Python Fundamentals For Ethical Hackers

Nigeria's Premier Managed Security Service Provider

About Your Instructor

Opeyemi Atoyebi

- ❑ Red Team Lead at Esentry Systems Limited
- ❑ Developer, Researcher, Jack-of-most-trades
- ❑ CTF, Science fiction, skating.
- ❑ CRTA, CEH, ISO/IEC-27001, NSE1&2

Zidelnet

esentry

Workshop Objectives

By the end of this workshop, you will :

- Understand the use of Python and it’s strength
- Understand Python basics
- Have knowledge of Python modules and libraries
- How Python is applicable in Cybersecurity
- Be able to build Python scripts for ethical hacking assessments

Workshop Structure

Activity 1 – Python Basics

Activity 2 – Building Security tools :

- dencoder : A script that performs base64 encoding and decoding withing a specific iteration.
- pscanner : A custom port scanner that uses the python-nmap module
- inforecon: A script to perform reconnaissance on any website

Prerequisites

- Python 3
- The python module: python-nmap
- Code Editor: Visual Studio Code

Project Repository:

https://github.com/zidelnet/shehacks2021

Knowledge Check

Which of the tools below can be used for network and target scanning?

a) Dirbuster
b) Nmap
c) Burpsuite

Introduction – Python Programming Language

- Python is a widely used general – purpose, high level programming language.
- Created by Guido van Rossum in 1991 and further developed by the Python Software Foundation.
- It was designed with an emphasis on code readability and its syntax allows programmers to express their concepts in fewer lines of code

Facts About Python

- The most widely used multi-purpose, high-level programming language
- Versatile, easy to read, learn and write
- Object-orient language, portal and interactive
- For rapid development of application
- Save time and increase productivity
- Used by almost all tech-giant companies like – Google (components of Google spider & search engine), Amazon, Microsoft, Youtube, Facebook, Instagram, Dropbox, Uber etc.
- Huge collection of standard library which can be used for : Machine Learning, GUI Application, Web Frameworks like Django, Image & text processing, Web scraping, Multimedia.

Activity 1: Python Basics

Main Python Data Types

Every value in Python is called an “object”. And every object has a specific data type. The 3 most-used data types are as follows:

Integers(int): -2, -1, 0, 1, 2, 3

Floating-point numbers(float) : -1.25, -1.0, -0.5, 0.0, 1.0, 1.5

Strings: “hello”, “how are doing”

Another 3 types worth mentioning are lists, dictionaries, and tuples.

Strings

How to create a string in Python

```
 name = “Opeyemi Atoyebi”
print(name)
```

String Concatenation

```
first_name = “Funmilayo”
last_name = “Bamidele”
print(first_name + last_name)
```

String Replication

```
print(“Mary” * 5)
```

How to store strings in variables

Variables: are used to temporary store values in the computer memory

```
greetings = “Hello World”
```

Let’s break it down a bit further

greetings is the variable name

= is the assignment operator

“Hello World” is the value that stored in the variable.

String Methods

Method	Description
Count()	Returns the number of times a specified value occurs in a string
Endswith()	Returns True if the string ends with the specified value
Split()	Splits the string at the specified separator, and returns a list
Lower()	Converts a string into a lower case
Upper()	Converts a string into upper case
Title()	Converts the first character of each word to upper case
Isalnum()	Returns True if all characters in the string are alphanumeric
Isalpha()	Returns True if all characters in the string are alphabet
Isdecimal()	Returns True if all characters in the string are decimals

https://www.w3schools.com/python/python_ref_string.asp

esentry

Math Operators

For reference, here’s a list of other math operations you can apply towards numbers:

Operators	Operation	Example
**	Exponent	2 **3 = 8
%	Modulus/Remainder	22 % 8 = 6
//	Integer division	22 // 8 = 2
/	Division	22 / 8 = 2.75
*	Multiplication	3 * 3 = 9
-	Subtraction	5 – 2 = 3
+	Addition	2 + 2 = 4

esentry

Python Loops

- for loops
- while loops

For Loop

Is a handy way for iterating over a sequence such as a list, tuple, dictionary, string, etc.

```
for x in “shehacks”
     print(x)
for i in range(1,10):
     print(i)
```

Python Loops

While Loop

While loop enables you to execute a set of statements as long as the condition for them is true

```
# print as long as x is less than 8
i = 1
X = “shehacks”
While i < 8:
     print(x)
for i in range(1,10):
     print(i)
```

While Loop

While Loop

While loop enables you to execute a set of statements as long as the condition for them is true

```
# print as long as x is less than 8
i = 1
X = “shehacks”
While i < 8:
     print(x)
for i in range(1,10):
     print(i)
# break a loop with break
```

Built-in Functions in Python

input(): simple way to prompt the user for some input

```
name = input(“Hello! What’s your name?”)
print(“Nice to meet you “ + name + “!”)


age = input(“How old are you “)
print (“Good to know that you are “ + str(age) + “ years old “ +
name + “!”)
```

len(): function helps you find the length of any string, list, tuple dictionary, or another data types.

```
password = “password123”
print(“The length of the password is :”, len(password))
```

Built-in Functions in Python

filter(): used to exclude items in an iterable object (lists, tuple, dictionaries, etc.)

```
adults.py
ages = [7, 13, 16, 18, 30, 32, 28]

def mature(x):
      if x < 18:
            return False
      else:
            return True
adults = filter(mature, ages)
for i in adults:
      print(i)
```

How to Define a Function

Apart from using in-built functions, you can also define your own functions for your program.

Function: is a block of coded instructions that perform a certain action.

Functions without keyword argument

```
def greet():
      print(“Good morning ladies!”)

# Call this function
greet()
```

Function with keyword argument

```
def add_two_numbers(x,y):
      result = x + y
      print(result)

# Call this function
add_two_numbers(4,5)
```

Lists

List is used to specify an ordered sequence of elements.

```
animals = [“rat”,”bat”,”cat”]
things = [“hello”, 2.343, True, None, 42]
```

How to Add items to a List

append()

```
fruit = [“apple”,”orange”]
fruit.append(“grape”)
print(fruit)
fruit
```

insert()

```
fruit = [“banana”,”grape”]
fruit.insert(2,”apple”)
print(fruit)
```

Lists

How to Remove an Item from a List

`remove()`

```
fruit = [“apple”,”orange”,”grape]
fruit.remove(“grape”)
print(fruit)
```

`pop()`

```
fruit = [“banana”,”apple”,”grape”]
fruit.pop()
print(fruit)
```

`del()`

```
fruit  = [“grape”,”apple”,”orange”]
del fruit[1]
print(fruit)
```

Lists

Combine Two List

To mash up two lists use the + operator

```
list1 = [1,2,3]
list2 = ["a","b","c"]
combo_list = list1 + list2
print(combo_list)
```

Create a Nested List

```
nested_list = [list1,list2]
print(nested_list)
```

Sort a List

sort() function to organize all item

```
alpha = [23, 7, 1, 24, 8, 4, 6]
alpha.sort()
print(alpha)
```

Slice a List

```
alpha[0:4]                  alpha[1:4]
[23, 7, 1, 24]              [7, 1, 24]
```

Lists

Update List

```
name = [“Funmi”,”Lola”,”Tosin”,”Lucy”]
name[1] = “Adeola”
print(name)
```

Copy a List

Use the built-in copy() to replicate your data:

```
fruit = [“apple”,”orange”,”mango”]
new_fruit = fruit.copy()
print(new_fruit)
```

List Comprehensions

```
list_variable = [x for x in iterable]

list1 = [x**2 for x in range(1,10)]

list2 = [x**2 for x in range(1,10) if
x%2 ==0]
```

Tuples

- Tuples are similar to lists
- They allow you to display an ordered sequence of elements
- They are immutable – you can’t change the values stored in a tuple
- They are slightly faster than list.
- Nice way to optimize your code.

How to Create a Tuple

```
my_tuple = (1,2,3,4,5)
my_tuple[0:3]
(1,2,3)
```

Convert Tuple to List

```
x = (“pear”,”mango”,”banana”)
y = list(x)
print(y)
```

Dictionary

- A dictionary holds indexes with keys that are mapped to certain values.
- These key-value pairs offer a great way of organizing and store data
- They are mutable (you can change the stored information)
- A key value can be either a string, Boolean, or integer
- E.g. customer = {‘username’:’James Smith’, ‘online’:True, ‘fiends’:100}

How to create a Python Dictionary

```
Option 1: customer = {}
Option 2: customer = dict()
Customer = {“username”:”yemi”, “online”: True, “frends”:100}
```

How to Access a value in a Dictionary

```
dict.()  isolates keys
dict.values() isolates values
dict.items()   returns items in a list format of (key, value) tuple pairs

name = customer[“username”]
friends = customer[“friends”]
```

Loop Through the Dictionary

```
# print all key names in the dictionary
for x in new_dict:
      print(x)

# print all values in the dictionary
for x in new_dict:
      print(new_dict[x])

# loop through both keys and values
for x, y in new_dict.items():
      print(x,y)
```

Drinks break...

10:00

Start | Stop | Reset | mins: 10 | secs: 0 | type: Drinks

Breaktime for PowerPoint by Flow Simulation Ltd.

Pin controls when stopped ✓

Projects

Project repository:

https://github.com/zidelnet/SheHacks2021

dencoder.py

pscanner.py

inforecon.py

Activity 2: Building Security Tools

Project

`Project repository:`

[`https://github.com/zidelnet/SheHacks2021`](https://github.com/zidelnet/SheHacks2021)

```
dencoder.py
pscanner.py
inforecon.py
```

Additional Readings & References

https://www.w3schools.com/python/

https://inventwithpython.com

https://www.guru99.com/python-tutorials.html

https://www.thepythoncode.com/

Automate boring stuff with python

THANK YOU

esentry