Python; an Information Security Researcher's Best Friend

Daniel Weinert

Metropolitan State University of Denver

CIS-3500

Professor Linton

Assignment #6 Final

May 9th, 2022

History

Python is a general-purpose programming language that was created to increase code readability and therefore ease of use. It's creator Guido van Rossum released the first version of the language in February of 1991 as "Python 0.9.0". Van Rossum used to be the sole decision maker over the language but since 2018 the language is now governed by a council made up of five core python developers called the "Steering Council". The language had two major releases throughout the years that changed the features of the language quite a bit. The first happened in October 2000 with the release of Python 2.0, the main features included a new central hosting place for the language with version control and switching from ASCII to Unicode which allows python programmers to use 16-bit character representation compared to the old 8-bit. The most recent major release was Python 3.0 in December 2008, this version introduced a lot of fixes and easier method of achieving certain tasks. It was also the first ever version not to be backwards compatible with previous versions in order to change parts of the language that are more deeply in baked. Like for example, the modules that come with a default installation had been changed and overhauled to be more update, but that meant previous version relying on these modules wouldn't run anymore, hence the backwards compatibility thing. The current stable version of the language is Python 3.10.4. The next paragraph will go more into details on how Python uses modules.

Features

Python is known as a multi-paradigm programming language which means that it supports a lot of different styles of programming. Among them are object-oriented programming, structured programming, and aspect-oriented programming. This allows the language to be

extremely applicable in all types of situations and be suitable for most tasks. This is also due to the part that python is very module heavy and tries to not rely on core libraries too much. This means that a clean installation will only come with the most needed and basic libraries to function, but you need to install modules to expand the language and use features, you need for certain tasks. This allows a programmer to have environments specifically tailored towards their projects and programs won't get shipped with unnecessary bloat. One of the most impactful features that makes Python such a useful tool for many technologists besides programmers is the syntax. Python employs whitespaces for indentation instead of curly brackets or keywords like many other languages. This allows it to be quite readable by the human eye and makes blocks of code easier to understand. Such features are created and maintained due to the design philosophy of the language which is summarized under the "Zen of Python":

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Password Generator Program

To showcase what you can do with python we will first create a simple password generator, that will allow you to generate a random password based on a length requirement.

```
1   import secrets
2   import string
3
4   def pass_gen(pass_len):
5           chars = string.ascii_letters + string.digits
6           sec_pass = "".join(secrets.choice(chars) for i in range(pass_len))
7           return sec_pass
8
9   def main():
10          user_pass_len = int(input("Input number of digits for password: "))
11          print("Password Generated: ", pass_gen(user_pass_len))
12
13  main()
```

The code snippet above shows the code needed to make a password generator. First, we need to import modules that are included by default called secrets and strings. These include the ability to generate cryptographically safe combinations. The first function “pass_gen” is concerned with actually generating the password based on the length given from the user. The next function “user_pass_len” asks the user what length password they require and then shows the generated password to the user after it’s been created. The full script in action looks like this:

```
C:\Users\dwein\OneDrive\MSU Denver\information security>python password_gen.py
Input number of digits for password: 16
Password Generated:  MkXwYwuMr7Sga6mH
```

Apache Log Parser

Next, we will create a log parser for Apache servers. This program will allow you to parse specific information into a csv file, in this case we are extracting how many times specific hosts access our Apache server.

```
5   def reader(filename):
6       with open(filename) as f:
7           log = f.read()
8           regexp = r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}'
9           ips_list = re.findall(regexp, log)
10          return ips_list
11
12  def count(ips_list):
13      return Counter(ips_list)
14
15  def write_csv(counter):
16      with open('output.csv', 'w') as csvfile:
17          writer = csv.writer(csvfile)
18          header = ['IP', 'Frequency']
19          writer.writerow(header)
20
21          for item in counter:
22              writer.writeow( (item, counter[item]))
```

The code snippet above is the main part of the parser program. The first function “reader” allows the program to take in an Apache log file and extract the exact information needed. The line that starts with “regexp” maps out the exact location of the information from the log file. Next the “count” function will record the frequency of a host connecting to the server. Last, we have the “write_csv” function which parses the recorded information into a cvs file and ensure the formatting is proper. The csv end result will look like this:

	A	B
1	IP	Frequency
2	69.162.81.155	25
3	192.199.248.75	22
4	162.254.206.227	15
5	209.142.68.29	5
6	207.250.234.100	3
7	184.107.126.165	2
8	206.71.50.230	1
9	65.49.22.66	1
10	23.81.0.59	1
11	207.228.238.7	1

Port Scanner

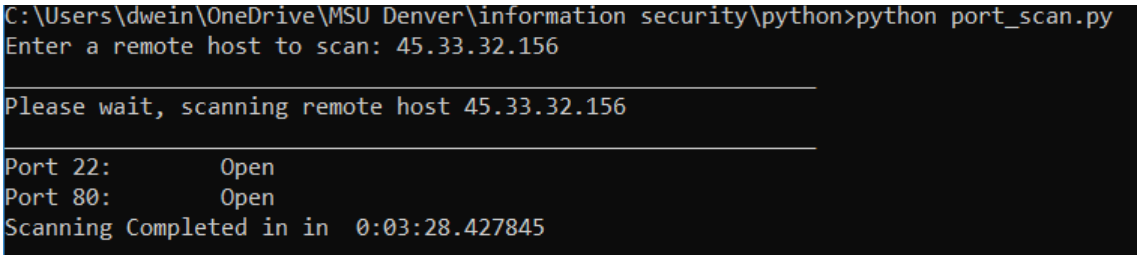
Lastly, we will be creating a simple port scanner from scratch. This will allow you to perform basic network reconnaissance without needing any extra programs or a specific operating system.

```
target = input("Enter a remote host to scan: ")
targetIP = socket.gethostbyname(target)

print("_" * 60)
print("Please wait, scanning remote host", targetIP)
print("_" * 60)

t1 = datetime.now()

try:
    for port in range (1,5000):
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        result = sock.connect_ex((targetIP, port))
        if result ==0:
            print("Port {}:        Open".format(port))
        sock.close()
```

Above is the first part of the code required for the program. The top part grabs the target host for the scan from the user. Then the print statements act as a kind of GUI to show the user that the program started scanning the targeted host. Then there is a small line to capture the current time which will be used later. Next the try statement is the actual main part of the program, here lies the sock module that actually scans the target host from port 1 to 5000 (this can be changed, depending on which ports you need scanned) and checks if they are open.

```
except KeyboardInterrupt:
    print("You pressed Ctrl+C")
    sys.exit()

except socket.gaierror:
    print("Hostname could not be resolved. Exiting")
    sys.exit()

except socket.error:
    print("Couldn't connect to server")
    sys.exit()

t2 = datetime.now()

#Calculate the difference in time to now how long the scan took
total = t2 - t1

#Printing the information on the screen
print('Scanning Completed in in ', total)
```

The second part of the program is shown above. It starts with three expect cases where the program stops the scanning, this can be due to three reasons: First because the user pressed ctrl+c, second because the hostname could not be resolved, and thirdly because no connection could be established. Then the program records the time again, in order to then calculate the total time, the scan took in the next line, which is then given to the user by the last line. The end result looks like this (Target = scanme.nmap.org (45.33.32.156) / port range = 1 to 100):



Citations:

Barot, P. (2021, June 28). How is Python Useful for Cybersecurity in 2022. *BoTree Technologies*. https://www.botreetechnologies.com/blog/python-in-cybersecurity/

Kuchling, A. M., & Zadka, M. (2022, April 26). *What's New in Python 2.0*. Python Docs. https://docs.python.org/3/whatsnew/2.0.html

Peters, T. (2004, August 22). *PEP 20 – The Zen of Python*. Python PEP. https://peps.python.org/pep-0020/

Rossum, G. V. (2009, January 20). The History of Python: A Brief Timeline of Python. *The History of Python*. https://python-history.blogspot.com/2009/01/brief-timeline-of-python.html

Singh, H. (2017, November 27). *Python for security professionals*. Infosec Resources. https://resources.infosecinstitute.com/topic/python-security-professionals-part-1/