Introduction to Web Science

Assignment 3

Prof. Dr. Steffen Staab
staab@uni-koblenz.de

René Pickhardt
rpickhardt@uni-koblenz.de

Korok Sengupta
koroksengupta@uni-koblenz.de

Institute of Web Science and Technologies
Department of Computer Science
University of Luxembourg

Submission until: November 16, 2016, 10:00 a.m.
Tutorial on: November 18, 2016, 12:00 p.m.

The main objective of this assignment is for you understand different concepts that are associated with the ”Web”. In this assignment we cover two topics: 1) DNS & 2) Internet.

These tasks are not always specific to “Introduction to Web Science”. For all the assignment questions that require you to write a 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: mike

1 DIG Deeper (5 Points)

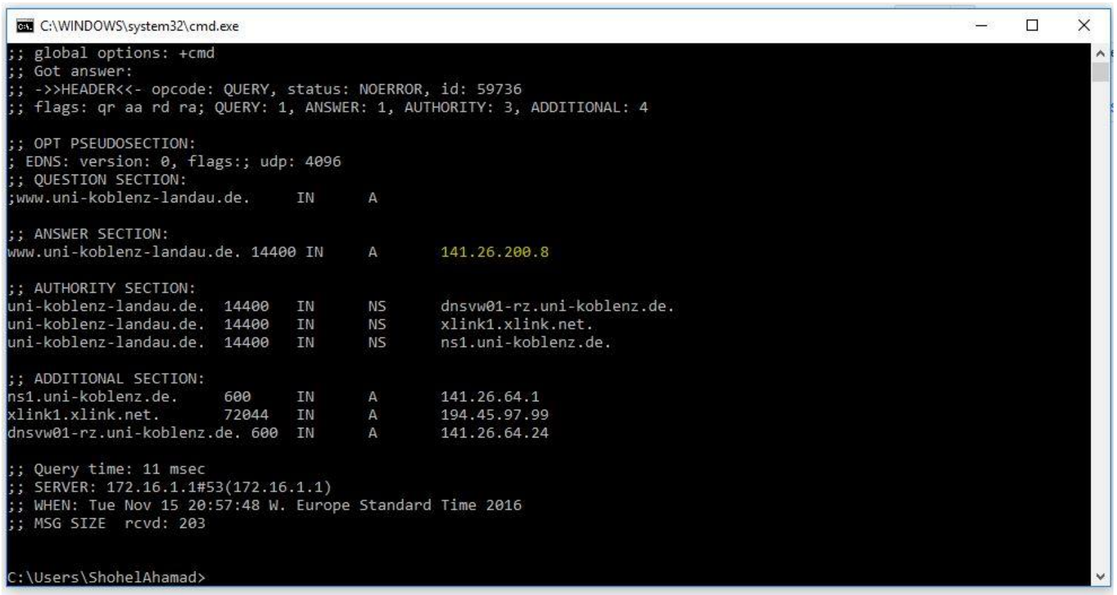
Assignment 1 started with you googling certain basic tools and one of them was "*dig*".

1. Now using that dig command, find the IP address of `www.uni-koblenz-landau.de`
2. In the result, you will find "SOA". What is SOA?
3. Copy the SOA record that you find in your answer sheet and explain each of the components of SOA with regards to your find. Merely integrating answers from the internet wont fetch you points.

Try the experiment once from University network and once from Home network and see if you can find any differences and if so, clarify why.

Answers:

1. The IP address of `www.uni-koblenz-landau.de` is 141.26.200.8

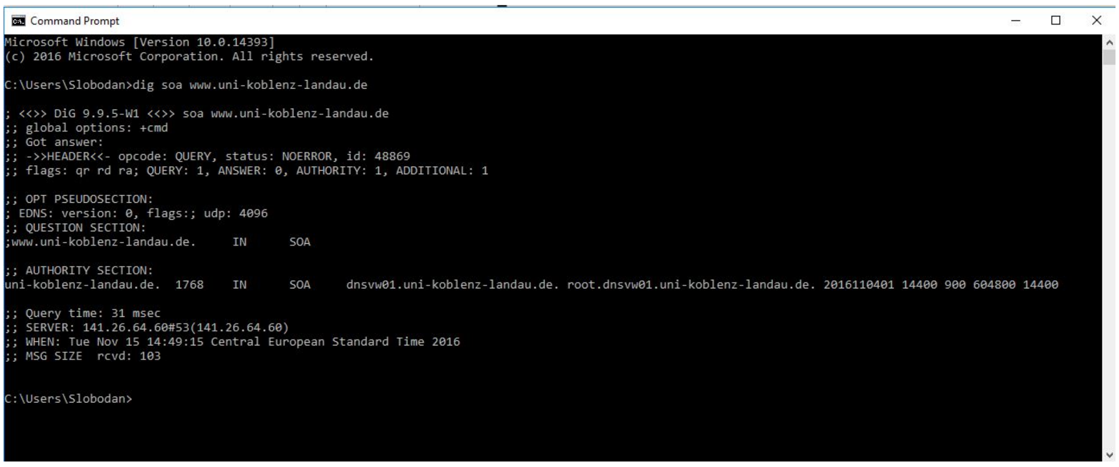


```
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 59736
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 4

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.uni-koblenz-landau.de.     IN      A

;; ANSWER SECTION:
www.uni-koblenz-landau.de. 14400 IN     A       141.26.200.8

;; AUTHORITY SECTION:
uni-koblenz-landau.de.  14400   IN      NS      dnsvw01-rz.uni-koblenz.de.
uni-koblenz-landau.de.  14400   IN      NS      xlink1.xlink.net.
uni-koblenz-landau.de.  14400   IN      NS      ns1.uni-koblenz.de.

;; ADDITIONAL SECTION:
ns1.uni-koblenz.de.     600     IN      A       141.26.64.1
xlink1.xlink.net.       72044   IN      A       194.45.97.99
dnsvw01-rz.uni-koblenz.de. 600  IN      A       141.26.64.24

;; Query time: 11 msec
;; SERVER: 172.16.1.1#53(172.16.1.1)
;; WHEN: Tue Nov 15 20:57:48 W. Europe Standard Time 2016
;; MSG SIZE  rcvd: 203


C:\Users\ShohelAhamad>
```

2. The SOA record defines the global parameters for the zone (domain) in Domain Name System (DNS) and represents the start of a zone and thus administer zone files . It defines which server is primary nameserver, contact information (E-mail), how secondary nameservers get updated, and the default (minimum) Time-To-Live values.

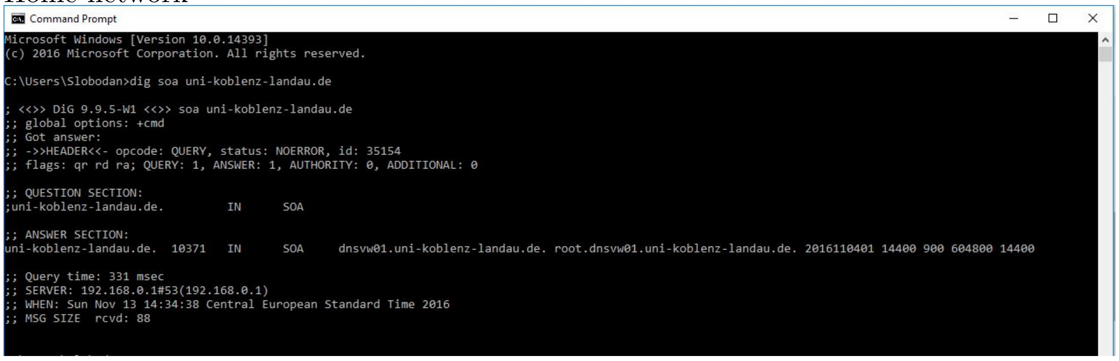
3. SOA record takens from both, Home network and Uni network show the same results for `www.uni-koblenz-landau.de`

   Uni network



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Slobodan>dig soa www.uni-koblenz-landau.de

; <<>> DiG 9.9.5-W1 <<>> soa www.uni-koblenz-landau.de
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 48869
;; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;www.uni-koblenz-landau.de.     IN      SOA

;; AUTHORITY SECTION:
uni-koblenz-landau.de.  1768    IN      SOA     dnsvw01.uni-koblenz-landau.de. root.dnsvw01.uni-koblenz-landau.de. 2016110401 14400 900 604800 14400

;; Query time: 31 msec
;; SERVER: 141.26.64.60#53(141.26.64.60)
;; WHEN: Tue Nov 15 14:49:15 Central European Standard Time 2016
;; MSG SIZE  rcvd: 103


C:\Users\Slobodan>
```

Home network



```
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Slobodan>dig soa uni-koblenz-landau.de

; <<>> DiG 9.9.5-W1 <<>> soa uni-koblenz-landau.de
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35154
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;uni-koblenz-landau.de.         IN      SOA

;; ANSWER SECTION:
uni-koblenz-landau.de.  10371   IN      SOA     dnsvw01.uni-koblenz-landau.de. root.dnsvw01.uni-koblenz-landau.de. 2016110401 14400 900 604800 14400

;; Query time: 331 msec
;; SERVER: 192.168.0.1#53(192.168.0.1)
;; WHEN: Sun Nov 13 14:34:38 Central European Standard Time 2016
;; MSG SIZE  rcvd: 88
```

The SOA record is
uni-koblenz-landau.de. 10371 IN SOA dnsvw01.uni-koblenz-landau.de. root.dnsvw01.uni-koblenz-landau.de. 2016110401 14400 900 604800 14400

a) **uni-koblenz-landau.de.** - is the main name in this zone.

b) **10371** - TTL defines the duration in seconds that the record may be cached by client side programs.In this case this record will be stored almost 3 hours.

c) **IN** - The class IN shows the type of record - IN equates to Internet

d) **SOA** - Shortcuts for Start Of Authority

e) **dnsvw01.uni-koblenz-landau.de.** - Nameserver -Is primary server which holds the zone files. It is an external server in this case, that is why the entire domain name is specified followed by a dot.

f) **root.dnsvw01.uni-koblenz-landau.de.** - This is a DOMAIN NAME that indicates the E-mail address of the person responsible for this zone, and to which email may be sent to report errors or problems. In this case, email is sent to root@dnsvw01.uni-koblenz-landau.de, but written as root.dnsvw01.uni-koblenz-landau.de.

g) **2016110401** - This is a serial number (32-bit integer) that must be incremented on the primary name server whenever a change is made. This number has to increment, whenever any change is made to the Zone file. The standard convention is to use the date of update YYYYMMDDnn, where nn is a revision number in case more than one updates are done in a day. In this case, the last update was made on 04.11.2016. And this was the only update made that day.

h) **14400** - Refresh, This time(in seconds) represents how often a secondary secondary name server gets a copy of the primary zone or sees if the serial number for the zone has increased (so it knows to request a new copy of the data for the zone). In this case, such a request is sent every 4 hours

i) **900** - Retry, 32 bit value in seconds. Defines the time between retries if the secondary name server fails to contact the primary when refresh has expired. In this case the secondary server will wait 15 minutes and then try to re-contact the master server after failing the first time it because it was down. The Retry value (time in seconds) will tell it when to get back.

j) **604800** - Expiry, The number of seconds (32-bit integer) that lets the secondary name server(s) know how long they can hold the information before it is no longer considered valid (authoritative). In this case, the secondary name server will keep the record for 7 days.

k) **14400** - MINIMUM ("Minimum TTL") - The number of seconds that the records in the zone are valid for (time-to-live, or TTL).

2 Exploring DNS (10 Points)

In the first part of this assignment you were asked to develop a simple TCP Client Server. Now, using **that** client server setup. This time a url should be send to the server and the server will split the url into the following:

`http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#InTheDocument`

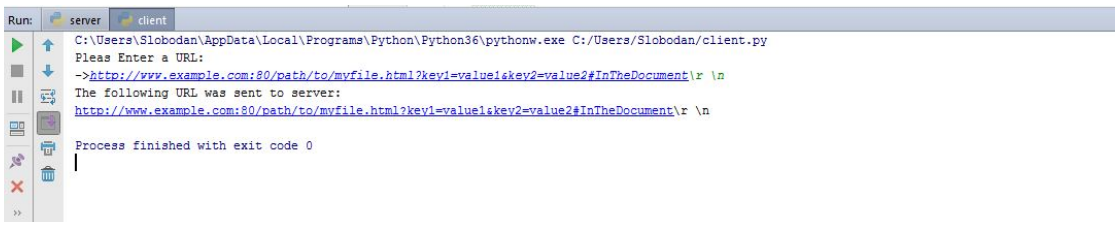
1. Protocol
2. Domain
3. Sub-Domain
4. Port number
5. Path
6. Parameters
7. Fragment

The Protocol for sending the URL will be a string terminated with \r \n.

P.S.: You are **not** allowed to use libraries like `urlparse` for this question. You will also not use "Regular Expressions" for this.

Answer:
Client.py

```python
import socket

#creating the socket
PORT = 8080
s = socket.socket()
s.connect(('localhost', PORT))

#User enters URL until URL ends with  \r\n
url=""
while True:
    print("Pleas Enter a URL:")
    url = input('->')
    if (url[len(url) - 5:len(url)]=="\\r \\n"):
        break

print("The following URL was sent to server: ")
print(url)
s.send(url.encode('utf-8')) #sending url variable over socket
```

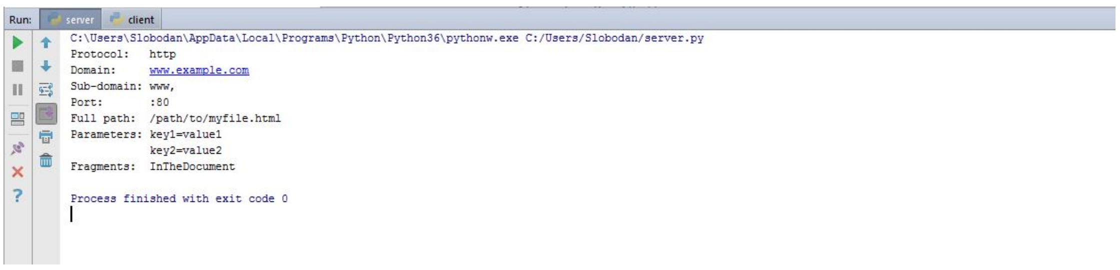


```
C:\Users\Slobodan\AppData\Local\Programs\Python\Python36\pythonw.exe C:/Users/Slobodan/client.py
Pleas Enter a URL:
->http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#InTheDocument\r \n
The following URL was sent to server:
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#InTheDocument\r \n

Process finished with exit code 0
```

Server.py

```
import socket

#creating socket
PORT = 8080
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('localhost', PORT))     #connecting the port with the localhost machine
s.listen(1) # server listens to one client

conn, addr = s.accept()  # accepting the connection
tmp = conn.recv(1024)

url = tmp.decode('utf-8')

i=0
protocol = ""
subdomain=""
domain=""
port=""
full_path=""
param=""
fragment=""


if (len(url)>0):
# splitting the protocol part of the url
    while(url[i:i+3]!= "://"):
        protocol = protocol + url[i]
        i = i + 1
    print("Protocol: \t" + protocol)
    i = i + 3

#splitting the domain part
    array=[]
    while (url[i] != "/" and  url[i] != ":"):
        domain=domain + url[i]
        i=i+1
    print("Domain: \t" +domain)

#splitting the subdoman part
```

```
    sub = domain.split(".")
    j=0
    while (j<len(sub) -2 ):
        subdomain = subdomain + sub[j] + ", "
        j=j+1
    print("Sub-domain:\t" + subdomain)

#splitting the port (if there is any in the url)
    if(url[i]==":"):
        while(url[i]!= "/"):
            port=port+url[i]
            i=i+1
    else:
        port="No port was specified!"
        i=i+1
    print("Port: \t\t" + port)

#splitting the full_path part
    while (url[i] != "?" and url[i:i+5] != "\\r \\n"):
        full_path = full_path + url[i]
        i = i + 1
    print("Full path: \t" + full_path)

#splitting the parameters
    i=i+1
    while (url[i] != "#" and url[i-1:i+4] != "\\r \\n"):
        if(url[i]=="&"):
            param = param + '\n \t\t\t'
        else:
            param = param + url[i]
        i = i + 1
    print("Parameters:\t" + param)

#splitting the fragments
    while (url[i-1:i+4] != "\\r \\n"):
        fragment = fragment + url[i]
        i = i + 1
    print("Fragments: \t" + fragment[1:len(fragment)-1])

else:
    print("No URL was received!")
```



```
C:\Users\Slobodan\AppData\Local\Programs\Python\Python36\pythonw.exe C:/Users/Slobodan/server.py
Protocol:    http
Domain:      www.example.com
Sub-domain: www,
Port:        :80
Full path:   /path/to/myfile.html
Parameters: key1=value1
             key2=value2
Fragments:   InTheDocument

Process finished with exit code 0
```

3 DNS Recursive Query Resolving (5 Points)

You have solved the ”Routing Table” question in Assignment 2. We updated the routing tables once more. resulting in the following tables creating the following topology

Table 1: Routing Table

Router1				**Router2**				**Router3**		
Destination	Next Hop	Interface		Destination	Next Hop	Interface		Destination	Next Hop	Interface
67.0.0.0	67.68.3.1	eth 0		205.30.7.0	205.30.7.1	eth 0		205.30.7.0	205.30.7.2	eth 0
62.0.0.0	62.4.31.7	eth 1		156.3.0.0	156.3.0.6	eth 1		88.0.0.0	88.6.32.1	eth 1
88.0.0.0	88.4.32.6	eth 2		26.0.0.0	26.3.2.1	eth 2		25.0.0.0	25.03.1.2	eth 2
141.71.0.0	141.71.20.1	eth 3		141.71.0.0	141.71.26.3	eth 3		121.0.0.0	121.0.3.1	eth 3
26.0.0.0	141.71.26.3	eth3		67.0.0.0	141.71.20.1	eth 3		156.3.0.0	205.30.7.1	eth 0
156.3.0.0	88.6.32.1	eth 2		62.0.0.0	141.71.20.1	eth 3		26.0.0.0	205.30.7.1	eth 0
205.30.7.0	141.71.26.3	eth 3		88.0.0.0	141.71.20.1	eth 3		141.71.0.0	205.30.7.1	eth 0
25.0.0.0	88.6.32.1	eth 2		25.0.0.0	205.30.7.2	eth 0		67.0.0.0	88.4.32.6	eth 1
121.0.0.0	88.6.32.1	eth 2		121.0.0.0	205.30.7.2	eth 0		62.0.0.0	88.4.32.6	eth 1

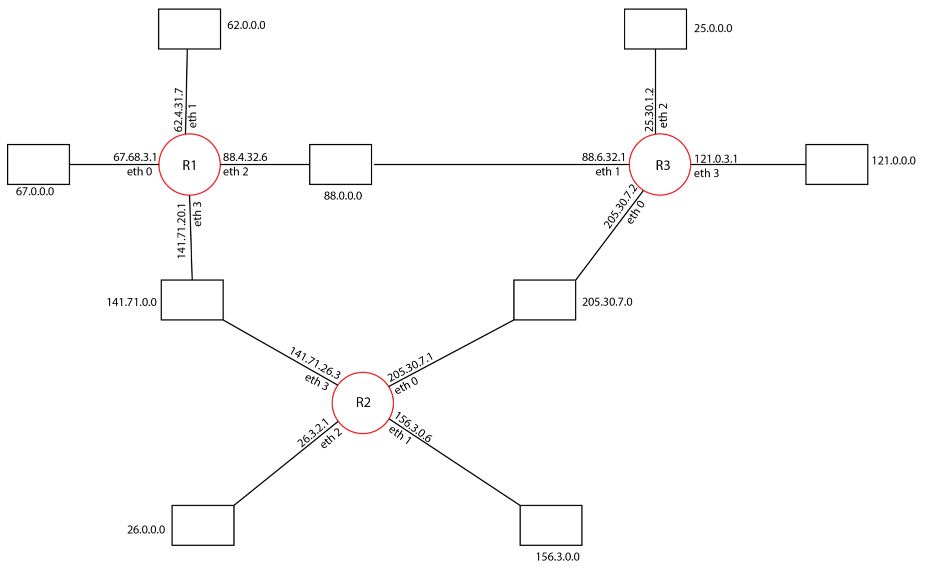


Figure 1: DNS Routing Network

Let us asume a client with the following ip address 67.4.5.2 wants to resolve the following domain `subdomain.webscienceexampledomain.com` using the DNS.

You can further assume the root name server has the IP address of 25.8.2.1 and the name-server for `webscienceexampledomain.com` has the IP address 156.3.20.2. Finally the sub-domain is handled by a name server with the IP of 26.155.36.7.

Please explain how the traffic flows through the network in order to resolve the recursive DNS query. You can assume ARP tables are cached so that no ARP-requests have to be made.

Hint: You can start like this:

67.4.5.2 creates an IP packet with the source address XXXXXX an destination address YYYYY inside there is the DNS request. This IP packet is send as an ethernet frame to ZZZZZ. ZZZZZ receives the frame and forwards the encapsulated IP packet to ....

Also you can assume the DNS requests and responses will fit inside one IP packet. You also don't have to write down the specific DNS requests and responses in hex.

Answer:

If we assume that Router1 acts as a ISP DNS server, then he is responsible for solving recursive quires. When the client (67.4.5.2) wants to resolve the IP address of subdomain.webscienceexampledomain.com using DNS, then he needs to send the request for finding the IP address of this domain to ISP server (Router1).

Router1 then forwards the request to the top level root server (25.8.2.1) to find the IP address of subdomain.webscienceexampledomain.com. Since this name server (25.8.2.1) is not responsible for a query that is not in his zone, that is why this name server (25.8.2.1) sends a respond, stating the IP address of the DNS server (156.3.20.2) that might be responsible for resolving this dns request.

After Router1 (ISP) receives this respond, he sends the query to (156.3.20.2) webscienceexampledomain.com. asking for the IP address of the domain subdomain.webscienceexampledomain.com. Because this DNS server is also not responsible for resolving the stated dns query, he responds in the same way as the top level root name server. He sends the IP address of a dns server that might be responsible for resolving the domain.

The ISP name server receives the respond from 156.3.20.2 and re- assemble new dns query and send it to 26.155.36.7. Now, this dns server, knows the answer, and sends it back to ISP in which case is then forwarder to the client (67.4.5.2).

Afterwards, ISP server keeps track of this record for the next time when the client is going to issue a resolving the same domain.

Important Notes

Submission

- Solutions have to be checked into the github repository. Use the directory name `groupname/assignment3/` 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 LaTeXengine to `LuaLaTeX`.