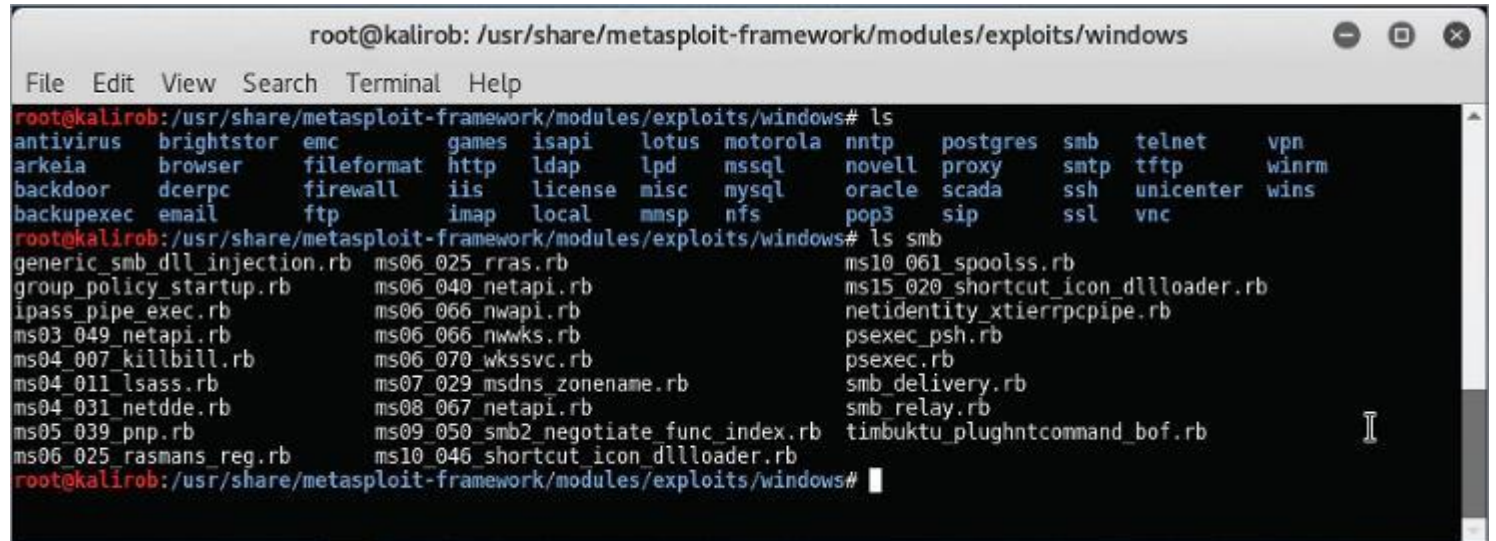
  include Msf::Payload::TransportConfig
  include Msf::Payload::Windows
  include Msf::Payload::Single
  include Msf::Payload::Windows::MeterpreterLoader
  include Msf::Sessions::MeterpreterOptions

  def initialize(info = {})
    super(merge_info(info,
      'Name' => 'Windows Meterpreter Shell, Reverse HTTPS Inline',
      'Description' => 'Connect back to attacker and spawn a Meterpreter shell',
      'Author' => [ '0J Reeves' ],
      'License' => MSF_LICENSE,
      'Platform' => 'win',
      'Arch' => ARCH_X86,
      'Handler' => Msf::Handler::ReverseHttps,
    ))
  end
end
```

Metasploit meterpreter\_reverse\_https.rb program

# Understanding Ruby (3 of 8)

- Ruby can be used to develop exploits using Metasploit Framework



```
root@kalirob: /usr/share/metasploit-framework/modules/exploits/windows
File Edit View Search Terminal Help
root@kalirob:/usr/share/metasploit-framework/modules/exploits/windows# ls
antivirus  brightstor  emc      games  isapi  lotus  mozilla  nntp  postgres  smb  telnet  vpn
arkeia    browser    fileformat  http  ldap  lpd  mssql  novell  proxy  smtp  tftp  winrm
backdoor  dcerpc    firewall  iis    license  misc  mysql  oracle  scada  ssh  unicenter  wins
backupexec  email    ftp      inap  local  mmsp  nfs    pop3  sip    ssl  vnc
root@kalirob:/usr/share/metasploit-framework/modules/exploits/windows# ls smb
generic_smb_dll_injection.rb  ms06_025_rras.rb  ms10_061_spoolss.rb
group_policy_startup.rb      ms06_040_netapi.rb  ms15_020_shortcut_icon_dllloader.rb
ipass_pipe_exec.rb          ms06_066_nwapi.rb  netidentity_xtierpcpipe.rb
ms03_049_netapi.rb          ms06_066_nwks.rb  psexec_psh.rb
ms04_007_killbill.rb        ms06_070_wkssvc.rb  psexec.rb
ms04_011_lsass.rb           ms07_029_msdns_zonename.rb  smb_delivery.rb
ms04_031_netdde.rb          ms08_067_netapi.rb  smb_relay.rb
ms05_039_pnp.rb             ms09_050_smb2_negotiate_func_index.rb  timbuktuplugntcommand_bof.rb
ms06_025_rasmans_reg.rb     ms10_046_shortcut_icon_dllloader.rb
root@kalirob:/usr/share/metasploit-framework/modules/exploits/windows#
```

Metasploit exploits written in Ruby

# Understanding Ruby (4 of 8)

## Background on Ruby

- Ruby was conceived by Yukihiro Matsumoto (a.k.a “Matz”) in 1993
- Ruby is a high-level language designed with an emphasis on programming productivity and simplicity

## Understanding the Basics of Ruby

- Code written in Ruby can be executed using the `ruby` command
- Help for the command can be found by entering in `ruby -h`

# Understanding Ruby (5 of 8)

## Understanding the BLT of Ruby

- Several important syntax rules are important to remember for Ruby:
  - Spacing and indentation do not matter
  - Variables do not begin with a special symbol
  - No special characters are required at the end of a line of code
  - Comment lines begin with the # symbol

# Understanding Ruby (6 of 8)

## Understanding the BLT of Ruby

### Branching in Ruby

- Functions are mini programs inside main program to carry out a task
- To jump from one function to another, call the function by entering its name, followed by function brackets
- A function must be defined before it can be called
- Functions can take parameters

# Understanding Ruby (7 of 8)

## Understanding the BLT of Ruby

### Looping in Ruby

- Ruby has `for` and `while` loops that use a different syntax than Perl or C
- A `for` loop will repeat until it has gone through a list of specified items

```
names = ["Larry", "Moe", "Curly"]  
for x in names do  
    print(x + "\n")  
end
```

# Understanding Ruby (8 of 8)

## Understanding the BLT of Ruby

### Testing Conditions in Ruby

- The standard Ruby operators are identical to the operators in Python

### If Statements and Logical Operators

- Ruby uses comparison or logical operators: `if`, `else`, `elsif` and others
- Code will perform action only when the `if` condition is true
- Otherwise the `else` case will be executed
  - An `elsif` will allow for nesting conditional operators



# Understanding JavaScript (1 of 7)

- JavaScript is a core technology for creating and managing websites
- 98% of websites use JavaScript to manage or render website content
- Threat actors commonly use or manipulate JavaScript in attacks

## Background on JavaScript

- JavaScript has existed since the first graphical web browser, Mosaic, was released in 1993
- JavaScript allowed dynamic changing webpage content for the first time

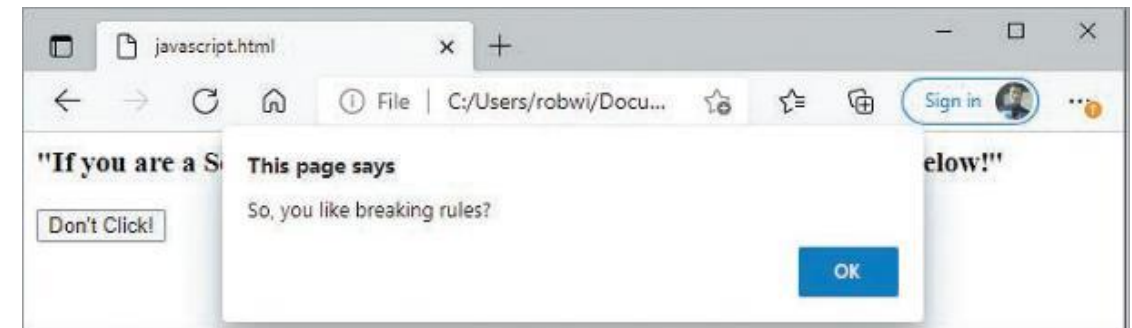
# Understanding JavaScript (2 of 7)

## Understanding the Basics of JavaScript

- A simple JavaScript code embedded in an HTML document can create buttons that trigger actions
- JavaScript is understood by web browsers, so no compiler or interpreter is needed
- The Nodejs tool allows executing JavaScript in Linux terminal shell



javascript.html in a browser



Clicking the button

# Understanding JavaScript (3 of 7)

## Understanding the BLT of JavaScript

- Several important syntax rules are important to remember for JavaScript:
  - Use the keyword `function` before a function's name to create it
  - Semicolon is used to mark the end of a line of code
  - Variables do not begin with a special symbol
  - Variables must be declared using the `var` or `let` keywords
  - Comment lines begin with a double forward slash `//`

# Understanding JavaScript (4 of 7)

## Understanding the BLT of JavaScript

### Branching in JavaScript

- To jump from one function to another, call the function by entering its name, followed by parentheses
- A function must be defined before it can be called
- Functions can take parameters

# Understanding JavaScript (5 of 7)

## Understanding the BLT of JavaScript

### Looping in JavaScript

- JavaScript has `for` and `while` loops that use a slightly different syntax than Perl or C
- A `for` loop will repeat until it has gone through a list of specified items

```
for (var i = 0; i < 10; i++) {  
  console.log (i);  
}
```
- The `console.log` function is commonly used to control output in JavaScript

# Understanding JavaScript (6 of 7)

## Understanding the BLT of JavaScript

### Testing Conditions in JavaScript

- The standard JavaScript operators are identical to the operators in Perl

### If Statements and Logical Operators

- JavaScript uses comparison or logical operators: `if`, `else`, and `else if`
- Code will perform action only when the `if` condition is true
- Otherwise the `else` case will be executed
  - An `else if` will allow for testing multiple conditions

# Understanding JavaScript (7 of 7)

## JavaScript Object Notation

- JavaScript Object Notation (JSON) is a data structure and format for exchanging information between web servers and browsers
- JSON uses a key/value pairing paradigm where key is variable name and value is what is assigned to the key

```
{  
    "system" : {  
        "hostname" : "arninstructor1.williscollege.com",  
        "ip" : "10.20.0.2",  
        "scanned" : true  
    }  
}
```

# Discussion Activity 13-2

In this module, both compiled and interpreted programming languages are introduced. Branching, looping, and testing using these languages is one common capability. The exact syntax and command structure for BLT differs from one language to another. One may argue that the underlying process by which each uses branching, looping and testing are fundamentally similar.

Discuss other similarities and differences between the programming languages covered in this module.



# Analyzing and Automating (1 of 5)

Two primary reasons pen testers benefit from learning programming are covered in this module:

1. Analyzing exploit code
2. Automating pen testing activities

## Analyzing Exploit Code

- Understanding what exploit code is attempting to do may develop pen testing ideas and help identify vulnerabilities
- Understanding Metasploit exploit code may allow for customizing the exploit code to more effectively use it in a pen test

# Analyzing and Automating (2 of 5)

## Analyzing Exploit Code

Exploit code commonly performs three activities

### 1. Enumeration

- Recognizing enumeration tactics can assist with catching attempts at exploit code

### 2. Downloading files or malicious payloads

- Catching exploits reaching to internet hosts can allow them to be blocked

### 3. Remote access connections

- Catching remote access attempts in exploits can prompt tester to check and recommend specific security countermeasures

# Analyzing and Automating (3 of 5)

## Using Code to Automate Penetration Tests

Automating pen testing is helpful in three areas:

### 1. Scanning

- Writing scripts can allow for complex tests and scanning automation

### 2. Target system configuration analysis

- Checking a remote system for OS type may invoke logic to follow up with OS specific vulnerability checks or launching exploits

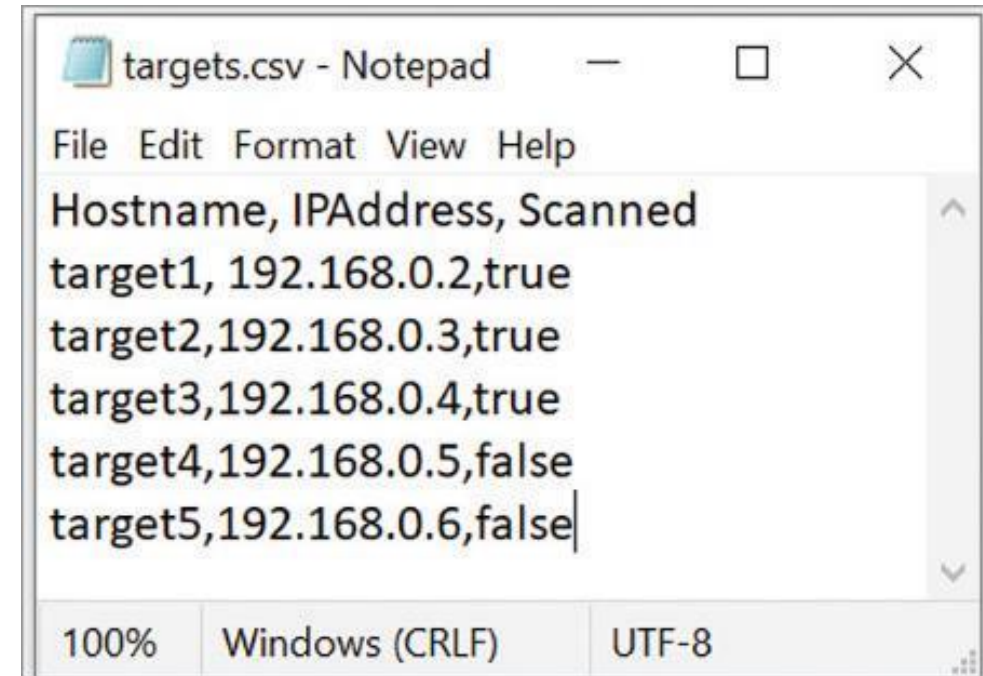
### 3. Modifying IP addresses during routine activities

- Looping through ranges of IP addresses while performing pen-testing activities can eliminate manual command execution

# Analyzing and Automating (4 of 5)

## Using Comma-Separated Value Files

- Comma-separated values (CSV) files contain lists of values separated by a comma with new records on separate lines
- Many tools or applications may generate data and output in CSV format
- Using a CSV as input for an automation script can save time by eliminating need to manually enter command parameters



CSV file targets.csv

# Analyzing and Automating (5 of 5)

## Reusing Code

- When a useful collection of code is created, developers may share it so it can be used by other programs or programmers

Reusing code may eliminate the need to create new instances of:

### 1. Classes

- OOP classes contain data and code for creating specific objects

### 2. Functions, procedures, and methods

- Existing functions may accomplish useful tasks

### 3. Libraries and modules

- Collections of other functions or classes packaged for easy reuse

# Discussion Activity 13-3

Learning to program or write scripts to automate pen-testing activities has the potential for saving pen testers time and effort. In addition to decreasing work required by automation, what other possible benefits might be realized by automating tasks using programs or scripts?

# Summary (1 of 2)

By the end of this module, you should be able to:

1. Explain basic programming concepts
2. Describe common data structures
3. Write a simple C program
4. Create Bash scripts
5. Create PowerShell scripts
6. Explain how webpages are created with HTML

# Summary (2 of 2)

By the end of this module, you should be able to:

- 7. Create basic Perl Programs
- 8. Explain basic object-oriented programming concepts
- 9. Create basic Python programs
- 10. Create basic Ruby programs
- 11. Create basic JavaScript programs
- 12. Describe some of the uses of programming in penetration testing