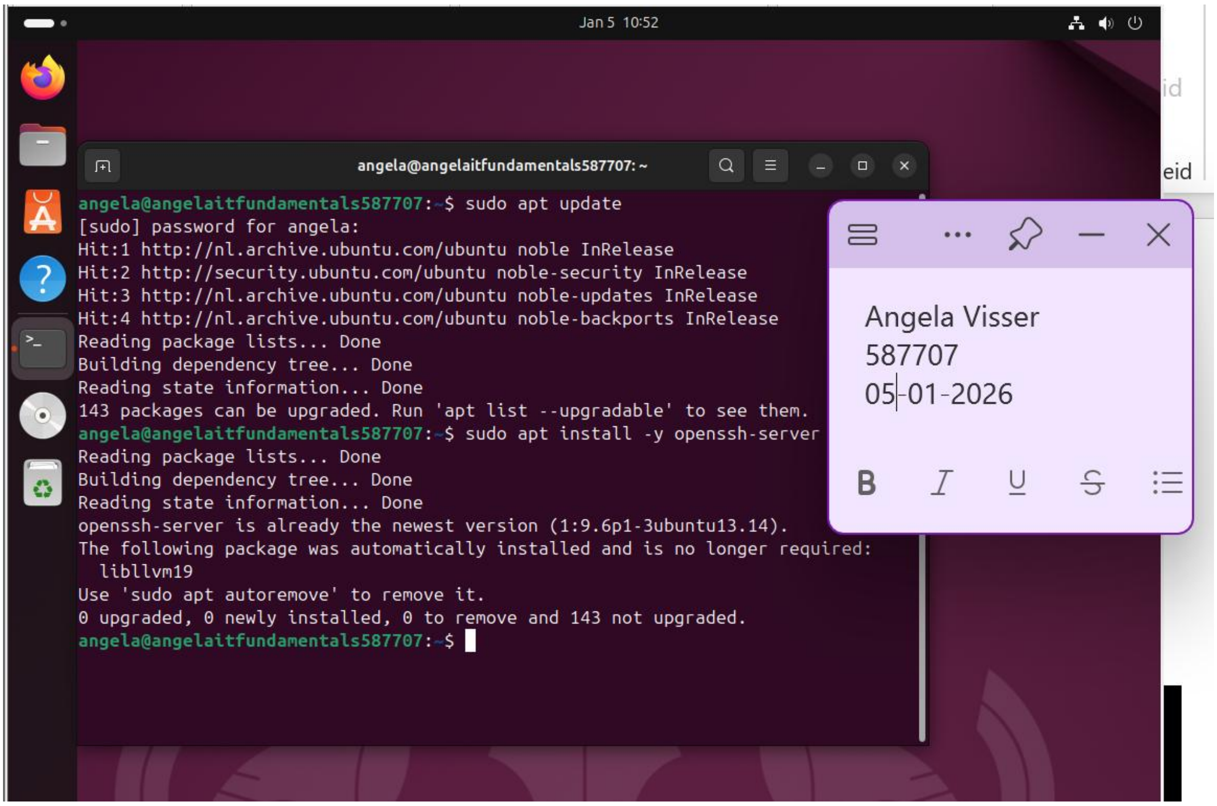
Template Week 6 – Networking

Student number:587707

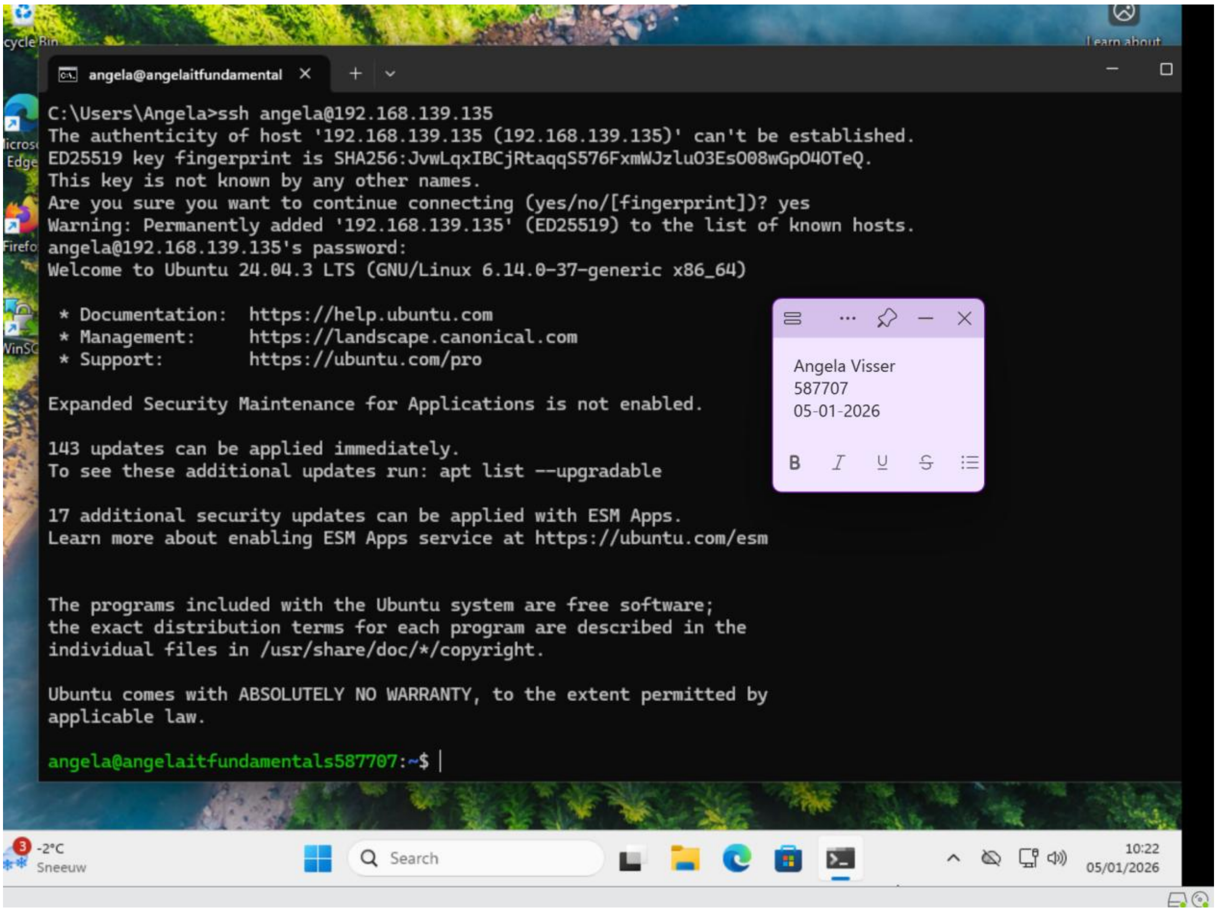
Assignment 6.1: Working from home

Screenshot installation openssh-server:



```
angela@angelaitfundamentals587707:~$ sudo apt update
[sudo] password for angela:
Hit:1 http://nl.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:3 http://nl.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:4 http://nl.archive.ubuntu.com/ubuntu noble-backports InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
143 packages can be upgraded. Run 'apt list --upgradable' to see them.
angela@angelaitfundamentals587707:~$ sudo apt install -y openssh-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openssh-server is already the newest version (1:9.6p1-3ubuntu13.14).
The following package was automatically installed and is no longer required:
  libllvm19
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 143 not upgraded.
angela@angelaitfundamentals587707:~$
```

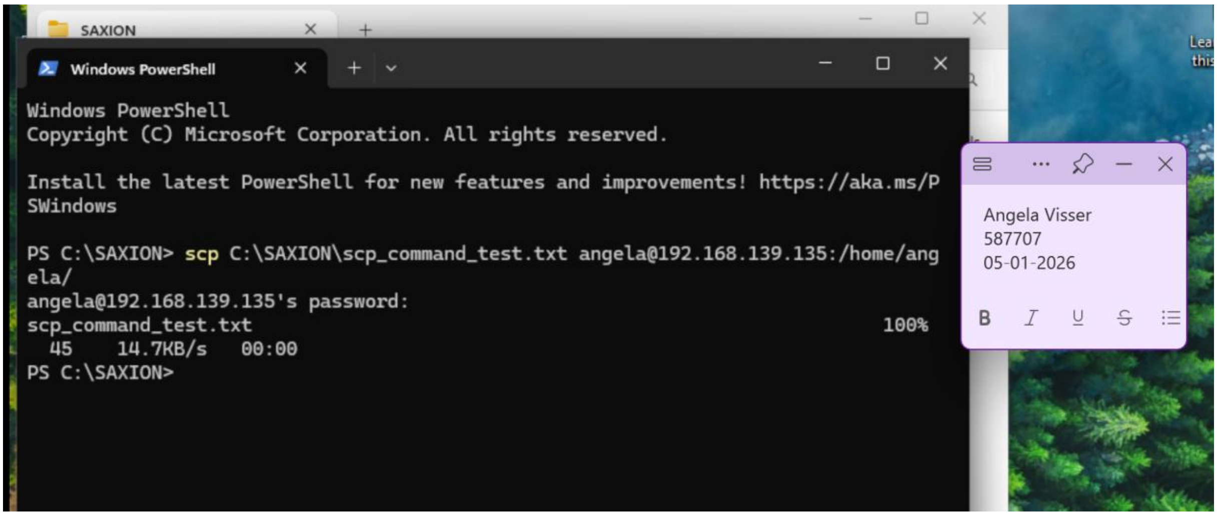
Screenshot successful SSH command execution:



```
C:\Users\Angela>ssh angela@192.168.139.135
The authenticity of host '192.168.139.135 (192.168.139.135)' can't be established.
ED25519 key fingerprint is SHA256:JvwLqxIBCjRtaqqS576FxmWJzluO3EsO08wGpO4OTeQ.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.139.135' (ED25519) to the list of known hosts.
angela@192.168.139.135's password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-37-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

143 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

17 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

angela@angelaitfundamentals587707:~$
```

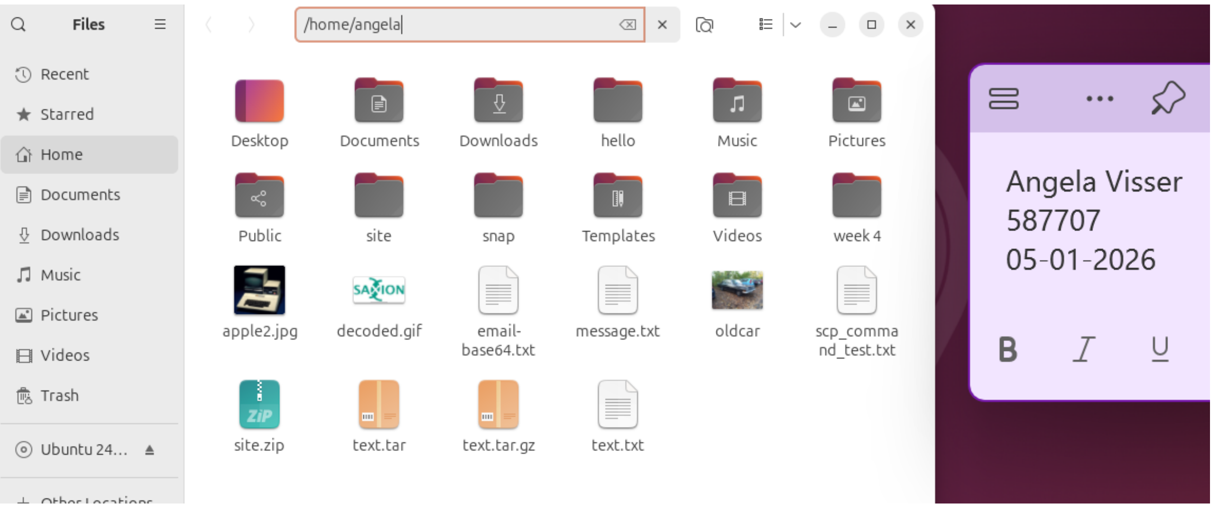
Angela Visser
587707
05-01-2026

Screenshot successful execution SCP command:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/P
SWindows

PS C:\SAXION> scp C:\SAXION\scp_command_test.txt angela@192.168.139.135:/home/ang
ela/
angela@192.168.139.135's password:
scp_command_test.txt                                                        100%
  45    14.7KB/s   00:00
PS C:\SAXION>
```

Angela Visser
587707
05-01-2026



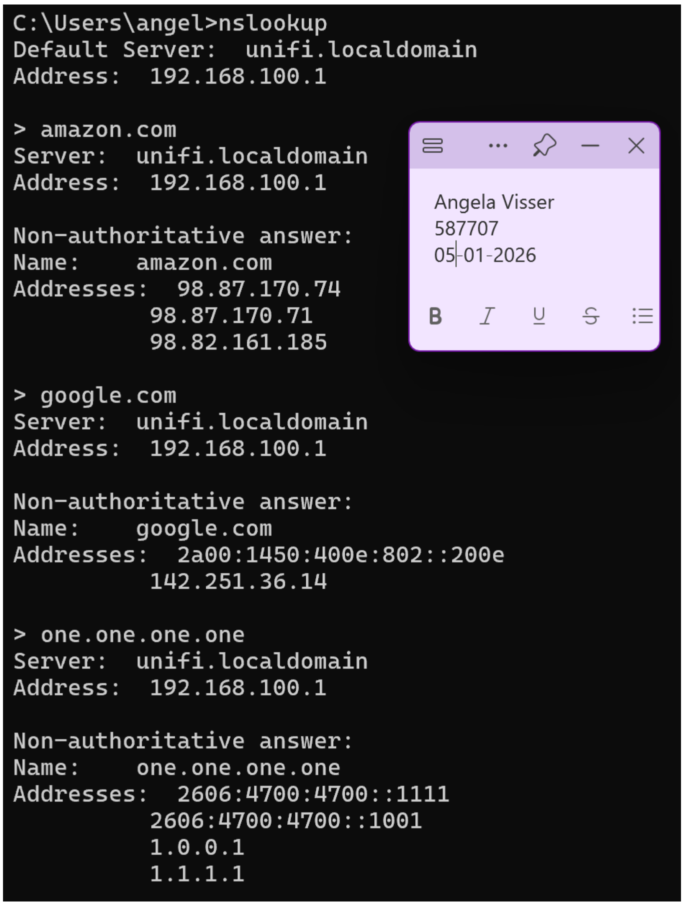
Angela Visser
587707
05-01-2026

Screenshot remmina:

REMMINA WIL NIET VERBINDEN MET WINDOWS.
This was discussed with the teacher.

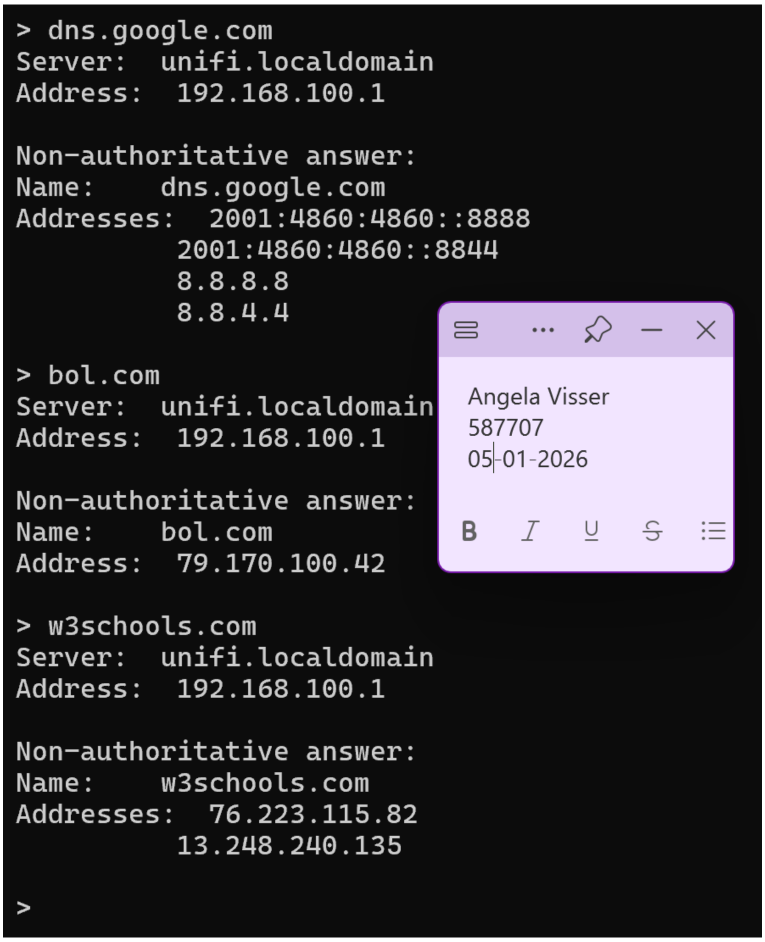
Assignment 6.2: IP addresses websites

Relevant screenshots nslookup command:



```
C:\Users\angel>nslookup
Default Server:  unifi.localdomain
Address:  192.168.100.1

> amazon.com
Server:  unifi.localdomain
Address:  192.168.100.1

Non-authoritative answer:
Name:    amazon.com
Addresses:  98.87.170.74
          98.87.170.71
          98.82.161.185

> google.com
Server:  unifi.localdomain
Address:  192.168.100.1

Non-authoritative answer:
Name:    google.com
Addresses:  2a00:1450:400e:802::200e
          142.251.36.14

> one.one.one.one
Server:  unifi.localdomain
Address:  192.168.100.1

Non-authoritative answer:
Name:    one.one.one.one
Addresses:  2606:4700:4700::1111
          2606:4700:4700::1001
          1.0.0.1
          1.1.1.1
```

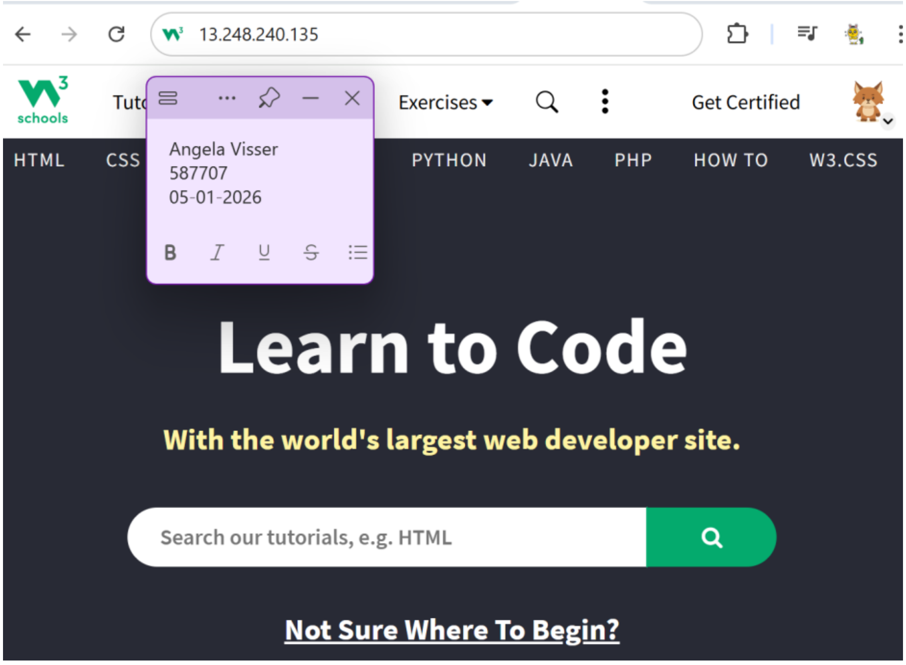
Angela Visser
587707
05-01-2026



```
> dns.google.com
Server:  unifi.localdomain
Address:  192.168.100.1

Non-authoritative answer:
Name:    dns.google.com
Addresses:  2001:4860:4860::8888
          2001:4860:4860::8844
          8.8.8.8
          8.8.4.4

> bol.com
Server:  unifi.localdomain
Address:  192.168.100.1

Non-authoritative answer:
Name:    bol.com
Address:  79.170.100.42

> w3schools.com
Server:  unifi.localdomain
Address:  192.168.100.1

Non-authoritative answer:
Name:    w3schools.com
Addresses:  76.223.115.82
          13.248.240.135

>
```

Angela Visser
587707
05-01-2026

Screenshot website visit via IP address:



13.248.240.135

Angela Visser
587707
05-01-2026

Assignment 6.3: subnetting

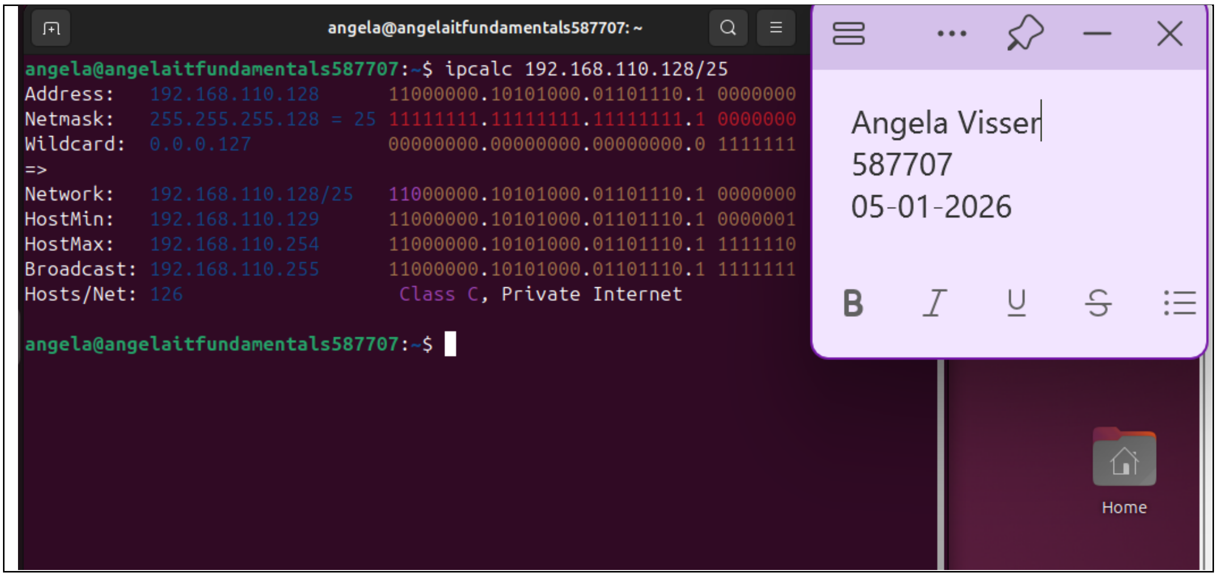
How many IP addresses are in this network configuration 192.168.110.128/25?

The 25 at the end mean that 25 bits are taken for the network, which leaves us (32-25=) 7 bits for hosts. The formula to calculate the total available IP addresses is 2^(host bits).
2^(7)= 128 IP addresses.

What is the usable IP range to hand out to the connected computers?

The first IP address is the network ID and the last is used for broadcasts. This gives us 2 IP adresses less to use. 128 – 2 = 126 IP addresses are usable.

Check your two previous answers with this Linux command: `ipcalc 192.168.110.128/25`



angela@angelaitfundamentals587707: ~

```
angela@angelaitfundamentals587707:~$ ipcalc 192.168.110.128/25
Address:   192.168.110.128        11000000.10101000.01101110.1 0000000
Netmask:   255.255.255.128 = 25   11111111.11111111.11111111.1 0000000
Wildcard:  0.0.0.127              00000000.00000000.00000000.0 1111111
=>
Network:   192.168.110.128/25     11000000.10101000.01101110.1 0000000
HostMin:   192.168.110.129        11000000.10101000.01101110.1 0000001
HostMax:   192.168.110.254        11000000.10101000.01101110.1 1111110
Broadcast: 192.168.110.255        11000000.10101000.01101110.1 1111111
Hosts/Net: 126                     Class C, Private Internet

angela@angelaitfundamentals587707:~$
```

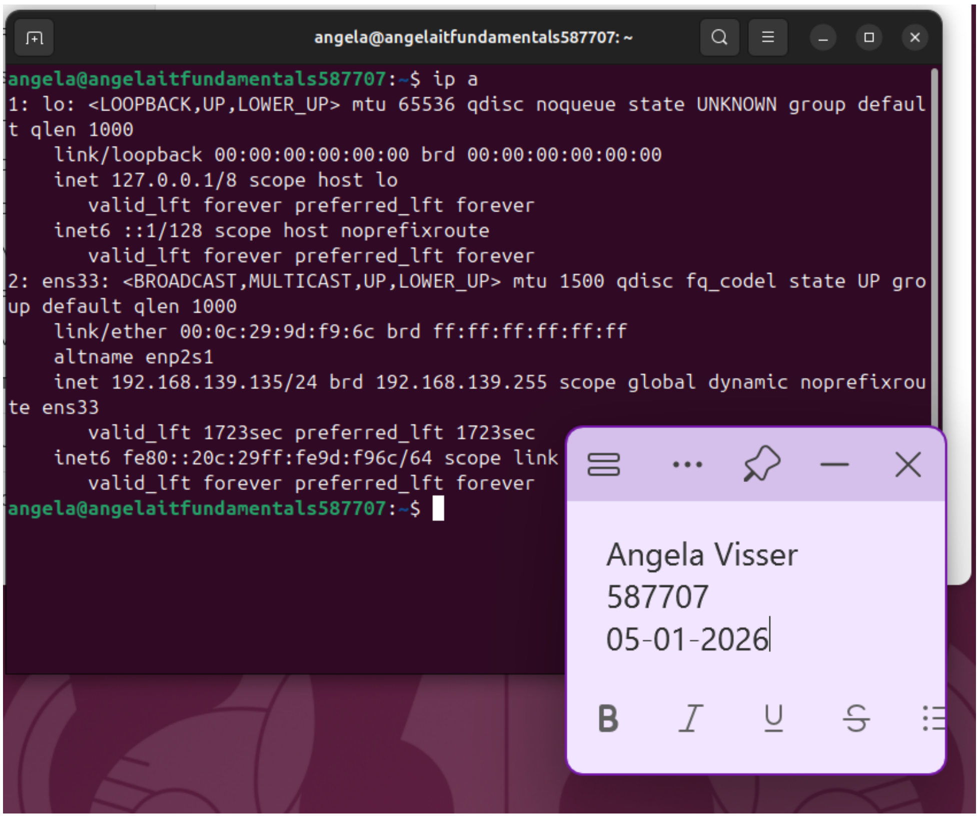
Angela Visser
587707
05-01-2026

Explain the above calculation in your own words.

The last octet of the subnet is 128(10000000). To determine the block size we can subtract the last octet number from 256 (11111111). 256-128= 128.
The first address is the network ID, this makes address 129 the first address to be used by a host (HostMin). The last possible host address is 254 because 255 is the broadcast address. 128 – 2 gives us 126 usable addresses (Hosts/Net).

Assignment 6.4: HTML

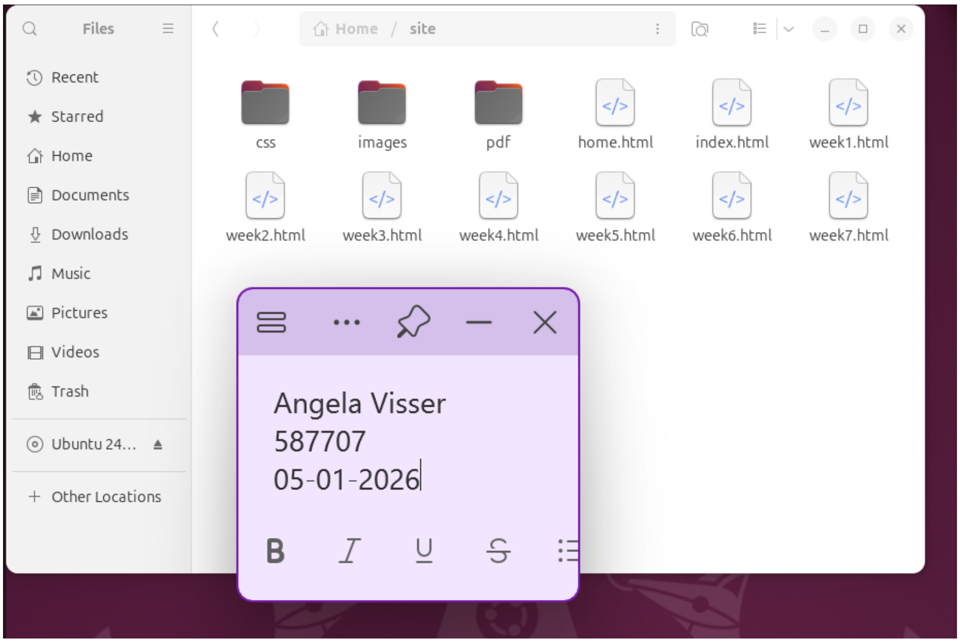
Screenshot IP address Ubuntu VM:



```
angela@angelaitfundamentals587707:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gro
up default qlen 1000
    link/ether 00:0c:29:9d:f9:6c brd ff:ff:ff:ff:ff:ff
    altname enp2s1
    inet 192.168.139.135/24 brd 192.168.139.255 scope global dynamic noprefixrou
te ens33
       valid_lft 1723sec preferred_lft 1723sec
    inet6 fe80::20c:29ff:fe9d:f96c/64 scope link
       valid_lft forever preferred_lft forever
angela@angelaitfundamentals587707:~$
```

Angela Visser
587707
05-01-2026

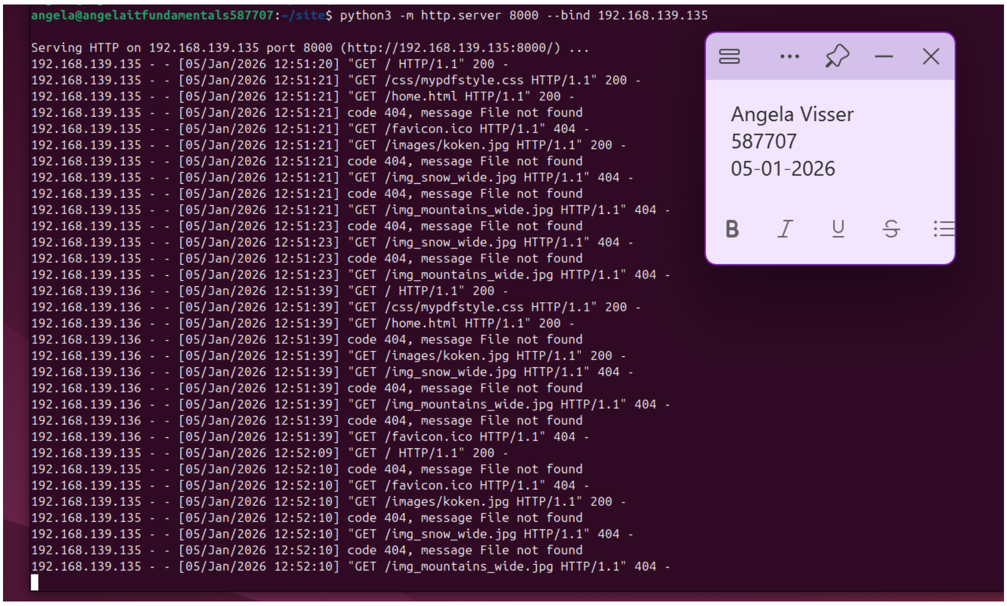
Screenshot of Site directory contents:



css
images
pdf
home.html
index.html
week1.html
week2.html
week3.html
week4.html
week5.html
week6.html
week7.html

Angela Visser
587707
05-01-2026

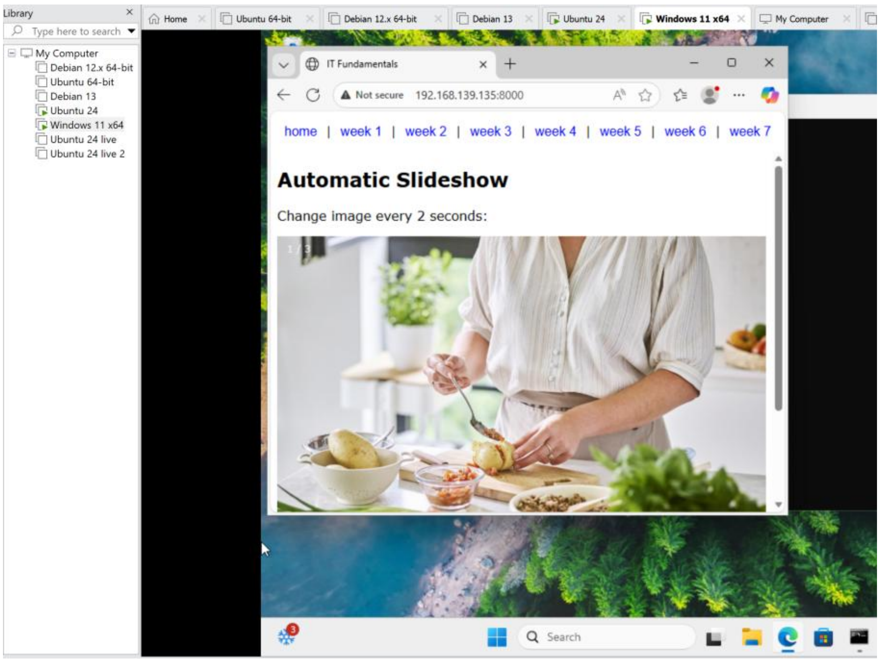
Screenshot python3 webserver command:



```
angela@angelaitfundamentals587707:~/site$ python3 -m http.server 8000 --bind 192.168.139.135

Serving HTTP on 192.168.139.135 port 8000 (http://192.168.139.135:8000/) ...
192.168.139.135 - - [05/Jan/2026 12:51:20] "GET / HTTP/1.1" 200 -
192.168.139.135 - - [05/Jan/2026 12:51:21] "GET /css/mypdfstyle.css HTTP/1.1" 200 -
192.168.139.135 - - [05/Jan/2026 12:51:21] "GET /home.html HTTP/1.1" 200 -
192.168.139.135 - - [05/Jan/2026 12:51:21] code 404, message File not found
192.168.139.135 - - [05/Jan/2026 12:51:21] "GET /favicon.ico HTTP/1.1" 404 -
192.168.139.135 - - [05/Jan/2026 12:51:21] "GET /images/koken.jpg HTTP/1.1" 200 -
192.168.139.135 - - [05/Jan/2026 12:51:21] code 404, message File not found
192.168.139.135 - - [05/Jan/2026 12:51:21] "GET /img_snow_wide.jpg HTTP/1.1" 404 -
192.168.139.135 - - [05/Jan/2026 12:51:21] code 404, message File not found
192.168.139.135 - - [05/Jan/2026 12:51:21] "GET /img_mountains_wide.jpg HTTP/1.1" 404 -
192.168.139.135 - - [05/Jan/2026 12:51:23] code 404, message File not found
192.168.139.135 - - [05/Jan/2026 12:51:23] "GET /img_snow_wide.jpg HTTP/1.1" 404 -
192.168.139.135 - - [05/Jan/2026 12:51:23] code 404, message File not found
192.168.139.135 - - [05/Jan/2026 12:51:23] "GET /img_mountains_wide.jpg HTTP/1.1" 404 -
192.168.139.136 - - [05/Jan/2026 12:51:39] "GET / HTTP/1.1" 200 -
192.168.139.136 - - [05/Jan/2026 12:51:39] "GET /css/mypdfstyle.css HTTP/1.1" 200 -
192.168.139.136 - - [05/Jan/2026 12:51:39] "GET /home.html HTTP/1.1" 200 -
192.168.139.136 - - [05/Jan/2026 12:51:39] code 404, message File not found
192.168.139.136 - - [05/Jan/2026 12:51:39] "GET /images/koken.jpg HTTP/1.1" 200 -
192.168.139.136 - - [05/Jan/2026 12:51:39] "GET /img_snow_wide.jpg HTTP/1.1" 404 -
192.168.139.136 - - [05/Jan/2026 12:51:39] code 404, message File not found
192.168.139.136 - - [05/Jan/2026 12:51:39] "GET /img_mountains_wide.jpg HTTP/1.1" 404 -
192.168.139.136 - - [05/Jan/2026 12:51:39] code 404, message File not found
192.168.139.136 - - [05/Jan/2026 12:51:39] "GET /favicon.ico HTTP/1.1" 404 -
192.168.139.135 - - [05/Jan/2026 12:52:09] "GET / HTTP/1.1" 200 -
192.168.139.135 - - [05/Jan/2026 12:52:10] code 404, message File not found
192.168.139.135 - - [05/Jan/2026 12:52:10] "GET /favicon.ico HTTP/1.1" 404 -
192.168.139.135 - - [05/Jan/2026 12:52:10] "GET /images/koken.jpg HTTP/1.1" 200 -
192.168.139.135 - - [05/Jan/2026 12:52:10] code 404, message File not found
192.168.139.135 - - [05/Jan/2026 12:52:10] "GET /img_snow_wide.jpg HTTP/1.1" 404 -
192.168.139.135 - - [05/Jan/2026 12:52:10] code 404, message File not found
192.168.139.135 - - [05/Jan/2026 12:52:10] "GET /img_mountains_wide.jpg HTTP/1.1" 404 -
```

Angela Visser
587707
05-01-2026

Screenshot web browser visits your site



Automatic Slideshow

Change image every 2 seconds:

Assignment 6.5: Network segment

Remember that bitwise java application you've made in week 2? Expand that application so that you can also calculate a network segment as explained in the PowerPoint slides of week 6. Use the bitwise & AND operator. You need to be able to input two Strings. An IP address and a subnet.

IP: 192.168.1.100 and subnet: 255.255.255.224 for /27

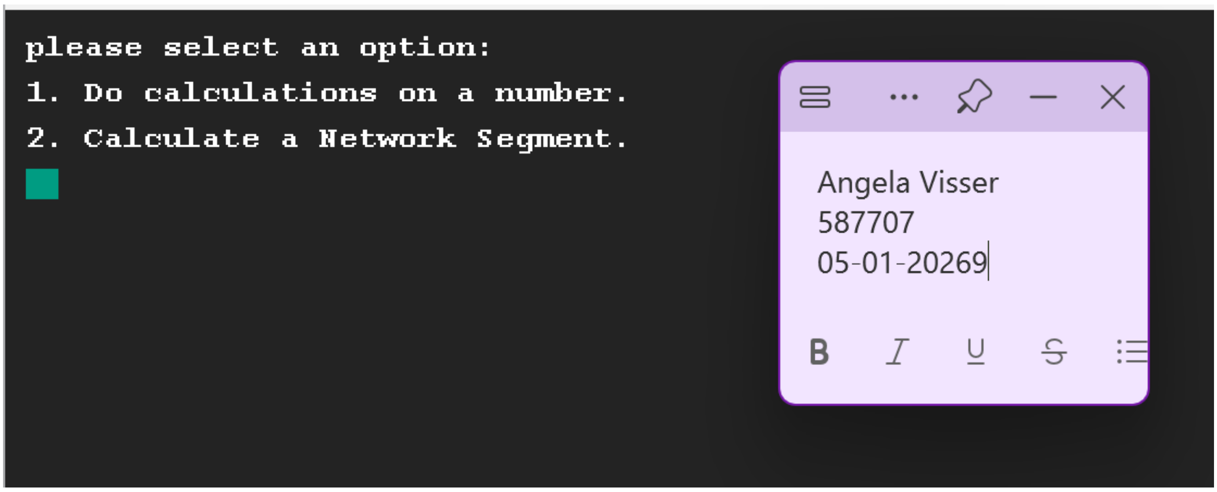
```
Example: 192.168.1.100/27
Calculate the network segment
IP Address:   11000000.10101000.00000001.01100100
Subnet Mask:  11111111.11111111.11111111.11100000
---------------------------------------------
Network Addr: 11000000.10101000.00000001.01100000
```

This gives 192.168.1.96 in decimal as the network address.
For a /27 subnet, each segment (or subnet) has 32 IP addresses (2^5).
The range of this network segment is from 192.168.1.96 to 192.168.1.127.

Paste source code here, with a screenshot of a working application.

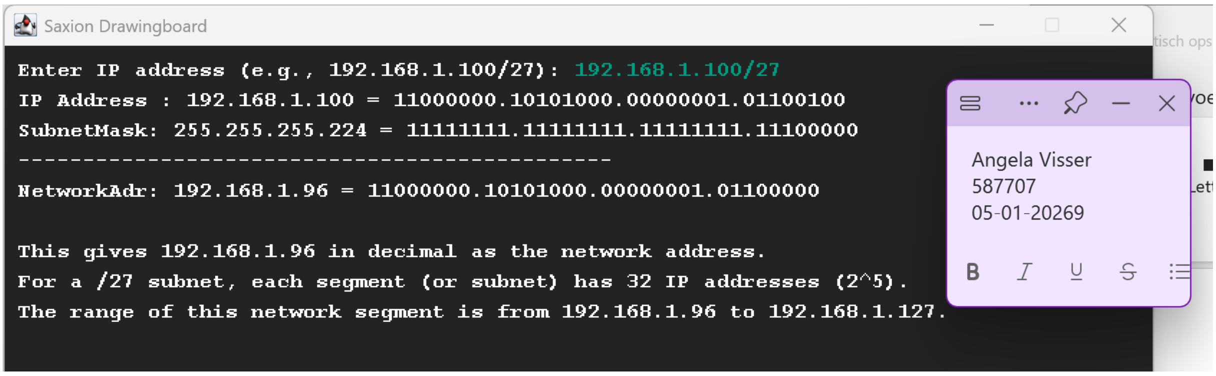
~~~
```import nl.saxion.app.SaxionApp;

import java.awt.*;

public class Application implements Runnable {

    public static void main(String[] args) {
        SaxionApp.start(new Application(), 800, 800);
    }

    public void run() {
        int input1=-1;
        SaxionApp.printLine("please select an option:");
        SaxionApp.printLine("1. Do calculations on a number.");
        SaxionApp.printLine("2. Calculate a Network Segment.");
        input1 = getUserInput(1,2, "Menu option does not exist");

        if (input1 == 1) {
            int input = -1;
            SaxionApp.print("Please enter a number: ");
            int number = SaxionApp.readInt();
            ShowMenu();
            while (input != 0) {

                SaxionApp.print("Choose option: ");
                input = getUserInput(0, 4, "Menu option does not exist");
~~~

```
        if (input == 1) {
            if ((number & 1) == 1) {
                SaxionApp.printLine(number + " is odd.");
            } else {
                SaxionApp.printLine(number + " is even.");
            }
            SaxionApp.pause();
        } else if (input == 2) {
            if ((number & (number - 1)) == 0) {
                SaxionApp.printLine(number + " is a power of 2");
            } else {
                SaxionApp.printLine(number + " is not a power of 2");
            }
            SaxionApp.pause();
        } else if (input == 3) {
            int numberInverted = (~number) + 1;
            SaxionApp.printLine(numberInverted + " is " + number + " as two's complement");
            SaxionApp.pause();

        } else if (input == 4) {
            SaxionApp.clear();
            SaxionApp.print("Please enter a number: ");
            number = SaxionApp.readInt();
            ShowMenu();
        }
    }
} else if (input1 == 2) {

    SaxionApp.clear();
    SaxionApp.print("Enter IP address (e.g., 192.168.1.100/27): ");
    String cidr = SaxionApp.readString();
    String[] parts = cidr.split("/");
    String ipStr = parts[0];
    int prefix = Integer.parseInt(parts[1]);

    int ip = ipToInt(ipStr);
    int mask = prefixToMask(prefix);
    int network = ip & mask;

    SaxionApp.printLine("IP Address : " + ipStr + " = " + toBinary(ip));
    SaxionApp.printLine("SubnetMask: " + intToIp(mask) + " = " + toBinary(mask));
    SaxionApp.printLine("--------------------------------------------");
    SaxionApp.printLine("NetworkAdr: " + intToIp(network) + " = " + toBinary(network));

    SaxionApp.printLine();
    SaxionApp.printLine("This gives " + intToIp(network)
            + " in decimal as the network address.");
        // total addresses in this subnet:
        int totalAddresses = 1 << (32 - prefix);
        SaxionApp.printLine("For a /" + prefix
                + " subnet, each segment (or subnet) has "
                + totalAddresses + " IP addresses (2^" + (32 - prefix) + ")."); //Host bits

        // first address = network, last (broadcast)= network + totalAddresses - 1
        int firstInRange = network;
        int lastInRange = network + totalAddresses - 1;

        SaxionApp.printLine("The range of this network segment is from "
                + intToIp(firstInRange) + " to " + intToIp(lastInRange) + ".");

    }

}
public void ShowMenu(){
    SaxionApp.printLine("Menu");
    SaxionApp.printLine("----------------------");
    SaxionApp.printLine("1. Is number odd?");
    SaxionApp.printLine("2. Is number a power of 2?");
    SaxionApp.printLine("3. Two's complement of number?");
    SaxionApp.printLine("4. Choose a new number.");
    SaxionApp.printLine("0. Exit");
}
public int getUserInput(int min, int max, String errorMessage){
    int input = SaxionApp.readInt();
    while (input <min || input > max){
        SaxionApp.printLine(errorMessage,Color.red);
        input = SaxionApp.readInt();
    }
    return input;
}

// convert dotted IP to 32-bit int
private static int ipToInt(String ip) {
    String[] octets = ip.split("\\.");
    int result = 0;
    for (int i = 0; i < 4; i++) {
        int octet = Integer.parseInt(octets[i]);
        result = (result << 8) | (octet & 0xFF);
    }
    return result;
}

// convert prefix length (e.g. 27) to mask as 32-bit int
private static int prefixToMask(int prefix) {
    return prefix == 0 ? 0 : (int) (0xFFFFFFFFL << (32 - prefix));
    }

    // convert 32-bit int back to dotted IP
    private static String intToIp(int value) {
        return ((value >>> 24) & 0xFF) + "." +
               ((value >>> 16) & 0xFF) + "." +
               ((value >>> 8)  & 0xFF) + "." +
               (value & 0xFF);
    }

    // format 32-bit int as 4 groups of 8-bit binary
    private static String toBinary(int value) {
        StringBuilder sb = new StringBuilder(35);
        for (int i = 31; i >= 0; i--) {
            sb.append(((value >>> i) & 1) == 1 ? '1' : '0');
            if (i % 8 == 0 && i != 0) {
                sb.append('.');
            }
        }
        return sb.toString();
    }
}
```



```
please select an option:
1. Do calculations on a number.
2. Calculate a Network Segment.
```

Angela Visser
587707
05-01-20269



```
Enter IP address (e.g., 192.168.1.100/27): 192.168.1.100/27
IP Address : 192.168.1.100 = 11000000.10101000.00000001.01100100
SubnetMask: 255.255.255.224 = 11111111.11111111.11111111.11100000
---------------------------------------------
NetworkAdr: 192.168.1.96 = 11000000.10101000.00000001.01100000

This gives 192.168.1.96 in decimal as the network address.
For a /27 subnet, each segment (or subnet) has 32 IP addresses (2^5).
The range of this network segment is from 192.168.1.96 to 192.168.1.127.
```

Angela Visser
587707
05-01-20269

Ready? Save this file and export it as a pdf file with the name: **week6.pdf**