Revised 8/2023

NORTHERN VIRGINIA COMMUNITY COLLEGE
ITP 270 – PROGRAMMING FOR CYBERSECURITY (4 CR.)

Course Description

Teaches scripting and software development techniques for automating security tasks such as network monitoring and penetration testing using Python. Additional topics include writing custom tools and the basics of developing software exploits. Lecture 4 hours per week.

General Course Purpose

This course provides students with the knowledge to develop and maintain effective programs/scripts, through demonstrations and hands-on labs. Students will learn to create high-level language scripts and programs using methods, modules, classes, and other pertinent parts of the object-orientated and structured paradigms. Students will learn how to use Python to manipulate data, automate tasks, perform error handling, storing, retrieving and usefulness in a variety in cybersecurity areas. The class will provide hands on exercises to reinforce learning and develop real competency, as students are guided through the process of developing and testing security tools. To conclude the course, students will participate in a capstone project to fortify knowledge and skills gained during the course. This course can be used to satisfy the programming or security elective in the Cybersecurity AAS.

Course Prerequisites/Corequisites

Prerequisite: ITP 100

Course Objectives

Upon successful completion of this course, the student will be able to:

- Design code, test, and implement Python programs using both console and Graphical User Interface (GUI) applications.
- Demonstrate effective knowledge in and use of language syntax, tools, models and idiom.
- Demonstrate mastery of skills necessary to construct software solutions to a variety of security areas.
- Demonstrate use and implementation of commonly used algorithms and data structures as they relate to problems in Cybersecurity.
- Code with fluency in the object-oriented paradigm.
- Identify and explain the libraries available for attack task automation.
- Use the high-level programming language to analyze & debug live applications.
- Explain the ethical and social implications of hacking within the context of cybersecurity.

Major Topics to be included

- Python development environment
- Python data types
  - Strings and Numbers
  - Lists, Tuplies, Dictionaries
- Python basic syntax
  - Sequence, Decision making, loops
  - Functions and Modules
- Exception Handling

- Regular Expressions
- Classes and Objects
  - Inheritance
- File I/O and Database Manipulation
- Networking
- GUI Development

<u>Student Learning Outcomes</u>

I. Python development environment
   a. Review, install, configure, and test (if necessary) the Python 3 interpreters, compiler, PIP3 (the Python package manager), pathing, and other packages/libraries as appropriate.
   b. Write functional code stubs using IDLE3, the interactive command interpreter.

II. Python data types
   a. Strings and Numbers
      (Write code statements that...)
      i. initialize and use Python's Numbers type -- int, long, float, complex – to perform arithmetic, comparison, and assignment operations.
      ii. initialize strings, and uses to concatenation operator to combine them and the slice operator to extract substrings.
      iii. use the comparison string comparison operators to test strings for identity and difference.
      iv. format strings
         1. Operators are*: %c,%s,%d,%x,%e,%f*
      v. Use string functions, such as *find()*, *index()*, *isalnum()*.
   b. Lists, Tuplies, Dictionaries
      (Write code statements that...)
      i. create and initialize a list, then store data to its elements; and subsequently manipulate them.
      ii. create and initialize a tuple. Explain the difference between tuples and lists, and in what cases the read only tuple is preferable.
      iii. create and initialize a dictionary, then demonstrate its use as a hashtable for storing and retrieving data.

III. Python basic syntax
   a. Sequence, Decision making, loops
      (Write code statements that...)
      i. demonstrate the use of commenting for program clarity, and that use proper line indentation to indicate code block subordination.
      ii. employ proper quotation -- ('), double ("), and triple ('' or "") -- for the appropriate circumstance
      iii. use the input and output commands -- *input()*, *print()* – as well as the command line to insert data into an application as required.
      iv. use *if*, *if else*, construct to alter the flow of control in a program based on a Boolean condition.
      v. implement a modular "switch" statement in Python using *def* and a dictionary.
      vi. use the immediately proceeding statements in a Suite.

- vii. See II. a. 1.
- viii. Operators
  (…use the following operators…)
  - 1. Arithmetic
    - a. See II. a. 1.
    - b. Operators are: +, -, *, /, **, %, // (floor division)
  - 2. Comparison
    - a. See II. a. 1. And II. b. 1.
    - b. Operators are: ==,!=,<>, <.>,>=,<=
  - 3. Assignment
    - a. Operators are: =,+=,-=,*=,/+,%=,**=,//=
  - 4. Bitwise
    - a. Operators are: &, |, ^, ~,<<,>>
  - 5. Logical operators
    - a. Operators are: *and*, *or*, *not* .
  - 6. Membership
    - a. in a sequence (string, list, tuple)
    - b. Operators are: in, not in
  - 7. Identity
    - a. Operators are: *is*, *is not.*
- ix. Loops
  - 1. use a *for* loop for iteration
  - 2. use a *while* loop where initial Boolean state changes
  - 3. Loop control statements
    - a. *Break*, *continue*, *pass*.

- b. Functions and Modules
  (Write code statements that…)
  - i. define the signature and body of a function taking 0, 1, and 2 parameters of differing types
  - ii. demonstrate calling functions with varying signatures
  - iii. demonstrate passing by reference
  - iv. demonstate variable scope
  - v. deploy functions in various files and combine them using the import command.

IV. Exception Handling
(Write code statements that…)
- a. uses the *assert* command to trap and recover from error conditions.
- b. uses a *try-except-else* and *finally* block to handle errors.

V. Regular Expressions
(Write code statements that…)
- a. use characters, character classes, and control characters to create patterns.
  - i. Control characters: (+ ? . * ^ $ ( ) [ ] { } | \ )
- b. demonstrate the use of matching, searching, and replacing patterns of interest.

VI. Classes and Objects

    a. demonstrate competency in Object Oriented terminology and concepts.

    (Write code statements that...)

    b. use OOP mechanics to create classes and instatiatie objects of those classes.
    c. use simply inheritance to simplify code.

VII. File I/O and Database Manipulation
(Write code statements that...)

    a. facilitates opening, reading, writing, and saving text files.
        i. Uses exception handling to test opening and saving files
    b. uses commands to navigate in code through file system.
    c. uses standard database interfaces and basic SQL to create, open, read from, write to (add and update), search for, and delete entries in a simple relational database.

VIII. Networking
(Write code statements that...)

    a. use socket module to create a simple client server communication application.

IX. GUI Development
(Write code statements that...)

    a. use the Tk GUI toolkit (Tkinter) to develop simple but functional GUI interfaces.

Recommended Demonstration Topics and Projects

I. Reconnaissance
    a. Web scraping
    b. Query apps, web crawling

II. Scanning
    a. Packet sniffing
    b. Replacing netcat
    c. Bot detection

III. Access
    a. HTML form authentication
    b. Replacing scapy
    c. Keylogging
    d. Pythonic Shellcode execution

IV. Maintaining Access
    a. Windows privilege escalation
    b. SSH tunneling

V. Covering Tracks
    a. Log file manipulation

In this course, the following VCCS General Education Outcomes are supported:

- Understand and interpret complex materials (#1.1)
- Use problem solving skills (#2.6)
- Determine the nature and extent of the information needed (#4.1)
- Use logical and mathematical reasoning within the context of various disciplines (#6.1)
- Interpret and use mathematical formulas (#6.2)
- Estimate and consider answers to mathematical problems in order to determine reasonableness (#6.5)