

# Introduction to Web Science

## Assignment 4

Prof. Dr. Steffen Staab

[staab@uni-koblenz.de](mailto:staab@uni-koblenz.de)

René Pickhardt

[rpickhardt@uni-koblenz.de](mailto:rpickhardt@uni-koblenz.de)

Korok Sengupta

[koroksengupta@uni-koblenz.de](mailto:koroksengupta@uni-koblenz.de)

Institute of Web Science and Technologies

Department of Computer Science

University of Koblenz-Landau

Submission until: November 23, 2016, 10:00 a.m.

Tutorial on: November 25, 2016, 12:00 p.m.

In this assignment we cover two topics: 1) **HTTP** & 2) **Web Content**

For all the assignment questions that require you to write code, make sure to include the code in the answer sheet, along with a separate python file. Where screen shots are required, please add them in the answers directly and not as separate files.

Team Name: Tango

1. Mariya Chkalova  
[mchkalova@uni-koblenz.de](mailto:mchkalova@uni-koblenz.de)
2. Arsenii Smyrnov  
[smyrnov@uni-koblenz.de](mailto:smyrnov@uni-koblenz.de)
3. Simon Schauß  
[sschauss@uni-koblenz.de](mailto:sschauss@uni-koblenz.de)
4. Lukas Härtel  
[lukashaertel@uni-koblenz.de](mailto:lukashaertel@uni-koblenz.de)

# 1 Implementing a simplified HTTP GET Request (15 Points)

The goal of this exercise is to review the hypertext transfer protocol and gain a better understanding of how it works.

Your task is to use the python programming language to create an HTTP client (`httpclient.py`) that takes a URL as a command line argument and is able to download an arbitrary file from the World Wide Web and store it on your hard drive (in the same directory as your python code is running). The program should also print out the complete HTTP header of the response and store the header in a separated file.

Your programm should only use the socket library so that you can open a TCP socket and and sys library to do command line parsing. You can either use `urlparse` lib or your code from assignment 3 in order to process the url which should be retrieved.

Your programm should be able to sucessfully download at least the following files:

1. `http://west.uni-koblenz.de/en/studying/courses/ws1617/introduction-to-web-science`
2. `http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/_IMG0076-Bearbeitet_03.jpg`

**Use of libraries like `httplib`, `urllib`, etc are not allowed in this task.**

## 1.1 Hints:

There will be quite some challenges in order to finnish the task

- Your program only has to be able to process HTTP-responses with status 200 OK.
- Make sure you receive the full response from your TCP socket. (create a function handling this task)
- Sperated the HTTP header from the body (again create a function to do this)
- If a binary file is requested make sure it is not stored in a corrupted way

## 1.2 Example

---

```
1: python httpclient.py http://west.uni-koblenz.de/index.php
2:
3: HTTP/1.1 200 OK
4: Date: Wed, 16 Nov 2016 13:19:19 GMT
5: Server: Apache/2.4.7 (Ubuntu)
6: X-Powered-By: PHP/5.5.9-1ubuntu4.20
7: X-Drupal-Cache: HIT
8: Etag: "1479302344-0"
9: Content-Language: de
```

```
10: X-Frame-Options: SAMEORIGIN
11: X-UA-Compatible: IE=edge,chrome=1
12: X-Generator: Drupal 7 (http://drupal.org)
13: Link: <http://west.uni-koblenz.de/de>; rel="canonical",<http://west.uni-koblenz.d
14: Cache-Control: public, max-age=0
15: Last-Modified: Wed, 16 Nov 2016 13:19:04 GMT
16: Expires: Sun, 19 Nov 1978 05:00:00 GMT
17: Vary: Cookie,Accept-Encoding
18: Connection: close
19: Content-Type: text/html; charset=utf-8
```

---

The header will be printed and stored in index.php.header. The retrieved html document will be stored in index.php

---

**Listing 1** HTTP Client

---

```
import socket
import sys
import gzip
from urllib.parse import urlparse

# Some separators as defined by HTTP standards
CRLF = str(bytes([13, 10]))
TERMINATOR = bytes([13, 10, 13, 10])

def get(url, port=80, buffersize=4096, encoding="UTF-8"):
    """HTTP GETs the given URL and returns the raw data response"""
    data = bytearray()
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as httpsock:
        httpsock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        httpsock.connect((url.hostname, port))

        # Make the request, getting path and query from the location, accepting
        # compress gzip encoding (and identity)
        request = CRLF.join([
            "GET %s%s HTTP/1.0" % (url.path, url.query),
            "Host: %s" % url.netloc,
            "Accept-Encoding: compress, gzip"
        ])

        # Send the entire request
        httpsock.sendall(request.encode(encoding) + TERMINATOR)

        # Read response into buffer until end
        while True:
            buffer = httpsock.recv(buffersize)
            data += buffer
            if not buffer:
                break

    return data

def decoderesponse(data, encoding="UTF-8"):
    """Parses the data of a HTTP response, returns: header, body"""
    # The terminator separates header and body, find it's position
    pos = data.index(TERMINATOR)

    # Split data at that point, only decode header, as it might be a binary
    # file
    return data[:pos].decode(encoding), data[pos + len(TERMINATOR):]

def program(arg):
    """Reads from the argument an URL and downloads it into a header file and
    the raw content"""
    # Extract input and output parameters from argument
    url = urlparse(arg)
    filename = url.path.split("/")[-1]
```

## 2 Download Everything (15 Points)

If you have successfully managed to solve the previous exercise you are able to download a web page from any url. Unfortunately in order to successfully render that very webpage the browser might need to download all the included images

In this exercise you should create a python file (`downloadEverything.py`) which takes two arguments. The first argument should be a name of a locally stored html file. The second argument is the url from which this file was downloaded.

Your program should

1. be able to find a list of urls the images that need to be downloaded for successful rendering the html file.
2. print the list of URLs to the console.
3. call the program from task 1 (or if you couldn't complete task 1 you can call `wget` or use any python lib to fulfill the http request) to download all the necessary images and store them on your hard drive.

**To finish the task you are allowed to use the 're' library for regular expressions and everything that you have been allowed to use in task 1.**

### 2.1 Hints

1. If you couldn't finish the last task you can simulate the relevant behavior by using the program `wget` which is available in almost any UNIX shell.
2. Some files mentioned in the html file might use relative or absolute paths and not fully qualified urls. Those should be fixed to the correct full urls.
3. In case you run problems with constructing urls from relative or absolute file paths you can always check with your web browser how the url is dereferenced.

---

**Listing 2** HTTP Client

---

```
import sys
import os
from urllib.parse import urlparse
import re

HTTP_CLIENT = 'http-client.py'

if __name__ == '__main__':
    file_name = sys.argv[1]
    url = urlparse(sys.argv[2])
    data = ''
    with open(file_name) as file:
        data = file.read()
    img_srcs = map(lambda s: re.findall('src="[^"]*"|src=\'[^\']*\'', s)[0][5:-1], re.findall('<img[^>*/>', data))
    for src in img_srcs:
        if str.startswith(src, 'http') or str.startswith(src, 'https'):
            os.system('python %s %s' % (HTTP_CLIENT, src))
        else:
            os.system('python %s %s://%s%s' % (HTTP_CLIENT, url.scheme, url.hostname, src))
```

---

## Important Notes

### Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignment4/` in your group's repository.
- The name of the group and the names of all participating students must be listed on each submission.
- Solution format: all solutions as *one* PDF document. Programming code has to be submitted as Python code to the github repository. Upload *all* `.py` files of your program! Use UTF-8 as the file encoding. *Other encodings will not be taken into account!*
- Check that your code compiles without errors.
- Make sure your code is formatted to be easy to read.
  - Make sure you code has consistent [indentation](#).
  - Make sure you comment and document your code adequately in English.
  - Choose consistent and intuitive names for your identifiers.
- Do *not* use any accents, spaces or special characters in your filenames.

### Acknowledgment

This latex template was created by Lukas Schmelzeisen for the tutorials of "Web Information Retrieval".

### $\text{\LaTeX}$

Currently the code can only be build using [LuaLaTeX](#), so make sure you have that installed. If on Overleaf, there's an error, go to settings and change the  $\text{\LaTeX}$ engine to LuaLaTeX.