

# 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: Golf

Members : Atique Baig, Mtarji Adam, Deepak Garg

# 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

---

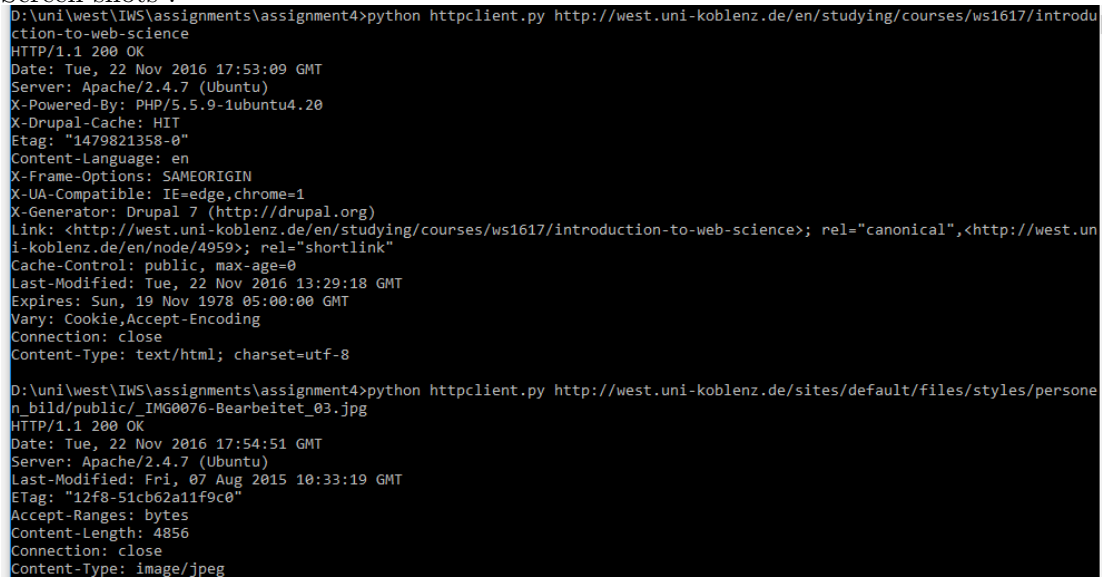
```
1: python httpclient.py http://west.uni-koblenz.de/index.php
2:
3: HTTP/1.1 200 OK
4: Date: Wed, 16 Nov 2016 13:19:19 GMT
5: Server: Apache/2.4.7 (Ubuntu)
6: X-Powered-By: PHP/5.5.9-1ubuntu4.20
7: X-Drupal-Cache: HIT
8: Etag: "1479302344-0"
9: Content-Language: de
```

```
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.de/de>; rel="shortlink"
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

## Answer

Screen shots :



```
D:\uni\west\IWS\assignments\assignment4>python httpclient.py http://west.uni-koblenz.de/en/studying/courses/ws1617/introduction-to-web-science
HTTP/1.1 200 OK
Date: Tue, 22 Nov 2016 17:53:09 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.20
X-Drupal-Cache: HIT
Etag: "1479821358-0"
Content-Language: en
X-Frame-Options: SAMEORIGIN
X-UA-Compatible: IE=edge,chrome=1
X-Generator: Drupal 7 (http://drupal.org)
Link: <http://west.uni-koblenz.de/en/studying/courses/ws1617/introduction-to-web-science>; rel="canonical",<http://west.uni-koblenz.de/en/node/4959>; rel="shortlink"
Cache-Control: public, max-age=0
Last-Modified: Tue, 22 Nov 2016 13:29:18 GMT
Expires: Sun, 19 Nov 1978 05:00:00 GMT
Vary: Cookie,Accept-Encoding
Connection: close
Content-Type: text/html; charset=utf-8

D:\uni\west\IWS\assignments\assignment4>python httpclient.py http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/IMG0076-Bearbeitet_03.jpg
HTTP/1.1 200 OK
Date: Tue, 22 Nov 2016 17:54:51 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Fri, 07 Aug 2015 10:33:19 GMT
Etag: "12f8-51cb62a11f9c0"
Accept-Ranges: bytes
Content-Length: 4856
Connection: close
Content-Type: image/jpeg
```

Figure 1: Running the httpclient.py with url argument

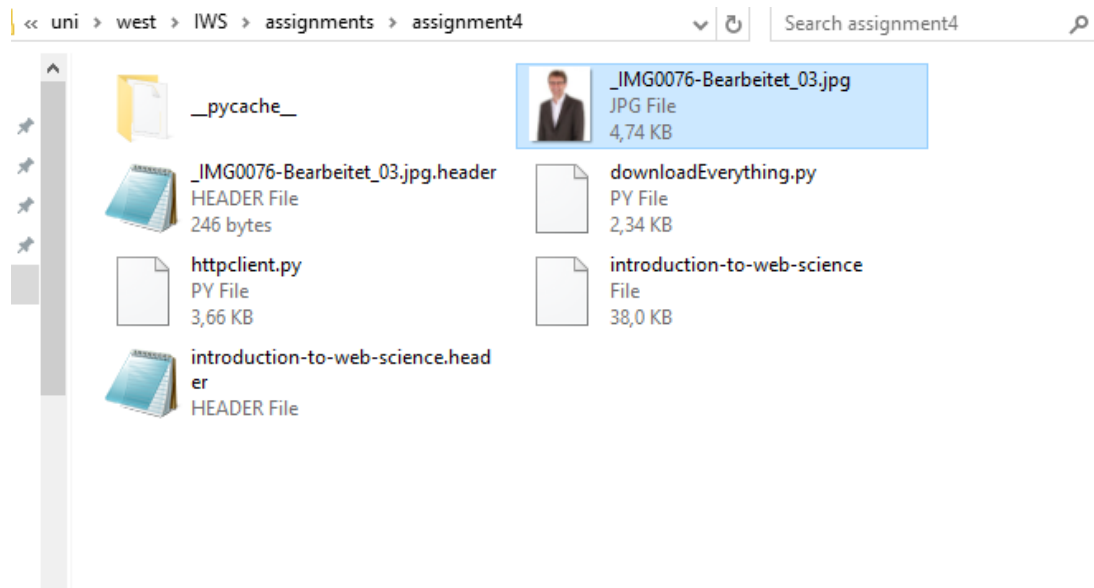


Figure 2: Downloaded files in the working directory

**httpclient.py:**

```
# -*- coding: utf-8 -*-
"""
Introduction to Web Science
Assignment 4
Question 1
Team : golf

Script description :
    This script takes a url as an argument and downloads the resource,
    separating the header from the content as 2 files
"""

# modules
import socket
from urllib.parse import urlparse
import sys

"""
Seperate_header function is used to split the received binary
into header and content(body), we keep the document as binary to be able
to correctly save images
"""
def seperate_header(document):
```

```
# splitting using the b'\r\n\r\n' as separator
# we know that after the header there is always a '\r\n\r\n'
# before the start of the body
header,content=document.split(b'\r\n\r\n',1)
return header,content

"""
receiv_data will retrieve any data from the url as binary
"""
def receiv_data(url):
    # we use the urlparse function to retrieve specific info on the url
    parsed_url = urlparse(url)
    path = parsed_url.path
    # we correcte the path incase its empty
    if path == "":
        path = "/"
    # the domaine will be the HOST used for the socket
    HOST = parsed_url.netloc
    # the path represents the resource to retrieve via GET request
    GET = path
    PORT = 80 # default http port
    # variable used to store the received data
    document=b''

    try:
        # Creating a TCP streaming socket
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # We allow the socket to to bind to a port on TIME_WAIT, this is
        # to reduce the 'address already in use' error while downloading in bulk
        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
        s.connect((HOST, PORT))
        # sending an HTTP GET request for the resource in the path,
        # requiring a 200 ok response, encoding as 'utf8' to stream binary
        s.send(("GET %s HTTP/1.0 200 OK\r\n\r\n"%GET).encode('utf8'))
        # we keep looping on the receive function using 1024 buffer until
        # no more data is received, this ensures that all data is received
        while True:
            data = (s.recv(1024))
            if not data:
                break
            document+=data
        s.shutdown(1)
        s.close()
    except socket.error as exc:
```

```
        print ("Error sending HTTP GET request: %s" % exc)
    return document

"""
function used to read a variable number of files
"""
def read_files(*files):
    for f in files:
        with open(f, 'rb') as file:
            print(file.read().decode('utf8'))

"""
the http_get function calls the other functions in a sequence to correctly
retrieve the data and split it, then write both the header and the content
in 2 separate files
"""
def http_GET(url):

    document = receiv_data(url)
    header, content = separate_header(document)
    # we split the url using '/' and get the last element as file name
    # this will deal dynamically with any resource having extension (.png...)
    file_name = url.split('/')[-1]
    # we append '.header' to the file name so we can store it on a different name
    # we also open the file as 'write binary'
    header_file = open(file_name + '.header', 'wb')
    header_file.write(header)
    header_file.close()
    content_file = open(file_name, 'wb')
    content_file.write(content)
    content_file.close()
    # we read the header file to the console
    read_files(file_name + '.header')

"""
Main script, it checks the number of arguments given to the script,
if it has 1 other argument then calls the http_get function
"""
if len(sys.argv) == 2:
    http_GET(sys.argv[1])
else:
    print('Invalid argument, httpclient requires a valid URL')
```

## 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.

Screen shots :

```
D:\uni\west\IWS\assignments\assignment4>python downloadEverything.py introduction-to-web-science http://west.uni-koblenz.de/en/studying/courses/ws1617/introduction-to-web-science
http://west.uni-koblenz.de/sites/all/themes/westomega/images/uni-logo-original.png
http://west.uni-koblenz.de/sites/all/themes/westomega/images/mail.png
http://west.uni-koblenz.de/sites/all/themes/westomega/images/facebook_blue.png
http://west.uni-koblenz.de/sites/default/files/cms/feed.png
http://west.uni-koblenz.de/sites/all/themes/westomega/images/twitter_blue.png
http://west.uni-koblenz.de/sites/all/themes/westomega/logo.png
http://west.uni-koblenz.de/sites/default/files/styles/headerbild/public/cms/header/west_mood_image_studying_courses.jpg
http://west.uni-koblenz.de/sites/all/modules/languageicons/flags/en.png
http://west.uni-koblenz.de/sites/all/modules/languageicons/flags/de.png
http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/_IMG0076-Bearbeitet_03.jpg
http://west.uni-koblenz.de/sites/default/files/styles/personen_bild/public/aboutus/team/persons/west-team-rene-pickhardt.png
```

**Figure 3:** Running the downloadEverything.py with file and link arguments

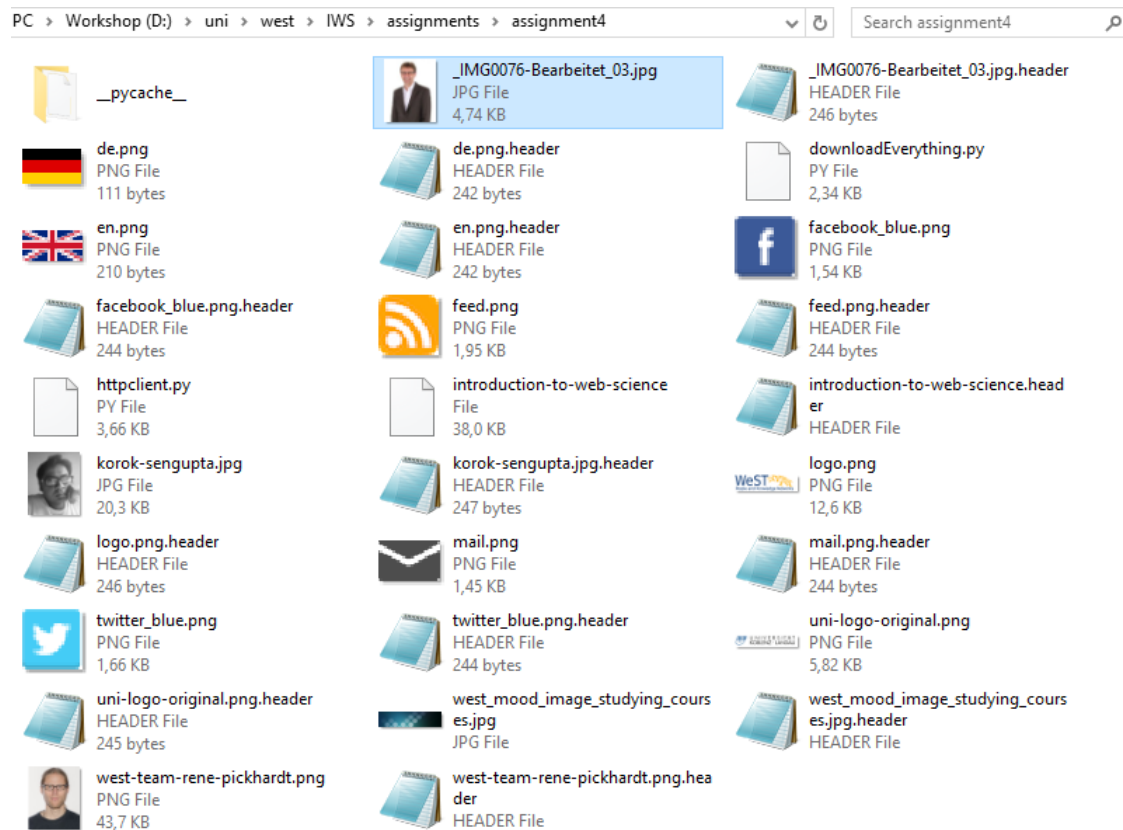


Figure 4: Downloaded images in the working directory

downloadEverything.py:

```
# -*- coding: utf-8 -*-
"""
Introduction to Web Science
Assignment 4
Question 1
Team : golf

Script description :
    This script reads an html file as input and parses it to retrieve
    all links to image files then calls the 'httpclient' script to download them
"""

# modules
from urllib.parse import urlparse
import sys
```



```
import re
import httpclient

"""
main function, takes the name of the html file and the link it was downloaded from
"""
def download(localfile,link):
    parsed_link = urlparse(link)
    # reading from the file
    html_file=open(localfile,'rb')
    # decoding using 'utf8' as it is the most commun encoding used
    # in order to dynamicly know the correct encoding we can probably read the
    # header first, and from the charset value we can deduce the encoding
    html=html_file.read().decode('utf8')
    # using a regular expression to check for .jpg | gif |png, we also use r'
    # for raw string values, and we tell it to extract the value between quotes
    image_urls=re.findall(r'src=[\']?([^\']" >]+(?:jpg|gif|png))',html)
    html_file.close()
    # we will store the complete urls in this variable after evaluating them
    complete_urls=[]
    # now we need to check for absolute paths
    for url in image_urls:
        # if it starts with http, then its already an absolute path
        # we check the negative case instead
        if(not url.startswith('http:')):
            # incase it starts with '/' then we need to append the protocol
            # and the domain to the url
            if(url.startswith('/')) :
                url='http://' + parsed_link.netloc + url
            else:
                # else it means that the resource is in a sub directory of the link
                url=link+url
            complete_urls.append(url)
        else:
            complete_urls.append(url)
    # we print the urls to the console
    print('\n'.join(complete_urls))

    # we call the httpclient.http_get function on every url
    for url in complete_urls:
        httpclient.http_GET(url)

# we check the number of arguments first (we need 2)
if len(sys.argv) == 3:
```

```
        download(sys.argv[1],sys.argv[2])
else:
    print('Invalid arguments')
```

## 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".

### $\LaTeX$

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  $\LaTeX$ engine to LuaLaTeX.