- Python basics:
  - Code indentation instead of brackets
  - No need to initialize the variables, you can directly start using them
  - Comments are preceded by '#'
  - Python is an interpreted language, so you do not need compile the program (Script: technical term for a program in Python). Hence, to execute a script, all you need to do is type in terminal
    - `python script_name.py arguments`
    - and press Enter
- Socket programming in python:
  - The process remains similar to C language.
  - We are supposed to only work on the server side, client would be an Internet browser for this Lab.

  - On the server side
  - Create -> Configure -> ? -> Listen -> Accept
    - Create a TCP server socket
      - `#(AF_INET is used for IPv4 protocols)`
      - `#(SOCK_STREAM is used for TCP)`
      - `serverSocket = socket(AF_INET, SOCK_STREAM)`
    - To avoid the "address already in use" error
      - `serverSocket.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)`
    - Assign a port number
      - `serverPort = int(sys.argv[1])`
    - Bind the socket to server address and server port
      - `# s.bind((HOST, PORT))`
      - `serverSocket.bind(("", serverPort))`
    - Listen to at most 1 connection at a time
      - `serverSocket.listen(1)`
    - Set up a new connection from the client
      - `connectionSocket, addr = serverSocket.accept()`
    - Receives the request message from the client
      - `message = connectionSocket.recv(1024)`

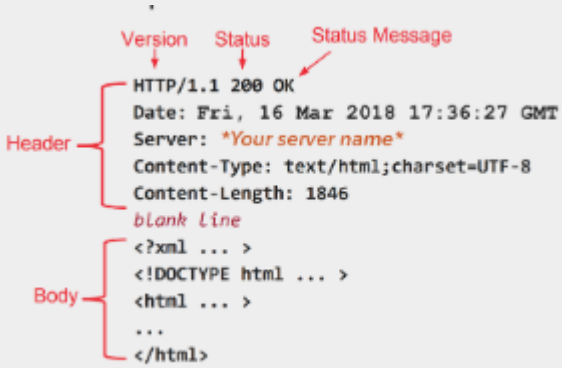
  - Processing the Request:
    - The request-message looks like this
      - *GET /scu_logo.jpg HTTP/1.1*
        Host: localhost:5000
        Connection: keep-alive
        Cache-Control: max-age=0
        Upgrade-Insecure-Requests: 1
        User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.132 Safari/537.36
        Sec-Fetch-Mode: navigate
        Sec-Fetch-User: ?1
        Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,/*;q=0.8,application/signed-exchange;v=b3*
        Sec-Fetch-Site: none
        Accept-Encoding: gzip, deflate, br
        Accept-Language: en-US,en;q=0.9
        Cookie: _xsrf=2|aca22bc5|a4554dfe459add9f2bce8d45d51bad6e|1569533606; username-localhost-8888="2|1:0|10:1571589658|23:username-localhost-

8888|44:NzIzMDRmZTIxYjYxNGNmNWI3YmVhNDdiNjgzMDkxY2I=|218f871960aac71e16ffae1d0101f2c1debfdb9686bc93ba42fb4f646c855dca"

- What we are interested in is the filename requested by the server: "/scu_logo.jpg"
  - Extract the file name using the *split()* function
    - Split function requires the string and the character as an input
  - Using the split function, Python separates the contents of a string delimited by the provided character.
- Remove "/" from the filename
  - "/scu_logo.jpg" -> "scu_logo.jpg"
  - Hint: use " **:** " operator to access a part of a string in Python

- Open the file using:
  - `f = open(filename, 'rb')`
  - Read the whole file and store the contents of the file to *outputdata*
    - `outputdata = f.read()`

- Build the html header, similar to previous lab.



- Hint:
  - `s1 = 'abc'`
  - `s1 = s1 + 'xyz'`
  - result : s1 -> 'abcxyz'
- Determine the file extension of the file requested by the using the split function. The filename and extension are separated by '.'
  - Accordingly, adjust the content type field in the header.

- Finally, use the send() or sendall() function to send the header and then the contents of the file