# Documentation

- Def ipcalc:

-Uses a single IP address.
-Runs all the IP Calculator tests from part 2.1.
-Creates a dictionary called output at the start, for any tests that were run on the IP address the results were then added into the dictionary and returned to the server.
-Calls on the convert_to_bin() method:
    -Converts an ip address in decimal dot notation represented as a string into a list of four binary strings each representing one byte of the address
    **-Parameter:** The ip address as a string in decimal dot notation. e.g. "132.206.19.7"
    **-Returns:** An array of four binary strings each representing one byte of ip e.g. ['10000100', '11001110', '00010011', '00000111']

-Code walkthrough:
Split IP into an array
For each number in it convert to an integer
Format the integer as binary
Return an array

- Def subnet:

-Uses a single IP address and a subnet mask
-Runs all the Subnet Calculator tests from part 2.2

-Calls on cidr_not method:
    - Take in an array of four strings representing the bytes of a subnet mask
    -Calculates the CIDR number

**Parameter**: An array containing the subnet mask of an IP address in binary

**Returns**: The CIDR number for the IP address in the subnet endpoint

-Code walkthrough:
Iterate through each of the 4 strings representing the bytes of subnet mask
If the bit is a 1, increment the CIDR number
The look will check every bit in the subnet mask and then return the CIDR number
Used for the subnet endpoint

-Calls on get_broadcasts method:
    -Calculates a list of broadcast addresses for each subnet

**-Parameter**: An array containing 4 ip addresses each represented as a string. The ip addresses are the valid subnets eg.
["192.168.10.0","192.168.10.64","192.168.10.128","192.168.10.192"]

**-Return**: An array containing 4 ip addresses each represented as a string. The ip addresses are the broadcast addresses eg.
["192.168.10.63","192.168.10.127","192.168.10.191","192.168.10.255"]

-Code walkthrough:
For each ip string in the list, split each byte of the ip into a list
Add this list of bytes to a new list 'tmp'
Pop off the last element of the list of bytes and call it last
Add 63 to the int version of the string byte and call it last_n
Append the string version of last_n to the list of bytes
Create new list called broadcasts and append each list of bytes to it
Call '.'.join() to make the list of bytes a string again

-Calls on get_firstAddress method:
-Calculates a list of ip addresses as strings that represent the network addresses of a subnet

-**Parameter**: An array containing 4 ip addresses each represented as a string. The ip addresses are the valid subnets eg.
["192.168.10.0","192.168.10.64","192.168.10.128","192.168.10.192"]

**-Returns**: An array containing 4 ip addresses each represented as a string. The ip addresses are the network addresses (first addresses of the subnet)

-Code walkthrough:
For each ip string in the list, split each byte of the ip into a list
Add this list of bytes to a new list 'tmp'
Pop off the last element of the list of bytes and call it last
Add 1 to the int version of the string byte and call it last_n
Append the string version of last_n to the list of bytes
Create new list called first_addr and append each list of bytes to it
Call '.'.join() to make the list of bytes a string again

-Calls on get_lastAddress method:
- Calculates a list of ip addresses as strings that represent the last addresses of a subnet

**-Parameter**: An array containing 4 ip addresses each represented as a string. The ip addresses are the valid subnets eg.
["192.168.10.0","192.168.10.64","192.168.10.128","192.168.10.192"]

**-Return**: An array containing 4 ip addresses each represented as a string. The ip addresses are the last addresses of the subnet

-Code walkthrough:
For each ip string in the list, split each byte of the ip into a list
Add this list of bytes to a new list 'tmp'
Pop off the last element of the list of bytes and call it last
Add 62 to the integer version of the string byte and call it last_n
Append the string version of last_n to the list of bytes
Create new list called last_addr and append each list of bytes to it
Call '.'.join() to make the list of bytes a string again

- Def supernet:

-Uses a list of contiguous Class C addresses
-Returns the network in CIDR notation and the network mask
-Uses convert_to_bin() method to convert Class C addresses into binary:
    -Described above.

-Uses convert_to_decimal() method:
    -Take in an array of four strings representing the bytes of an ip address and convert it back into decimal dot notation
**Parameter**: An array of four binary strings each representing one byte of ip_addr e.g. ['10000100', '11001110', '00010011', '00000111']
**Returns**: The ip address as a string in decimal dot notation e.g. '132.206.19.7'

-Code walkthrough:
For each string in the list use str(int(x,2)) to convert it into a decimal number
Then turn that number into a string e.g. '10000100' -> '132'
Put all converted numbers into a list ["132","206","19","7"]
Call ".".join on the list to merge them into a string separated by "."

# Reflection

- What challenges did you face in doing the assignment and what were your solutions?

  Some of the challenges I faced when doing this assignment were translating the theory of what we went through in lectures to my code. For the tests being run on the ipcalc, subnet and supernet endpoints I understood the theory behind them from the beginning. It just took me some time to be able to leverage my knowledge of how to carry out these tests into my code. I was able to get to grips with it somewhat quickly though, just by researching online and breaking the problems down into simpler forms.

  One of the main problems I faced during the whole process was setting up the post request to receive JSON data. Initially I had no clue at all how to do this but with practice I was able to figure it out somewhat.

- What part of your submission are you particularly proud of?

  I would probably say that the part I'm most proud of is the subnet endpoint. The reason for this is because I felt that the tests being run for this were the most difficult out of any of the others. Maybe not the most difficult, but this is where I struggled the most when writing up the code for the assignment. It was finicky. I definitely spent more time getting this endpoint correct than any of the others. So when I finally got it done, I got a lovely feeling of accomplishment which was nice.

- What did you learn from doing the assignment?

  I learned quite a lot doing this assignment. To start I got a much better understanding of creating endpoints for a server. Obviously this wasn't the first time I had done it, but this was the first time I had been tasked to do this completely by myself without any guidance from a lecturer. I also learned to come up with good variable names. This makes your code easier to read when going back over it and definitely something I didn't emphasise as much at the start of this assignment. I see the value in it more clearly now though.

  I also felt like I learned a lot by including comments in my code to explain exactly what was happening. This is arguably the most valuable thing I learned from this assignment and I'm actually a bit sick to admit that I've only realised how important this is until now. But hey, better late than never!

- What did you think of the assignment?

  I thought this assignment was tough at first, I won't lie. When I first read the project spec I found it to be quite daunting. There was a lot of material in it and I wasn't even really sure where to start. However, after a few days of trying and seeing what works I was able to get a better understanding of how to do it. I found that the more time I spent on this assignment, the easier it appeared to me. I'm now at a point where it doesn't seem anywhere near as difficult to me as it did when the assignment first released. For me, it was important to go through the assignment by myself in my own time to try to get it to work but it was also extremely valuable to go through the assignment with my classmates as well so we could bounce some ideas off each other. Then when working on it by myself I could try to implement these ideas myself. Overall I thought the assignment was very valuable and definitely improved my abilities in programming.