Program #1: Blind SQL injection (WFP2: MongoDB Example #2)

- Consider
  http://<wfp2_site>/mongodb/example2/?search=admin

- Searches for usernames, but we want to steal passwords

- But, if injectable, then we can use conjunctions and try regular expressions against password

- Consider
http://<wfp2_site>/mongodb/example2/?search=admin%27%20%26%26%20this.password.match(/^a/)//+%00
  - Assuming password alphabetic
  - If entry remains, first character of password is 'a'
    - Add 'a' to test condition and move on to second character of password
  - If entry disappears, move on to next candidate letter (e.g. 'b')

- Now, consider
http://<wfp2_site>/mongodb/example2/?search=admin%27%20%26%26%20this.password.match(/^[a-zA-Z]/)//+%00
  - Checks for passwords with alphanumeric first character
  - If entry remains, first character is a letter
    - Split search space in half and try again
  - If entry disappears, first character is not a letter
    - Search half of non-alphabetic characters
  - Continue to narrow regexp until next character of password Found

- Write a Python program that performs a blind SQL injection to obtain the password of the user admin
  - Note that the query is passed in URL parameters and should be accessed via a GET request not a POST

- Rubric
  - Your program must take a single argument from the command line (sys.argv[1]) that represents the IP address or name of <wfp2_site>
    - (e.g. python3 program1.py wfp.oregonctf.org)
  - Your program must implement a binary search algorithm that uses conjunctions and regular expressions within MongoDB

(as shown in the URLs above) against password
○ Your program should be concise and modular
○ Your program should check for errors such as missing arguments or HTTP errors
○ Your program should include some code documentation via Python docstrings