Project #1 – IPv4 Break Down

Objectives

- Code a script / program that produces the required output.

Your mission...

You goal is to create a Python script (Program) that will ask the user for **an IPv4 address with the /prefix mask**. (i.e. `192.168.2.155/25`) From this information, you can determine the following and output this to the user.

- The **IP Address** (The user entered)
- The **Subnet Mask** (Converted from ***/prefix*** to dotted decimal format – i.e. `/25` → `255.255.255.128`)
- The **Wildcard Mask** (The Inverse of the Subnet Mask – i.e. `/25` → `0.0.0.127`)
- The **Network Address** that the host belongs too. (Bitwise **AND** the IP Address and Subnet Mask)
- The **Minimum Host IP** Address for this network. (Network Address + 1)
- The **Maximum Host IP** Address for this network. (Broadcast Address - 1)
- The **Broadcast Address** for this network. (Bitwise **OR** the IP Address and Wildcard Mask)
- The **Number of Hosts per Network** allowed on this network.

Requirements

Create a source code / script file named ***Your_Initials*_IPv4BreakDown.py**.
For example, if your name is Harley Quinn, you would name the script **HQ_ IPv4BreakDown.py**.

A **documentation header and Comments** are required though out your code.)

Write a short program that does all of the following:

1. **Asks the user** for an IP Address with the subnet prefix. (i.e. `192.168.2.155/25`)
2. Validate and convert the **user input string**.
    - Check for that there are **no spaces** " " in the input string.
    - **Break the input string** into **IP** and **Prefix** (using "/" to split)
    - **Break the IP** (from above) into a list of octets (using "." to split)
    - Validate that the prefix must be between (**8-31**) and each Octet (element in the IP list) must be between (**0-255**)
    - If any of the information is incorrect loop back and ask for the input again. (Provide output of the correct format)
3. You must use **Separate Lists** to store all the Addresses. (~7 List Variables for IP Address, Network Address, Subnet Mask, Wildcard Mask, Network Broadcast Address, Minimum Host IP, and Maximum Host IP)

    i.e. The IP Address should be stored in a 4 element list [205, 210, 49, 10]
    The Subnet Mask should be stored in another 4 element list [255, 255, 255, 0] etc…
4. Using the IP List and Prefix variable to **calculate the required** answers (to populate the other lists variables).
    - To calculate the subnet mask convert the ***/prefix*** to a string with so many 1s and the rest 0s. Then break this down into groups of 8 bits for each octet.
      (i.e. **/25** → 11111111111111111111111110000000 = 25 x 1s + (32-25) 7 x 0s.
5. Output should provide all answers in **dotted decimal and binary**. (Must match the provide output sample).

NOTE: This is **NOT** a group assignment and must be completed individually.

Console Output Sample

```
>>>
= RESTART: C:/CH_IPv4BreakDown.py
IPv4 Break Down Program

Please enter an IPv4 Address and prefix (#.#.#.#/Prefix): 300.168.2.1/24
- The correct format is [0-255].[0-255].[0-255].[0-255]/[8-31] Prefix
- Example: 192.168.2.1/24 (no spaces)

Please enter an IPv4 Address and prefix (#.#.#.#/MM): 192.168.2.142/20

Address:      192.168.2.142      11000000.10101000.00000010.10001110
Netmask:      255.255.240.0      11111111.11111111.11110000.00000000
Wildcard:     0.0.15.255         00000000.00000000.00001111.11111111
-----------------------------------------------------------------------
Network:      192.168.0.0        11000000.10101000.00000000.00000000
Broadcast:    192.168.15.255     11000000.10101000.00001111.11111111
HostMin:      192.168.0.1        11000000.10101000.00000000.00000001
HostMax:      192.168.15.254     11000000.10101000.00001111.11111110
Host/Net:     4094
>>>
```

Marking Scheme

Marks Available	*Description*
4	Code runs correctly (Meets the requirements) without crashing (All or nothing)
2	Has a comment header block listing program name, purpose, coder, and date. Comments are used throughout to explain what is happening. (All or nothing)
2	Creating / Using needed variables and lists correctly.
2	User is prompted and inputs from the keyboard (use appropriate data types)
2	Divide the input string using the *string*.**split()** method, into a variable for *prefix* and a 4-element list containing the *IP Address* (4 Octets).
2	If there is a problem with the above splitting, prompt the user (Noting the correct input format) and ask again for the IP Address.
2 (B)	**BONUS**: Validate user input is in correct ranges, if not asked to try again. (Valid for Octets = 0 to 255, Prefix = 8 to 31) NOTE: Yes, I know Octet 1 should not allowed zero and the Last Octet cannot be 255 but just keeping it simple.
2	Calculate the Subnet Mask and Inverse Wildcard Mask.
2	Calculate the Network Address. (4 Octets)
2	Calculate the Number of Hosts allowed on the network from number of Hosts bits.
2	Calculate the First IP Address in range.
2	Calculate the Last IP Address in range.
2	Calculate the Broadcast Address for the range.
2	Output shows the correct answers. (You can verify here)
2	Output is formatted correctly
30	TOTAL (Max grade is 30 / 30 = 100% with Bonus)

NOTE: This is **NOT** a group assignment and must be completed individually.

Project #1 – IPv4 Break Down – Appendix (Logic / Math Aids)

The follow appendix has pointers to help with the tools, logic and math needed.

1) Example of the List / IP logic and math:

Example User Input: "172.35.175.23/20" (String)					
IP: "172.35.175.23"				Prefix:"20"	Split into a List by "/"
"172"	"35"	"175"	"23"	"20"	Split IP into another List by "."
172	35	175	23	20	Convert List to Integers.
172	35	175	23	IP Address	**IP Address** (List of 4 Octets)
255	255	240	0	Subnet Mask (/20)	**Subnet Mask** (Dotted Decimal).
172	35	175	26	IP AND Subnet Mask	**Network Address** (Bitwise AND between IP and Subnet Mask).
255	255	240	0		
172	35	160	0	Network IP	
255	255	240	0	Subnet Mask XOR 1s	**Wildcard Mask**. (Bitwise XOR the Subnet Mask with all 1s)
255	255	255	255		
0	0	15	255	Wildcard Mask	
172	35	160	0	Network IP OR Wildcard Mask	**Network Broadcast Address** (Bitwise OR the Network IP with the Wildcard Mask.)
0	0	15	255		
172	35	175	255	Net Broadcast	
172	35	160	1	Network IP + 1	**Minimum HOST IP**
172	35	175	254	Net Broadcast - 1	**Maximum HOST IP**

2) Example of the Number of Hosts per Network math: HOSTs / Network

$2^{(32-prefix)} = 2^{(32-20=\mathbf{12})} = 4096 - 2$ (For Network and Broadcast) = 4094 Hosts

NOTE: This is **NOT** a group assignment and must be completed individually.

3) Example of formatting and conversion to binary in Python.

This is a bit complicated but a great way to format output.

First you have a LIST with the IP Address (one of many lists in the project).

If your said `print( list_variable )` you would see [172, 35, 175, 23]
However, you want dotted decimal format. 172.35.175.23
This can be achieved with the `"separator".join( list_variable )` method.

i.e. `".".join( IPAddress )` will return a string `"172.32.175.23"`

WARNING: `.join()` only works with strings and your list at this point will contain integers.

Solution: Use a `for` loop convert each list element to a string.

i.e. `[ str(x) for x in IPAddress ]` will return a copy of the IPAddress list with each element converted to a string. (Explanation: You are saying, for each octet (x) in the *IPAddress* list convert that element to a str() and store in a new temp list. You can store this newlist in another variable or use it directly in the .join() method above.

Now for the Binary Conversion:

From our labs, you know how to convert a Decimal value and print its binary equivalent.

i.e. 18 = 0b10010 in binary. (Notice that the leading zeros are not there and the ugly '0b')

Solution: We know that strings can be used like a list and we can use slices.

i.e. `str = "CHAD"`, so `str[2:]` says element 2 to the end of the string = `"AD"`

So how do we get the leading zeros and remove the "0b"?

If we add **256** to the value we would be turn on the 9th bit in binary.

i.e. 18 = 0b10010 in binary. So, 18 + 256 = 274 in binary that is "0b100010010" = *string*

If you then take a slice of that *string* we could get "00010010" and we have now are leading zeros and cut off the "0b1".

Again you can do this with a `for` loop for all elements in a list.

LAST NOTE: To line up the output in columns, remember the escape character `"\t"`

GOOD LUCK

Take your time, focus on the Math and Lists first
and then worry about the output and format.

NOTE: This is **NOT** a group assignment and must be completed individually.